macromedia®

DREAMWEAVER® 8

Extension de Dreamweaver

Remerciements

Gestion de projet : Charles Nadeau, Robert Berry

Rédaction : Anne Sandstrom

Mise en forme : Anne Szabla, John Hammett

Gestion de la production et de l'édition : Patrice O'Neill et Rosana Francescato

Conception et production : Adam Barnett, Aaron Begley, Paul Benkman, John Francis, Geeta Karmarkar, Paul Rangel, Arena Reed, Mario Reynoso

Gestion de la localisation : Melissa Baerwald

Remerciements particuliers à Jay London, Raymond Lim, Alain Dumesny, Masayo Noda, Kristin Conradi, Yuko Yagi, ainsi qu'à tous les membres des équipes techniques et d'assurance qualité de Dreamweaver.

Première édition : Septembre 2005

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

Table des matières

Introduction

Ce guide décrit l'API (interface de programmation d'application) et la plate-forme Macromedia Dreamweaver 8 qui vous permettent des créer des extensions de Dreamweaver. Il contient des informations sur le fonctionnement de chaque type d'extension, les fonctions API appelées par Dreamweaver pour mettre en œuvre les divers objets, menus, panneaux flottants, comportements de serveur, etc. qui composent Dreamweaver, ainsi qu'un exemple simple de chaque type d'extension. Il explique également comment personnaliser Dreamweaver en modifiant des balises dans divers fichiers HTML et XML pour ajouter des éléments de menu, des types de documents, etc.

Pour ajouter un objet, un menu, un panneau flottant ou toute autre fonctionnalité à Dreamweaver, vous devez encoder les fonctions requises par le type d'extension désirée. Ce guide décrit les arguments transmis par Dreamweaver à ces fonctions, ainsi que les valeurs attendues en retour par le programme.

Pour plus d'informations sur l'utilité et les possibilités offertes par les API JavaScript, utilisables pour effectuer diverses opérations dans vos extensions Dreamweaver, voir le *Guide des API de Dreamweaver*. Si vous envisagez de créer des extensions fonctionnant avec des bases de données, vous pouvez également consulter les sections du manuel *Bien démarrer avec Dreamweaver* consacrées aux connexions de bases de données.

Arrière-plan

La plupart des extensions Dreamweaver sont rédigées en langage HTML et en langage JavaScript. Les instructions présentées ici supposent que vous possédez une bonne maîtrise des éléments suivants : Dreamweaver et programmation JavaScript, HTML et XML. Par ailleurs, si vous implémentez des extensions C, vous devez savoir comment créer et utiliser des bibliothèques de liens dynamiques (DLL) C. Pour rédiger vos propres extensions afin de créer des applications Web, vous devez connaître les langages de script côté serveur et au moins l'une des plates-formes suivantes : Active Server Pages (ASP), ASP.net, PHP: Hypertext Preprocessor (PHP), Macromedia ColdFusion ou Java Server Pages (JSP).

Installation d'une extension

Pour vous familiariser avec la procédure de rédaction d'extensions, vous pouvez si vous le souhaitez consulter les extensions ainsi que les ressources disponibles sur le site Web de Macromedia Exchange (http://www.macromedia.com/go/exchange_fr). L'installation d'une extension existante vous permet de découvrir quelques-uns des outils qui vous seront utiles pour travailler avec vos propres extensions.

Pour installer une extension, procédez de la manière suivante :

1. Téléchargez Extension Manager à partir du site Web des téléchargements de Macromedia (http://www.macromedia.com/fr/software/downloads/), puis installez-le.
2. Connectez-vous au site Web de Macromedia Exchange (http://www.macromedia.com/go/exchange_fr).
3. Sélectionnez l'extension que vous souhaitez récupérer parmi celles disponibles. Cliquez sur le lien de téléchargement pour télécharger le progiciel d'extension.
4. Enregistrez le progiciel d'extension dans le dossier Downloaded Extensions de votre dossier Dreamweaver 8.
5. Depuis Dreamweaver, choisissez Commandes > Gérer les extensions pour lancer Extension Manager, puis dans Extension Manager, choisissez Fichier > Installer extension.

   Extension Manager installe automatiquement l'extension dans Dreamweaver, depuis le dossier Downloaded Extension.

Il est nécessaire de redémarrer Dreamweaver avant de pouvoir utiliser certaines extensions. Si Dreamweaver est en cours d'exécution pendant l'installation de l'extension, il vous sera peut-être demandé de fermer et de redémarrer l'application avant de poursuivre l'installation.

Pour afficher des informations de base sur l'extension à la suite de son installation, depuis Dreamweaver, ouvrez Extension Manager (Commandes > Gérer les extensions).

Création d'une extension

Avant de créer une extension Dreamweaver, consultez le site Web de Macromedia Exchange (www.macromedia.com/go/exchange_fr) pour vérifier que l'extension n'existe pas encore. Si aucune extension existante ne répond à vos besoins, procédez comme suit pour en créer une :

- Déterminez le type d'extension à créer. Pour plus d'informations sur les types d'extension, voir *Types d'extensions Dreamweaver*, page 106.
- Recherchez dans la documentation le type d'extension que vous souhaitez créer. Pour vous familiariser avec la procédure de création du type d'extension requis, il est recommandé de créer l'exemple d'extension simple en vous reportant au chapitre correspondant.

- Déterminez les fichiers à modifier ou créer.
- Le cas échéant, planifiez l'interface utilisateur requise pour l'extension.
- Créez les fichiers requis et enregistrez-les dans les dossiers appropriés.
- Redémarrez Dreamweaver pour qu'il reconnaisse la nouvelle extension.
- Testez l'extension.
- Empaquetez l'extension pour pouvoir la partager avec d'autres utilisateurs. Pour plus d'informations, consultez la section *Utilisation de Extension Manager*, page 116.

Ressources supplémentaires pour les créateurs d'extensions

Pour entrer en contact avec d'autres développeurs d'extensions, vous pouvez rejoindre le forum de discussion consacré à l'extensibilité de Dreamweaver. Pour ce faire, il vous suffit de vous rendre à l'adresse suivante : www.macromedia.com/go/extending_newsgrp.

Nouveautés de Dreamweaver

Dreamweaver 8 comprend les nouvelles fonctionnalités et interfaces extensibles suivantes. A chacune de ces fonctionnalités correspondent de nouvelles fonctions, répertoriées dans le *Guide des API de Dreamweaver*.

- Synchronisation améliorée des sites

  La comparaison des fichiers locaux et distants est plus fiable dans Dreamweaver 8.
- Enrichissements apportés aux fonctions de copie et collage

  Les options de copie et collage ont été simplifiées. Les utilisateurs peuvent maintenant régler le comportement par défaut d'une opération de collage sur texte uniquement.
- Le mappage des liens relatifs à la racine du site a été amélioré.
- Affichage du code

  Dreamweaver permet maintenant aux utilisateurs de réduire ou de développer sélectivement les segments de code.
- Barre d'outils en mode Code

  En mode Code, Dreamweaver propose maintenant une barre d'outils assurant un accès rapide aux commandes fréquemment utilisées.
- Transfert de fichiers en arrière-plan

  Les utilisateurs peuvent désormais continuer à travailler dans Dreamweaver pendant le traitement des tâches associées au serveur.

- Intégration de fonctions de comparaison de fichiers

  Dreamweaver permet maintenant aux utilisateurs de lancer une application de comparaison de fichiers tierce, qui compare deux fichiers locaux, deux fichiers distants ou les versions locale et distante d'un fichier.
- Prise en charge rationalisée des styles CSS

  Les styles CSS et les panneaux CSS correspondants ont été combinés. Le panneau de création s'appelle maintenant CSS, le panneau Styles CSS s'intitule désormais Styles. Le menu Fenêtre comprend à présent une option associée aux styles CSS de document et de sélection. Un bouton Modifier une règle a été intégré à l'inspecteur de propriétés.
- Assistances visuelles associées aux DIV et calques CSS.

  Dreamweaver comprend maintenant des assistances visuelles permettant aux utilisateurs d'afficher leur mise en page CSS.
- Zoom avant et arrière

  Dreamweaver permet désormais aux utilisateurs d'effectuer un zoom avant ou arrière sur leurs pages Web.
- Guides

  Dreamweaver permet maintenant aux utilisateurs de créer des guides dans leurs documents.

Changements dans la documentation

Extension de Dreamweaver a été enrichi comme suit pour aider les créateurs novices à se familiariser avec les extensions.

- Exemples nouveaux et mis à jour

  De nouveaux exemples de rapports et de comportements ont été ajoutés. L'exemple relatif aux inspecteurs de propriétés a été amélioré. La procédure de création de chaque type d'extension est présentée comme un didacticiel, qui vous permet de comprendre le rôle des fichiers et leurs interactions.
- Nouvelle organisation

  Chaque chapitre débute à présent par un tableau qui répertorie les fichiers requis pour créer le type d'extension décrit.

Pour plus d'informations sur les nouvelles fonctions ajoutées à l'API d'utilitaire ou à l'API JavaScript, voir le *Guide des API de Dreamweaver*.

Macromedia Press

Approfondissez votre connaissance de Dreamweaver grâce aux ouvrages de Macromedia Press. Pour connaître les derniers ouvrages rédigés par des experts, rendez-vous à l'adresse www.macromedia.com/go/dw2004_help_mmp.

Fonctions supprimées

Diverses fonctions ont été supprimées de Dreamweaver 8. Pour plus d'informations sur les fonctions éliminées de l'API d'utilitaire et de l'API JavaScript, voir le *Guide des API de Dreamweaver*.

Errata

Vous trouverez une liste des problèmes connus dans la section Extensibility (Extension) du centre de support de Dreamweaver (www.macromedia.com/go/extending_errata).

Conventions utilisées dans ce manuel

Ce manuel utilise les conventions typographiques suivantes :

- La police de `code` indique des fragments de code et des constantes d'API, notamment des noms de classe, des noms de méthodes, des noms de fonctions, des noms de type, des scripts, des instructions SQL et des noms de balises et d'attributs HTML et XML.
- La police de *`code en italique`* identifie les éléments remplaçables dans le code.
- Le symbole de continuation (¬) indique qu'une longue ligne de code a été fractionnée sur deux lignes ou plus. En raison des limites de marge du format de ce guide, une ligne de code continue doit ici être coupée. Lorsque vous copiez les lignes de code, supprimez le symbole de continuation et entrez-les comme une seule ligne.
- Les accolades ({ }) placées avant et après un argument de fonction indiquent que cet argument est facultatif.
- Vous pouvez remplacer le préfixe `dreamweaver.` présent dans le nom de certaines fonctions (par exemple, `dreamweaver.`*`funcname)`* par dw. (`dw.`*`funcname`*) lorsque vous rédigez du code. Ce manuel utilise le préfixe `dreamweaver.` complet dans les définitions de fonctions et dans l'index. De nombreux exemples utilisent néanmoins le préfixe court (`dw.`)

Conventions de noms utilisées dans ce manuel :

- Vous — le développeur responsable de la rédaction des extensions

- L'utilisateur — la personne utilisant Dreamweaver
- Le visiteur — la personne qui visualise la page Web créée par l'utilisateur.

PARTIE 1

Personnalisation de Dreamweaver

1

Vous pouvez personnaliser Macromedia Dreamweaver 8 pour l'adapter à vos besoins de développement Web, y compris modifier les paramètres dans des boîtes de dialogue, définir vos préférences dans divers domaines et modifier les raccourcis clavier. Vous pouvez en outre personnaliser les indicateurs et colorations de code en mode Code, le profil CSS (feuille de style en cascade) et la mise en forme HTML par défaut de Dreamweaver.

CHAPITRE 1

Personnalisation de Dreamweaver

1

Outre créer et utiliser des extensions Dreamweaver, vous pouvez personnaliser Dreamweaver de différentes manières afin de l'adapter au mieux à vos besoins.

Techniques de personnalisation de Dreamweaver

Il existe plusieurs approches pour personnaliser Dreamweaver. Certaines sont présentées dans *Utilisation de Dreamweaver*. Ces approches vous permettent de personnaliser l'espace de travail. Vous pouvez également modifier les paramètres dans les boîtes de dialogue de Dreamweaver. Vous pouvez définir des préférences dans divers domaines (accessibilité, coloration du code, polices, surbrillance et prévisualisation dans un navigateur), à l'aide du panneau Préférences (Edition > Préférences). Vous pouvez également modifier les raccourcis clavier à l'aide de l'éditeur de raccourcis clavier (Edition > Raccourcis clavier).

La liste suivante présente les diverses méthodes permettant de personnaliser Dreamweaver par la modification des fichiers de configuration :

- Modifier l'ordre des objets dans la barre Insérer, créer de nouveaux onglets pour réorganiser les objets ou ajouter de nouveaux objets. Voir *Modification de la barre Insérer*, page 156.
- Changer le nom des éléments de menu, ajouter de nouvelles commandes aux menus et supprimer des commandes existantes des menus. Voir Chapitre 8, “Menus et commandes de menu,” on page 191.
- Modifier les profils de navigateurs ou en créer de nouveaux. Voir *Utilisation des profils de navigateurs*, page 32.
- Modifier l'affichage des balises propriétaires (balises ASP et JSP comprises) dans la fenêtre de document en mode Création. Voir *Personnalisation de l'interprétation de balises propriétaires*, page 21.

Vous pouvez en outre adapter Dreamweaver à vos besoins en procédant comme suit :

Personnalisation de documents par défaut

Le dossier DocumentTypes/NewDocuments contient, par défaut, un document (vierge) de chaque type de document que vous pouvez créer à l'aide de Dreamweaver. Lorsque vous créez un nouveau document vierge en sélectionnant Fichier > Nouveau et en sélectionnant un élément dans les listes Page de base, Page dynamique ou Autres catégories, Dreamweaver base le nouveau document sur le document par défaut correspondant dans ce dossier. Pour modifier les éléments par défaut d'un type de document donné, modifiez le document correspondant dans ce dossier.

REMARQUE

Si vous souhaitez que toutes les pages de votre site contiennent certains éléments communs (par exemple une mention sur le copyright) ou une présentation constante, il est préférable d'utiliser des modèles et des éléments de bibliothèque plutôt que de changer les documents par défaut. (Pour plus d'informations sur les modèles et les éléments de bibliothèque, voir Utilisation de Dreamweaver.)

Personnalisation de conceptions de pages

Dreamweaver propose divers éléments prédéfinis : feuilles de style CSS, jeux de cadres et conceptions de pages. Vous pouvez créer des pages basées sur ces conceptions en cliquant sur Fichier > Nouveau.

Pour personnaliser les conceptions disponibles, modifiez les fichiers présents dans les dossiers BuiltIn/css, BuiltIn/framesets, BuiltIn/Templates et BuiltIn/TemplatesAccessible

REMARQUE

Les conceptions répertoriées dans les catégories Conception de page et Conception de page (Accessibilité) sont les fichiers modèles de Dreamweaver. Pour plus d'informations sur les modèles, voir Utilisation de Dreamweaver.

Vous pouvez également créer des conceptions de pages en ajoutant des fichiers dans les sous-dossiers du dossier BuiltIn. Pour qu'une description du fichier s'affiche dans la boîte de dialogue Nouveau document, créez un fichier Design Notes (dans le dossier _notes approprié) correspondant au fichier de conception de page.

Personnalisation de l'aspect des boîtes de dialogue

La mise en forme des boîtes de dialogue pour les objets, les commandes et les comportements sont définis comme des formulaires HTML. Ils résident dans des fichiers HTML du dossier Configuration au sein du dossier d'application Dreamweaver. Ces formulaires peuvent être modifiés comme tout autre formulaire Dreamweaver. Pour plus d'informations, voir *Utilisation de Dreamweaver*.

REMARQUE

N'oubliez pas que, dans un système d'exploitation multiutilisateur, vous devez modifier les copies des fichiers de configuration contenues dans votre dossier de configuration utilisateur plutôt que les fichiers Configuration de Dreamweaver. Pour plus d'informations, consultez la section *Dossiers de configuration multiutilisateur*, page 111.

Pour modifier l'aspect d'une boîte de dialogue :

1. Dans Dreamweaver, sélectionnez Edition > Préférences, puis la catégorie Correction du code.
2. Désélectionnez l'option Renommer les éléments de formulaire lors du collage.

   Cette opération vous assure que les éléments de formulaire conserveront leur nom original une fois ceux-ci copiés et collés.
3. Cliquez sur OK pour fermer la boîte de dialogue Préférences.
4. Sur votre disque, localisez le fichier HTM approprié dans le dossier Configuration/Objects, Configuration/Commands ou Configuration/Behaviors.
5. Faites une copie du fichier dans un dossier autre que le dossier Configuration.
6. Ouvrez cette copie dans Dreamweaver, modifiez le formulaire et enregistrez le document.
7. Quittez Dreamweaver.
8. Copiez le fichier modifié à la place de l'original dans le dossier Configuration. Il est toutefois conseillé de conserver une sauvegarde du fichier original pour pouvoir l'utiliser à nouveau en cas de besoin.
9. Redémarrez Dreamweaver pour constater les changements.

Il est recommandé de ne modifier que l'aspect de la boîte de dialogue et non son fonctionnement. Le type des éléments de formulaire doit être identique, tout comme les noms, afin que l'information obtenue par Dreamweaver à partir de la boîte de dialogue puisse être utilisée de la même manière.

Par exemple, l'objet Commentaire prend le texte entré dans une zone de texte de boîte de dialogue et utilise une fonction JavaScript simple pour transformer ce texte en commentaire HTML et l'insérer dans votre document. Le formulaire qui décrit la boîte de dialogue se trouve dans le fichier Comment.htm dans le dossier Configuration/Objects/Invisibles. Vous pouvez ouvrir ce fichier et modifier la taille ou d'autres attributs dans la zone de texte, mais si vous supprimez intégralement la balise `textarea` ou que vous modifiez la valeur de son attribut name, l'objet Commentaire ne fonctionne plus correctement.

Modification du type de fichier par défaut

Par défaut, Dreamweaver affiche tous les types de fichiers reconnus dans la boîte de dialogue Fichier > Ouvrir. Vous pouvez utiliser un menu contextuel dans cette boîte de dialogue pour limiter l'affichage de certains types de fichiers. Si la plupart de vos travaux impliquent un type de fichier spécifique (par exemple, les fichiers ASP), vous pouvez modifier l'affichage par défaut. Le type de fichier indiqué sur la première ligne du fichier Extensions.txt de Dreamweaver devient le type par défaut.

REMARQUE

Pour afficher tous les types de fichiers dans la boîte de dialogue Fichier › Ouvrir (y compris les fichiers que Dreamweaver ne peut pas ouvrir), vous devez sélectionner Tous les fichiers (*.*). Ceci n'est pas identique à Tous les documents, qui ne répertorie que les fichiers que Dreamweaver peut ouvrir.

Pour modifier le type de fichier par défaut de Dreamweaver dans Fichier › Ouvrir :

1. Créez une copie de sauvegarde du fichier Extensions.txt dans le dossier Configuration.
2. Ouvrez le fichier Extensions.txt dans Dreamweaver ou dans un éditeur de texte.
3. Coupez la ligne correspondant au nouveau type par défaut et collez-la en tête du fichier, en première ligne.
4. Enregistrez le fichier.
5. Redémarrez Dreamweaver.

   Pour voir le nouveau type par défaut, sélectionnez Fichier > Ouvrir, puis examinez le menu contextuel de types de fichiers.

Pour ajouter de nouveaux types de fichier dans le menu de la boîte de dialogue Fichier > Ouvrir :

1. Créez une copie de sauvegarde du fichier Extensions.txt dans le dossier Configuration.
2. Ouvrez le fichier Extensions.txt dans Dreamweaver ou dans un éditeur de texte.
3. Ajoutez une nouvelle ligne pour chaque nouveau type de fichier.
   a. Tapez en majuscules les extensions de fichier gérées par le nouveau type de fichier, en les séparant par une virgule.
   b. Ajoutez le signe deux points et une brève description, qui apparaîtra dans le menu contextuel de types de fichiers associé à la boîte de dialogue qui s'affiche lorsque vous sélectionnez Fichier > Ouvrir.

   Par exemple, pour les fichiers JPEG, entrez les informations suivantes :

   ```
   JPG,JPEG,JFIF:Fichiers image JPEG
   ```
4. Enregistrez le fichier.
5. Redémarrez Dreamweaver.

   Pour observer les modifications, sélectionnez Fichier > Ouvrir, puis cliquez sur le menu contextuel de types de fichiers.

Personnalisation de l'interprétation de balises propriétaires

Les technologies côté serveur telles que ASP, Macromedia ColdFusion, JSP et PHP utilisent du code spécial non HTML au sein des fichiers HTML. Les serveurs créent et servent du contenu HTML basé sur ce code. Lorsque Dreamweaver rencontre des balises non HTML, il les compare aux informations contenues dans les fichiers de balises propriétaires, lesquels définissent comment lire et afficher ces balises non HTML.

Par exemple, les fichiers ASP contiennent (outre le code HTML habituel) du code ASP que le serveur doit interpréter. Le code ASP ressemble à une balise HTML, mais il est signalé par des délimiteurs : il commence par <% et s'arrête par %>. Le dossier Configuration/ThirdPartyTags de Dreamweaver contient un fichier nommé Tags.xml. Ce fichier décrit le format des diverses balises propriétaires, dont le code ASP, et décrit comment Dreamweaver doit afficher ce code. La façon dont le code ASP est spécifié dans le fichier Tags.xml conduit Dreamweaver à ne pas tenter d'interpréter le code situé entre les délimiteurs. A la place, en mode Création, seule une icône indiquant la présence de code ASP s'affiche. Vos propres fichiers de données de balises peuvent définir l'affichage et la lecture de vos balises par Dreamweaver. Créez un nouveau fichier de données de balises pour chaque jeu de balises afin d'indiquer à Dreamweaver comment les afficher.

REMARQUE Cette section vous explique comment définir l'affichage d'une balise personnalisée par Dreamweaver, mais ne décrit pas comment modifier le contenu ou les propriétés d'une balise personnalisée. Pour plus d'informations sur la création d'un inspecteur de propriétés permettant de vérifier et modifier les propriétés d'une balise personnalisée, voir Chapitre 12, “Inspecteurs de propriétés,” on page 291.

Chaque fichier de données de balises définit le nom, le type, le modèle de contenu, le modèle de rendu et l'icône pour une ou plusieurs balises. Vous pouvez créer un nombre illimité de fichiers de données de balises, mais tous doivent se trouver dans le dossier Configuration/ThirdPartyTags afin d'être consultés et traités par Dreamweaver. Les fichiers de données de balises portent l'extension .xml.

CONSEIL Si vous travaillez sur plusieurs sites distincts à la fois (par exemple, en tant que développeur indépendant), vous pouvez regrouper toutes les spécifications de balises relatives à un site dans un fichier. Il suffit ensuite de remettre ce fichier de données de balises avec les icônes personnalisées et les inspecteurs de propriétés aux responsables qui géreront le site.

Vous pouvez définir une spécification de balise à l'aide d'une balise XML nommée `tagspec`. Par exemple, le code suivant décrit les spécifications d'une balise nommée `happy` :

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false"
  content_model="marker_model" icon="happy.gif" icon_width="18"
  icon_height="18"></tagspec>
```

Vous pouvez définir deux types de balises à l'aide de `tagspec` :

- Balises de type HTML standard

  `happy` est un exemple de balise de type HTML standard. Elle débute par une balise d'ouverture `<happy>`, des données sont placées entre les balises d'ouverture et de fermeture, et elle se termine par une balise `</happy>` de fermeture.
- Balises délimitées par des chaînes

Les balises délimitées par des chaînes débutent par une chaîne et se terminent par une autre chaîne. Elles se comportent comme des balises HTML vides (telle `img`), car elles n'entourent pas les données de balises et ne comportent pas de balises de fermeture. Si la balise `happy` était délimitée par des chaînes, les spécifications de balises incluraient les attributs `start_string` et `end_string`. Une balise ASP est une balise délimitée par des chaînes. Elle commence par la chaîne `<%` et se termine par la chaîne `%>`, sans balise de fermeture.

Les informations suivantes décrivent les attributs et les valeurs valides de la balise `tagspec`. Les attributs marqués d'un astérisque (*) sont ignorés dans le cadre des balises délimitées par des chaînes. Les attributs optionnels sont définis par des accolades (`{}`) dans la liste des attributs ; les attributs ne comportant pas d'accolades sont donc requis.

<tagspec>

Description

Fournit des informations à propos d'une balise propriétaire.

Attributs

```
tag_name, {tag_type}, {render_contents}, {content_model}, {start_string},
  {end_string}, {detect_in_attribute}, {parse_attributes}, icon,
  icon_width, icon_height, {equivalent_tag}, {is_visual}, {server_model}
```

- `tag_name` est le nom de la balise personnalisée. Pour les balises délimitées par des chaînes, `tag_name` n'est utilisé que pour déterminer si un inspecteur de propriétés donné peut être utilisé pour la balise. Si la première ligne de l'inspecteur de propriétés contient ce nom de balise encadré par des astérisques, l'inspecteur peut être utilisé pour les balises de ce type. Par exemple, le nom de balise associé au code ASP correspond à `ASP`. Les inspecteurs de propriétés pouvant examiner le code ASP doivent comporter la mention `*ASP*` sur la première ligne. Pour plus d'informations sur l'inspecteur de propriétés API, voir Chapitre 12, “Inspecteurs de propriétés,” on page 291.
- `tag_type` détermine si la balise est vide (telle que `img`) ou si un contenu est présent entre les balises d'ouverture et de fermeture (telles que `code`). Cet attribut est requis pour les balises normales (non délimitées par des chaînes). Cet élément est ignoré pour les balises délimitées par des chaînes, ces dernières étant toujours vides. Les valeurs valides sont `"empty"` et `"nonempty"`.

- `render_contents` détermine si le contenu de la balise doit s'afficher en mode Création ou s'il est remplacé par l'icône spécifiée. Cet attribut est requis pour les balises nonempty et n'est pas pris en compte pour les balises empty (les balises empty sont vides de contenu). Cet attribut ne s'applique qu'aux balises affichées hors des attributs. Le contenu des balises imbriquées à l'intérieur des valeurs d'attributs d'autres balises n'est pas affiché. Les valeurs valides sont `"true"` et `"false"`.
- `content_model` décrit les différents types de contenu que la balise peut contenir et l'emplacement où la balise peut s'afficher dans un fichier HTML. Les valeurs valides sont `"block_model"`, `"head_model"`, `"marker_model"` et `"script_model"` :
  - `block_model` spécifie que la balise peut contenir des éléments de niveau de bloc comme `div` et `p` et que la balise peut apparaître uniquement dans la section body ou dans d'autres balises au contenu body, telles que `div`, `layer` et `td`.
  - `head_model` spécifie que le contenu de la balise peut être composé de texte et que la balise peut uniquement apparaître dans la section `HEAD`.
  - `marker_model` spécifie que la balise peut contenir tout code HTML valide et peut se trouver à tout endroit du fichier HTML. Le validateur HTML de Dreamweaver ignore les balises spécifiées comme marker_model. Néanmoins, le validateur n'ignore pas le contenu de ces balises. En conséquence, même si la balise peut apparaître à n'importe quel emplacement, le contenu de la balise peut corrompre le document HTML à certains endroits. Par exemple, du texte simple ne peut apparaître dans la section head d'un document (à l'exception des éléments head valides). Il est donc impossible de placer une balise marker_model qui contient du texte simple dans la section head. Pour placer une balise personnalisée dans la section head, définissez le modèle de contenu de la balise comme head_model au lieu de marker_model. Utilisez marker_model pour les balises qui doivent s'afficher en ligne (dans un élément de niveau de bloc, comme p ou div, par exemple, dans un paragraphe). N'utilisez pas ce modèle si la balise doit être affichée seule dans un paragraphe, encadrée par des sauts de ligne.
  - `script_model` permet un emplacement libre de la balise entre les balises d'ouverture et de fermeture d'un document. Lorsque Dreamweaver rencontre une balise de ce modèle, il ignore totalement le contenu de la balise. Ce dernier est utilisé pour le marquage (comme certaines balises ColdFusion) que Dreamweaver ne doit pas analyser.
- `start_string` spécifie un délimiteur qui marque le début d'une balise délimitée par des chaînes. Les balises délimitées par des chaînes peuvent être présentes en tout point du document pouvant contenir un commentaire. Dreamweaver n'analyse pas les balises et ne décode pas les entités ou URL comprises entre `start_string` et `end_string`. Cet attribut est requis si `end_string` est spécifié.

- `end_string` spécifie un délimiteur qui marque la fin d'une balise délimitée par une chaîne. Cet attribut est requis si `start_string` est spécifié.
- `detect_in_attribute` indique si les éléments contenus entre `start_string` et `end_string` (ou entre les balises d'ouverture et de fermeture si ces chaînes ne sont pas définies) doivent être ignorés même si ces chaînes apparaissent dans les valeurs ou noms d'attributs. Vous devez généralement régler la valeur sur `"true"` pour les balises délimitées par des chaînes. La valeur par défaut est `"false"`. Ainsi, les balises ASP sont parfois imbriquées dans des valeurs d'attributs et contiennent parfois des guillemets ("). Parce que la balise ASP spécifie `detect_in_attribute="true"`, Dreamweaver ne tient pas compte des balises ASP, y compris des guillemets internes, lorsqu'elles sont imbriquées dans des valeurs d'attributs.
- `parse_attributes` indique si les attributs de la balise doivent être analysés. Si la valeur définie est `"true"` (par défaut), Dreamweaver analyse les attributs. Si elle est définie sur `"false"`, Dreamweaver ignore tous les éléments jusqu'au crochet situé hors des guillemets. Par exemple, cet attribut doit être réglé sur `"false"` pour les balises telles que `cfif` (par exemple, `<cfif a is 1>`, que Dreamweaver ne peut pas analyser comme ensemble de paires nom d'attribut/valeur).
- `icon` spécifie le chemin et le nom de fichier de l'icône associée à la balise. Cet attribut est requis pour les balises empty ainsi que pour les balises nonempty dont le contenu ne s'affiche pas dans la fenêtre de document en mode Création.
- `icon_width` spécifie la largeur de l'icône en pixels.
- `icon_height` spécifie la hauteur de l'icône en pixels.
- `equivalent_tag` spécifie des équivalents HTML simples pour certaines balises liées à des formulaires ColdFusion. Ceci ne doit pas être utilisé avec d'autres balises.
- `is_visual` indique si la balise a un impact visuel sur la page. Par exemple, la balise ColdFusion `cfgraph` ne spécifie aucune valeur pour `is_visual` ; la valeur par défaut `"true"` est donc appliquée. La valeur `is_visual` de la balise ColdFusion `cfset` est définie sur `false`. La visibilité des balises de marquage de serveur est contrôlée par la catégorie Eléments invisibles dans la boîte de dialogue Préférences. La visibilité des balises de marquage de serveur visuel peut être définie indépendamment de celle des balises de marquage de serveur non visuel.

- `server_model`, si spécifié, indique que la balise `tagspec` s'applique uniquement aux pages appartenant au modèle de serveur spécifié. Si `server_model` n'est pas spécifié, la balise `tagspec` s'applique à toutes les pages. Par exemple, les délimiteurs des balises ASP et JSP sont identiques, mais la balise `tagspec` pour JSP spécifie un modèle de serveur `"JSP"`. Par conséquent, lorsque Dreamweaver rencontre ce code avec les délimiteurs adéquats sur une page JSP, il affiche une icône JSP. Lorsque ce code est rencontré sur une page autre qu'une page JSP, une icône ASP s'affiche.

Contenu

Aucun (balise vide).

Contenant

Néant

Exemple

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false"
  content_model="marker_model" icon="happy.gif" icon_width="18"
  icon_height="18"></tagspec>
```

Affichage des balises personnalisées dans le mode Création

L'affichage des balises personnalisées dans la fenêtre de document en mode Création dépend des valeurs entrées pour les attributs `tag_type` et `render_contents` de la balise `tagspec`. (Voir *Personnalisation de l'interprétation de balises propriétaires*, page 21.) Si la valeur de `tag_type` est `"empty"`, l'icône spécifiée dans l'attribut `icon` s'affiche. Si la valeur de `tag_type` est `"nonempty"` mais que la valeur de `render_contents` est `"false"`, l'icône s'affiche de la même façon que pour une balise vide. L'exemple suivant indique comment une instance de la balise `happy` définie antérieurement peut apparaître dans le code HTML :

```
<p>This is a paragraph that includes an instance of the <code>happy</code>
tag (<happy>Joe</happy>).</p>
```

Parce que `render_contents` est réglé sur `"false"` dans la spécification de balise, le contenu de la balise `happy` (le mot `Joe`) ne s'affiche pas. Les balises de début et de fin et leur contenu sont alors remplacés par une icône unique.

Pour les balises nonempty dont la valeur `render_contents` est réglée sur `"true"`, l'icône ne s'affiche pas dans le mode Création. C'est le contenu inséré entre les balises d'ouverture et de fermeture qui s'affiche (par exemple, le texte contenu entre les balises dans `<mytag>Ceci est le contenu inséré entre les balises d'ouverture et de fermeture</mytag>`). Si l'option Affichage > Eléments invisibles est activée, le contenu est mis en surbrillance à l'aide de balises propriétaires, comme spécifié dans les préférences de surbrillance (la mise en surbrillance ne s'applique qu'aux balises définies dans les fichiers de données de balises).

Pour modifier la couleur de surbrillance des balises propriétaires :

1. Sélectionnez Edition > Préférences, puis la catégorie Surbrillance.
2. Cliquez sur la zone de sélection de couleurs de balises propriétaires pour afficher le sélecteur de couleur.
3. Sélectionnez une couleur, puis cliquez sur OK pour fermer la boîte de dialogue Préférences. Pour plus d'informations sur la sélection d'une couleur, voir *Utilisation de Dreamweaver*.

Comment éviter la modification de balises propriétaires

Dreamweaver corrige certains types d'erreurs dans le code HTML. Pour plus d'informations, voir *Utilisation de Dreamweaver*. Par défaut, Dreamweaver évite toute modification du code HTML dans les fichiers portant certaines extensions, comme .asp (ASP), .cfm (ColdFusion), .jsp (JSP) et .php (PHP). Ce paramètre est défini pour éviter toute modification accidentelle du code contenu dans de telles balises non HTML. Vous pouvez modifier le comportement de correction de Dreamweaver par défaut afin que le code HTML soit modifié lors de l'ouverture de tels fichiers. Vous pouvez également ajouter d'autres types de fichiers auxquels Dreamweaver n'apportera pas de modifications.

Dreamweaver encode certains caractères spéciaux en les remplaçant par des valeurs numériques lorsque vous les entrez dans l'inspecteur de propriétés. Il est normalement préférable de laisser Dreamweaver effectuer cet encodage car, ainsi, les caractères spéciaux seront reconnus par un nombre plus important de plates-formes et de navigateurs. Néanmoins, cette opération d'encodage peut interférer avec les balises propriétaires. Vous pouvez donc modifier le comportement de Dreamweaver concernant l'encodage des balises lorsque vous travaillez avec des fichiers comportant des balises propriétaires.

Pour permettre à Dreamweaver de corriger le code HTML dans plus de types de fichiers :

1. Sélectionnez Edition > Préférences, puis la catégorie Correction du code.
2. Sélectionnez l'une des options suivantes :
   - Corriger les balises incorrectement imbriquées et non fermées
   - Supprimer les balises de fermeture superflues
3. Procédez comme suit, au choix :
   - Supprimez une ou plusieurs des extensions de la liste dans l'option Ne jamais corriger le code : Dans les fichiers avec extensions.
   - Désélectionnez l'option Ne jamais corriger le code : Dans les fichiers avec extensions. La désactivation de cette option permet à Dreamweaver de réécrire le code HTML dans tous types de fichiers.

Pour ajouter des fichiers dans lesquels Dreamweaver ne doit pas apporter de corrections :

1. Sélectionnez Edition > Préférences, puis la catégorie Correction du code.
2. Sélectionnez l'une des options suivantes :
   - Corriger les balises incorrectement imbriquées et non fermées
   - Supprimer les balises de fermeture superflues
3. Assurez-vous que l'option Ne jamais corriger le code : Dans les fichiers avec extensions est sélectionnée et ajoutez les nouvelles extensions de fichiers à la liste dans la zone de texte.

Si le nouveau type de fichier ne s'affiche pas dans le menu contextuel Types de fichiers dans la boîte de dialogue Fichier > Ouvrir, vous pouvez l'ajouter dans le fichier Configuration/Extensions.txt. Pour plus de détails, voir *Modification du type de fichier par défaut*, page 20.

Pour désactiver les options d'encodage de Dreamweaver :

1. Sélectionnez Edition > Préférences, puis la catégorie Correction du code.
2. Désélectionnez les options de caractères spéciaux de votre choix.

Pour plus d'informations sur les autres préférences de correction de code, voir *Utilisation de Dreamweaver*.

Personnalisation de Dreamweaver dans un environnement multi-utilisateurs

Vous pouvez personnaliser Dreamweaver dans un système d'exploitation multi-utilisateurs tel que Windows 2000, Windows XP ou Mac OS X. Aucune configuration personnalisée de Dreamweaver ne peut affecter celle d'un autre utilisateur. Pour ce faire, lors de la première exécution de Dreamweaver sous un système d'exploitation multiutilisateur reconnu, Dreamweaver copie divers fichiers de configuration dans un dossier Configuration utilisateur destiné à votre utilisation personnelle. Lorsque vous personnalisez Dreamweaver à l'aide des panneaux et boîtes de dialogue, l'application modifie vos fichiers de configuration utilisateur au lieu de modifier les fichiers de configuration de Dreamweaver. Pour personnaliser Dreamweaver en modifiant un fichier de configuration dans un environnement multiutilisateur, modifiez le fichier de configuration utilisateur correspondant au lieu de modifier les fichiers du dossier Configuration de Dreamweaver. Pour apporter une modification affectant la plupart des utilisateurs, vous pouvez modifier un fichier Configuration de Dreamweaver, mais les utilisateurs disposant déjà de fichiers de configuration personnels ne seront pas affectés. En général, si vous souhaitez apporter une modification affectant tous les utilisateurs, il est préférable de créer une extension et de l'installer à l'aide de Extension Manager.

REMARQUE

Sur des systèmes d'exploitation plus anciens (Windows 98, Windows ME et Mac OS 9.x), un seul groupe de fichiers de configuration de Dreamweaver est partagé par tous les utilisateurs, même si la configuration du système d'exploitation prend en charge plusieurs utilisateurs.

L'emplacement du dossier Configuration varie selon la plate-forme utilisateur.

Les plate-formes Windows 2000 et Windows XP utilisent l'emplacement suivant :

lecteur:\Documents and Settings*nom_utilisateur*\Application Data\Macromedia\Dreamweaver 8\Configuration

REMARQUE

Sous Windows XP, ce dossier peut se trouver dans un dossier masqué.

Les plate-formes Mac OS X utilisent l'emplacement suivant :

lecteur:Users/*nom_utilisateur*/Library/Application Support/Macromedia/Dreamweaver 8/Configuration

REMARQUE	Pour installer des extensions accessibles à tous les utilisateurs dans un système d'exploitation multiutilisateur, vous devez disposer des droits Administrateur (Windows) ou root (Mac OS X).

Lorsque vous exécutez Dreamweaver pour la première fois, il ne copie que certains fichiers de configuration dans votre dossier de configuration utilisateur. (les fichiers copiés sont spécifiés dans le fichier version.xml dans le dossier Configuration). Lorsque vous personnalisez Dreamweaver depuis l'application même (par exemple, lorsque vous modifiez un des fragments de code prédéfinis dans le panneau Fragments de code), Dreamweaver copie les fichiers concernés dans votre dossier de configuration utilisateur. La version d'un fichier présente dans le dossier de configuration utilisateur prime toujours sur celle du dossier Configuration de Dreamweaver. Pour personnaliser un fichier de configuration qui n'a pas été copié par Dreamweaver dans le dossier de configuration utilisateur, copiez en premier lieu le fichier du dossier Configuration de Dreamweaver vers l'emplacement correspondant dans votre dossier de configuration utilisateur. Modifiez ensuite cette copie dans votre dossier de configuration utilisateur.

Suppression de fichiers de configuration dans un environnement multiutilisateur

Lors de toute opération engendrant l'effacement d'un fichier de configuration dans Dreamweaver sous un système d'exploitation multiutilisateur (par exemple, la suppression d'un fragment de code prédéfini dans le panneau Fragments de code), Dreamweaver crée un fichier nommé mm_deleted_files.xml dans votre dossier Configuration. Lorsqu'un fichier est répertorié dans le fichier mm_deleted_files.xml, Dreamweaver se comporte comme si le fichier n'existait plus.

Pour désactiver un fichier de configuration :

1. Quittez Dreamweaver.

2. A l'aide d'un éditeur de texte, modifiez le fichier mm_deleted_files.xml dans votre dossier de configuration utilisateur, ajoutez une balise d'élément dans ce fichier en indiquant le chemin (relatif au dossier Configuration de Dreamweaver) du fichier de configuration à désactiver.

REMARQUE	Ne modifiez pas le fichier mm_deleted_files.xml dans Dreamweaver.

3. Enregistrez et fermez mm_deleted_files.xml.
4. Lancez à nouveau Dreamweaver.

A propos de la syntaxe des balises mm_deleted_files.xml

Le fichier mm_deleted_files.xml contient une liste structurée d'éléments qui indiquent les fichiers de configuration que Dreamweaver doit ignorer. Ces éléments sont spécifiés par des balises XML, que vous pouvez modifier dans un éditeur de texte.

Les sections suivantes décrivent la syntaxe des balises du fichier mm_deleted_files.xml. Les attributs optionnels sont définis par des accolades (`{}`) dans la liste des attributs ; les attributs ne comportant pas d'accolades sont donc requis.

<deleteditems>

Description

Balise de conteneur renfermant une liste d'éléments à considérer comme supprimés par Dreamweaver.

Attributs

Néant

Contenu

Cette balise doit contenir une ou plusieurs balises `item`.

Contenant

Néant

Exemple

```
<deleteditems>
<!-- placez les balises item ici -->
</deleteditems>
```

<item>

Description

Définit un fichier de configuration à ignorer par Dreamweaver.

Attributs

`name`

- `name` Chemin du fichier de configuration par rapport au dossier Configuration. Sous Windows, utilisez une barre oblique inversée (\) pour séparer les éléments constitutifs du chemin ; sur Macintosh, utilisez deux points (:).

Contenu

Aucun (balise vide).

Contenant

Cette balise doit se trouver dans une balise `deleteditems`.

Exemple

```
<item name="snippets\headers\5columnwith4links.csn" />
```

Réinstallation et désinstallation de Dreamweaver dans un environnement multi-utilisateurs

Après avoir installé Dreamweaver, si vous souhaitez réinstaller le programme ou le mettre à jour, Dreamweaver sauvegarde automatiquement une copie des fichiers de configuration utilisateur existants. Vos paramètres personnels resteront donc accessibles. Lors de la désinstallation de Dreamweaver dans un environnement multiutilisateur (opération possible uniquement pour les utilisateurs disposant de privilèges administratifs), Dreamweaver peut supprimer tous les dossiers de configuration utilisateur à votre demande.

Utilisation des profils de navigateurs

Les profils de navigateurs sont les fichiers utilisés par Dreamweaver pour vérifier vos documents lorsque vous exécutez la vérification d'un navigateur cible (voir *Utilisation de Dreamweaver*). Chaque profil contient des informations à propos des balises et attributs HTML pris en charge par un navigateur donné. Un profil de navigateur peut également renfermer des avertissements, des messages d'erreur ainsi que des suggestions de substitution de balises.

Les profils de navigateurs sont stockés dans le dossier Configuration/BrowserProfiles dans le dossier de l'application Dreamweaver. Vous pouvez modifier les profils existants ou en créer de nouveaux à l'aide de Dreamweaver ou d'un éditeur de texte. Il n'est pas nécessaire de quitter Dreamweaver avant de modifier ou de créer des profils de navigateurs.

A propos de la mise en forme des profils de navigateurs

Les profils de navigateur suivent un format spécifique. Afin d'éviter toute erreur d'analyse durant la vérification des navigateurs cibles, observez les instructions suivantes lors de la modification ou de la création de profils :

- La première ligne est destinée au nom du profil. Elle doit être suivie d'un retour chariot simple. Le nom défini sur cette ligne s'affiche dans la boîte de dialogue Vérification du navigateur cible et dans le rapport de vérification du navigateur cible. Ce nom doit être unique.
- La seconde ligne est réservée à l'indicateur `PROFILE_TYPE=BROWSER_PROFILE`. Cette ligne permet à Dreamweaver de déterminer quels documents sont des profils de navigateurs. Cette ligne ne doit pas être modifiée ou déplacée.
- Si une ligne commence par deux tirets (--), cela indique un commentaire (cette ligne est donc ignorée durant la vérification de la cible). Tout commentaire doit être placé en début de ligne. Il ne faut pas insérer deux tirets en milieu de ligne.
- Vous devez utiliser un espace aux emplacements suivants :
  - Avant le crochet de fermeture (>) dans la ligne `!ELEMENT`
  - Après la parenthèse d'ouverture dans une liste de valeurs d'un attribut.
  - Avant une parenthèse de fermeture dans une liste de valeurs.
  - Avant et après chaque trait vertical (|) dans une liste de valeurs.
- Vous devez entrer un point d'exclamation (!) (non précédé d'un espace) avant chacun des mots suivants :

  `ELEMENT`, `ATTLIST`, `Error` et `msg` (`!ELEMENT`, `!ATTLIST`, `!Error`, `!msg`).
- Vous pouvez inclure `!Error`, `!Warning` et `!Info` dans la zone `!ELEMENT` ou `!ATTLIST`.
- Les messages `!msg` peuvent uniquement contenir du texte simple.
- Les commentaires HTML (`!---->`) ne peuvent être répertoriés comme balises dans les profils de navigateurs car ils interfèrent dans l'analyse. Dreamweaver ne consigne aucune erreur à propos des commentaires car ils sont pris en charge par tous les navigateurs.

L'exemple suivant indique la syntaxe d'une entrée de balise :

```
<!ELEMENT htmlTag NAME="tagName ">
```

```
<!ATTLIST htmlTag
unsupportedAttribute1 !Error !msg="The unsupportedAttribute1
attribute of the htmlTag tag is not supported.Try using
supportedAttribute1 for a similar effect."
supportedAttribute1
supportedAttribute2 (validValue1 |validValue2 |validValue3 )
unsupportedAttribute2 !Error !msg="Don't ever use the
unsupportedAttribute2 attribute of the htmlTag tag!"
>
```

Les éléments utilisés dans cette syntaxe se définissent comme suit :

- `BaliseHTML` est la balise telle qu'elle apparaît dans le document HTML.
- `NomDeBalise` est un nom explicatif de la balise. Par exemple, le nom de la balise `HR` est "Horizontal Rule" (barre horizontale). L'attribut `NAME` est facultatif. Si cet élément est défini, *`NomDeBalise`* est utilisé pour les messages d'erreur. Si aucun nom n'est défini, *`BaliseHTML`* est utilisé dans les messages d'erreur.
- `AttributNonSupporte` est un attribut non pris en charge. Tout attribut ou balise n'étant pas explicitement mentionné comme pris en charge sera considéré comme non pris en charge. Spécifiez les balises ou attributs non pris en charge uniquement lorsque vous souhaitez créer un message d'erreur personnalisé.
- `AttributSupporte` est un attribut pris en charge par *`htmlTag`*. Seules les balises répertoriées sans désignation `!Error` sont considérées comme prises en charge par le navigateur.
- `ValeurValide` indique une valeur prise en charge par l'attribut.

L'exemple suivant indique une entrée de la balise `APPLET` exacte pour Netscape Navigator 3.0 :

```
<!ELEMENT APPLET Name="Java Applet">
<!ATTLIST APPLET
Align (top |middle |bottom |left |right |absmiddle |
absbottom |baseline |texttop )
Alt
Archive
Class !Warning !msg="This browser ignores the CLASS attribute for the APPLET
tag."
Code
Codebase
Height
HSpace
ID !Warning !msg="This browser ignores the ID attribute for the APPLET tag.
Use NAME instead."
Name
Style !Warning !msg="This browser ignores the STYLE attribute for the APPLET
tag."
VSpace
```

```
Width
>
```

Création et modification d'un profil de navigateur

Vous pouvez créer un profil de navigateur en modifiant un profil existant. Par exemple, pour créer un profil pour une version future de Microsoft Internet Explorer disposant déjà d'un profil, ajoutez les nouvelles balises ou attributs introduits dans la nouvelle version et enregistrez-le comme profil pour la nouvelle version.

REMARQUE	Avant de créer un profil de navigateur pour une nouvelle version d'un navigateur, rendez-vous sur le site de Macromedia Exchange pour Dreamweaver à l'adresse suivante : www.macromedia.com/go/dreamweaver_exchange_fr/. Vous pourrez ainsi vérifier si Macromedia a fourni un profil de navigateur à télécharger et à installer à l'aide de Extension Manager.

Pour créer ou modifier un profil de navigateur :

1. Ouvrez le fichier existant à modifier.

   Pour créer un nouveau profil, ouvrez un profil le plus identique possible à celui que vous souhaitez créer, puis enregistrez-le sous un nouveau nom.
2. Si vous créez un profil, changez le nom qui s'affiche sur la première ligne de texte dans le fichier (deux profils ne peuvent pas porter le même nom).
3. Ajoutez les nouvelles balises ou attributs pris en charge par le navigateur en utilisant la syntaxe présentée dans *A propos de la mise en forme des profils de navigateurs*, page 33.

   Si vous ne souhaitez pas recevoir de messages d'erreur à propos d'une balise donnée non prise en charge, ajoutez cette dernière à la liste des balises prises en charge. Si vous effectuez cette opération, enregistrez le profil dans un fichier séparé sous un nouveau nom de fichier (par exemple, NomduNavigateur x.x limited). Le fait de donner un nouveau nom à cet autre profil conserve le profil original avec uniquement les balises réellement prises en charge.
4. Supprimez tous les attributs et balises qui ne sont pas pris en charge par le navigateur.

   Cette étape est normalement facultative si vous créez un profil pour une nouvelle version de Netscape Navigator ou de Internet Explorer car les navigateurs n'abandonnent que rarement la prise en charge d'une balise.
5. Ajoutez des messages d'erreur en respectant la syntaxe présentée dans *A propos de la mise en forme des profils de navigateurs*, page 33.

Les profils fournis avec Dreamweaver répertorient toutes les balises prises en charge pour les navigateurs spécifiés. Pour ajouter un message d'erreur personnalisé à une balise, entrez `!msg = "message"` après `!Error`. L'exemple suivant présente des informations qui s'affichent dans le profil de Netscape Navigator 3.0 (avec d'autres attributs non présentés ici) :

```
<!ELEMENT HR name="Horizontal Rule">
<!ATTLIST HR
COLOR !Error
>
```

Pour ajouter un message d'erreur personnalisé, entrez **!msg=** suivi de votre message d'erreur entre guillemets (") :

```
<!ELEMENT HR name="Horizontal Rule">
<!ATTLIST HR
COLOR !Error !msg="Internet Explorer 3.0 supports the COLOR tag in
horizontal rules,but Netscape Navigator 3.0 does not."
>
```

6. Vous pouvez utiliser l'élément `!Error` pour toutes les situations d'erreur ou spécifier l'élément `!Warning` ou `!Info` pour indiquer qu'une balise sera ignorée sans causer d'erreur.

Modification des mappages FTP

Le fichier FTPExtensionMap.txt (Windows) et le fichier FTPExtensionMapMac.txt (Macintosh) mappent les extensions de nom de fichier en modes de transfert FTP (ASCII ou BINAIRE).

Chaque ligne dans chacun des deux fichiers comprend une extension de nom de fichier (par exemple, GIF) ainsi que le mot ASCII ou le mot BINARY (binaire) pour indiquer quel mode utiliser lors du transfert d'un fichier comportant cette extension. Sur Macintosh, chaque ligne comprend également un code créateur (tel que DmWr) et un type de fichier (tel que TEXT). Lorsque vous téléchargez un fichier portant l'extension donnée, Dreamweaver lui assigne alors le créateur et le type de fichier désignés.

Si un fichier que vous transférez ne comporte pas d'extension de nom de fichier, Dreamweaver utilise le mode de transfert binaire.

REMARQUE	Dreamweaver ne peut pas transférer de fichiers en mode Macbinary. Si vous devez transférer des fichiers en mode Macbinary, vous devez utiliser un autre client FTP.

Les exemples suivants illustrent une ligne (du fichier Macintosh) indiquant que les fichiers portant l'extension .html doivent être transférés en mode ASCII :

```
HTML DmWr TEXT ASCII
```

Dans le fichier FTPExtensionMap.txt et le fichier FTPExtensionMapMac.txt (Macintosh), tous les éléments placés sur une même ligne sont séparés par des tabulations. L'extension et le mode de transfert sont écrits en majuscules.

Pour modifier un paramètre par défaut, modifiez le fichier dans un éditeur de texte.

Pour ajouter des informations relatives à une nouvelle extension de nom de fichier :

1. Modifiez le fichier extension-map dans un éditeur de texte.
2. Sur une ligne vierge, entrez l'extension de nom de fichier (en majuscules), puis appuyez sur la touche de tabulation.
3. Sur Macintosh, ajoutez le code créateur, un caractère de tabulation, puis le type de fichier suivi d'un autre caractère de tabulation.
4. Choisissez entre ASCII et BINARY (binaire) pour spécifier un mode de transfert FTP.
5. Enregistrez le fichier.

Types de documents extensibles dans Dreamweaver

XML est doté d'un système performant pour définir des documents et des structures de données complexes. Dreamweaver organise selon plusieurs schémas XML les informations sur les comportements de serveur, les balises et les boîtes de dialogue de balises, les composants, les types de documents et des références.

Lorsque vous créez et utilisez des extensions dans Dreamweaver, vous pouvez souvent créer ou modifier les fichiers XML existants afin de gérer les données utilisées par ces extensions. Dans la plupart des cas, vous pouvez copier un fichier existant du sous-dossier approprié du dossier Configuration vers le dossier à utiliser comme modèle.

Fichier de définition de type de document

Le concept de type de document s'articule autour d'un composant central, à savoir le fichier de définition de type de document. Vous pouvez être en présence de plusieurs fichiers de définition ; le cas échéant, ils résident tous dans le dossier Configuration/DocumentTypes. Chaque fichier de définition contient des informations concernant au moins un type de document. Des informations essentielles, telles que le modèle de serveur, le style de codage par couleurs, les descriptions, etc., sont décrits pour chacun de ces types de documents.

REMARQUE

Attention, il ne faut pas confondre fichiers de définition de type de document de Dreamweaver et définition de type de document XML (DTD). Les fichiers de définition de type de document de Dreamweaver contiennent un ensemble d'éléments `documenttype`, chacun d'entre eux définissant une collection prédéfinie de balises et d'attributs associés à un type de document. Au lancement, Dreamweaver analyse les fichiers de définition de type de document et place en mémoire une base de données d'informations concernant tous les types de documents définis.

Dreamweaver fournit un fichier de définition de type de document initial Ce fichier, nommé MMDocumentTypes.xml, contient les définitions de type de document fournies par Macromedia :

Type de document	Modèle de serveur	Type interne	Extensions de fichier	Modèle de serveur antérieur
ASP.NET C#	ASP.NET-Csharp	Dynamic	aspx, ascx	
ASP.NET VB	ASP.NET-VB	Dynamic	aspx, ascx	
ASP JavaScript	ASP-JS	Dynamic	asp	
ASP VBScript	ASP-VB	Dynamic	asp	
ColdFusion	ColdFusion	Dynamic	cfm, cfml	UltraDev 4 ColdFusion
Composant ColdFusion		Dynamic	cfc	
JSP	JSP	Dynamic	jsp	
PHP	PHP	Dynamic	php, php3	
Elément de bibliothèque		Extension DW	lbi	
Modèle ASP.NET C#		Modèle DW	axcs.dwt	

Type de document	Modèle de serveur	Type interne	Extensions de fichier	Modèle de serveur antérieur
Modèle ASP.NET VB		Modèle DW	axvb.dwt	
Modèle ASP JavaScript		Modèle DW	aspjs.dwt	
Modèle ASP VBScript		Modèle DW	aspvb.dwt	
Modèle ColdFusion		Modèle DW	cfm.dwt	
Modèle HTM		Modèle DW	dwt	
Modèle JSP		Modèle DW	jsp.dwt	
Modèle PHP		Modèle DW	php.dwt	
HTML		HTML	htm, html	
ActionScript		Text	as	
CSharp		Text	cs	
CSS		Text	css	
Java		Text	java	
JavaScript		Text	js	
VB		Text	vb	
VBScript		Text	vbs	
Text		Text	txt	
EDM		XML	edml	
TLD		XML	tld	
VTML		XML	vtm, vtml	
WML		XML	wml	
XML		XML	xml	

Si vous avez besoin de créer un nouveau type de document, vous pouvez soit ajouter votre entrée dans le fichier de définition de document fourni par Macromedia (MMDocumentTypes.xml), soit ajouter votre propre fichier de définition dans le dossier Configuration/DocumentTypes.

REMARQUE

Le sous-dossier NewDocuments résidant dans le dossier Configuration/DocumentTypes contient des pages par défaut (modèles) propres à chaque type de document.

Structure des fichiers de définition de type de document

L'exemple suivant représente un fichier classique de définition de type de document :

```
<?xml version="1.0" encoding="utf-8"?>
<documenttypes
      xmlns:MMString="http://www.macromedia.com/schemes/data/string/">
  <documenttype
    id="dt-ASP-JS"
    servermodel="ASP-JS"
    internaltype="Dynamic"
    winfileextension="asp,htm, html"
    macfileextension=asp, html"
    previewfile="default_aspjs_preview.htm"
    file="default_aspjs.htm"
    priorversionservermodel="UD4-ASP-JS" >
    <title>
      <loadString id="mmdocumenttypes_0title" />
    </title>
    <description>
      <loadString id="mmdocumenttypes_0descr" />
    </description>
  </documenttype>
  ...
</documenttypes>
```

REMARQUE

Le codage par couleurs des types de documents est défini dans les fichiers XML qui résident dans le dossier Configuration/CodeColoring.

Dans l'exemple précédent, l'élément `loadstring` identifie les chaînes localisées que Dreamweaver devrait utiliser pour le titre et la description des documents de type ASP-JS. Pour plus d'informations sur les chaînes localisées, voir *Chaînes localisées*, page 48.

Le tableau ci-dessous recense les balises et les attributs autorisés dans un fichier de définition de type de document.

Type d'élément		Obligatoire	Description
Balise	**Attribut**		
`documenttype` *(root)*		Oui	Nœud parent.
	`id`	Oui	Identificateur unique pour tous les fichiers de définition de type de document.

	`servermodel`	Non	Spécifie le modèle de serveur associé (casse prise en compte). Les valeurs suivantes sont valides par défaut :
			`ASP.NET C#`
			`ASP.NET VB`
			`ASP VBScript`
			`ASP JavaScript`
			`ColdFusion`
			`JSP`
			`PHP MySQL`
			Un appel des fonctions `getServerModelDisplayName()` renvoie ces noms. Les fichiers d'implémentation des modèles de serveur se trouvent dans le dossier Configuration/ServerModels. Les nouveaux modèles de serveur créés par les développeurs d'extensions viendront compléter cette liste.

Type d'élément		Obligatoire	Description
Balise	**Attribut**		

	`internaltype`	Oui	Classification générale des modes de traitement des fichiers dans Dreamweaver. Le `type interne` détermine si le mode Création est activé pour ce document et traite des cas particuliers tels que les extensions ou les modèles Dreamweaver.
			Les valeurs suivantes sont valides :
			`Dynamic`
			`DWExtension` (zones d'affichage spéciales)
			`DWTemplate` (zones d'affichage spéciales)
			`HTML`
			`HTML4`
			`Text` (mode Code uniquement)
			`XHTML1`
			`XML` (mode Code uniquement)
			Tous les types de documents liés au serveur de modèle doivent être associés à la valeur `Dynamic`. HTML doit être en correspondance avec `HTML`. Les fichiers de script, notamment les fichiers .css, .js, .vb et .cs doivent être en correspondance avec `Text`.
			Si `internaltype` correspond à `DWTemplate`, il convient également de spécifier `dynamicid`. Si, dans le cas présent, vous omettez de spécifier `dynamicid`, le type de document du nouveau modèle vierge créé à partir de la boîte de dialogue Nouveau document ne sera pas reconnu par les panneaux Comportements de serveur ou Liaisons. Les instances de ce modèle sont des modèles HTML.

Type d'élément		Obligatoire	Description
Balise	**Attribut**		
	`dynamicid`	Non	Référence à l'identificateur unique d'un type de document dynamique. Cet attribut n'est pertinent que lorsque `internaltype` correspond à `DWTemplate`. Il vous permet d'associer un modèle dynamique avec un type de document dynamique.
	`winfileextension`	Oui	Extension de fichier associée au type de document sous Windows. Utilisez une liste séparée par des virgules pour spécifier plusieurs extensions. La première extension de cette liste correspond à l'extension utilisée par Dreamweaver lorsque l'utilisateur enregistre un document de type `documenttype`. Lorsque deux types de documents non associés à un modèle de serveur portent la même extension de fichier, Dreamweaver reconnaît le premier comme le type de document associé à l'extension.
	`macfileextension`	Oui	Extension de fichier associée au type de document sur Macintosh. Utilisez une liste séparée par des virgules pour spécifier plusieurs extensions. La première extension de cette liste correspond à l'extension utilisée par Dreamweaver lorsque l'utilisateur enregistre un document de type `documenttype`. Lorsque deux types de documents non associés à un modèle de serveur portent la même extension de fichier, Dreamweaver reconnaît le premier comme le type de document associé à l'extension.

Type d'élément		Obligatoire	Description
Balise	**Attribut**		
	`previewfile`	Non	Fichier rendu dans la zone Aperçu de la boîte de dialogue Nouveau document.
	`file`	Oui	Le fichier situé dans le dossier DocumentTypes/NewDocuments contenant le contenu du modèle des nouveaux documents de type `documenttype`.
	`priorversionservermodel`	Non	S'il existe un équivalent du modèle de serveur de ce document dans Dreamweaver UltraDev 4, spécifiez le nom de la version antérieure du modèle de serveur. UltraDev 4 ColdFusion est un modèle de serveur antérieur valide.
`title` (sous-balise)		Oui	Chaîne qui apparaît comme élément de catégorie sous la section Document vierge de la boîte de dialogue Nouveau document. Vous pouvez insérer cette chaîne directement dans le fichier de définition ou pointer vers elle indirectement pour la localiser. Pour plus d'informations sur la localisation de cette chaîne, voir *Chaînes localisées*, page 48. La mise en forme n'étant pas autorisée, il est impossible de spécifier les balises HTML.
`description` (sous-balise)		Non	Chaîne de description du type de document. Vous pouvez insérer cette chaîne directement dans le fichier de définition ou pointer vers elle indirectement pour la localiser. Pour plus d'informations sur la localisation de cette chaîne, voir *Chaînes localisées*, page 48. La mise en forme étant autorisée, il est possible de spécifier les balises HTML.

REMARQUE

Lorsque l'utilisateur enregistre un nouveau document, Dreamweaver examine la liste des extensions de la plate-forme actuelle qui sont associées avec le type de document (`winfileextension` et `macfileextension`). Dreamweaver sélectionne la première chaîne de la liste pour l'utiliser comme l'extension de fichier par défaut. Pour changer cette extension de fichier par défaut, les extensions de la liste séparées par des virgules doivent être réorganisées de telle sorte que la nouvelle extension par défaut soit la première entrée de la liste.

Au lancement, Dreamweaver lit tous les fichiers de définition de type de document et crée une liste des types de documents valides. Dreamweaver traite toutes les entrées des fichiers de définition pour lesquels des modèles de serveur inexistants sont utilisés comme types de documents non serveur de modèle. Dreamweaver ignore les entrées dont le contenu est incorrect ou dont les ID ne sont pas uniques.

Si, pendant l'analyse du dossier Configuration/DocumentTypes, Dreamweaver ne détecte aucun fichier de définition de type de document ou si l'un des fichiers de définition semble endommagé, Dreamweaver s'arrête en affichant un message d'erreur.

Modèles dynamiques

Vous pouvez créer des modèles d'après des types de documents dynamiques, appelés des *modèles dynamiques*. La notion de modèle dynamique repose sur les deux éléments fondamentaux suivants :

- La valeur de l'attribut `internaltype` d'un nouveau type de document doit être `DWTemplate`.
- L'attribut `dynamicid` doit être défini et sa valeur doit faire référence à l'identificateur d'un type de document existant.

L'exemple suivant définit un type de document dynamique :

```
<documenttype
    id="PHP_MySQL"
    servermodel="PHP MySQL"
    internaltype="Dynamic"
    winfileextension="php,php3"
    macfileextension="php,php3"
    file="Default.php">
    <title>PHP</title>
    <description><![CDATA[PHP document]]></description>
</documenttype>
```

Vous pouvez désormais définir le modèle dynamique suivant, basé sur ce type de document PHP_MySQL :

```
<documenttype
    id="DWTemplate_PHP"
    internaltype="DWTemplate"
    dynamicid="PHP_MySQL"
    winfileextension="php.dwt"
    macfileextension="php.dwt"
    file="Default.php.dwt">
    <title>PHP Template</title>
    <description><![CDATA[Dreamweaver PHP Template document]]></
  description>
</documenttype>
```

Lorsqu'un utilisateur de Dreamweaver crée un nouveau modèle vierge de type DWTemplate_PHP, Dreamweaver lui permet de créer des comportements de serveur PHP dans le fichier. En outre, lorsque cet utilisateur crée des instances du nouveau modèle, il peut leur définir aussi des comportements de serveur PHP.

Dans l'exemple précédent, où l'utilisateur enregistre le modèle, Dreamweaver ajoute automatiquement l'extension .php.dwt au fichier. De même, lorsque cet utilisateur enregistre une instance du modèle, le fichier reçoit l'extension .php.

Extensions de documents et types de fichiers

Par défaut, Dreamweaver affiche tous les types de fichiers reconnus dans la boîte de dialogue Fichier > Ouvrir. Une fois le nouveau type de document créé, il incombe aux développeurs d'extensions de mettre à jour le fichier Extensions.txt approprié. Si l'utilisateur travaille sur un système multi-utilisateurs (par exemple, Windows XP, Windows 2000 ou Mac OS X), le dossier Configuration contient un autre fichier Extensions.txt. L'utilisateur doit mettre à jour le fichier Extensions.txt car il s'agit de l'instance recherchée et analysée par Dreamweaver.

L'emplacement du dossier Configuration varie selon la plate-forme utilisateur.

Les plate-formes Windows 2000 et Windows XP utilisent l'emplacement suivant :

lecteur:\Documents and Settings*nom_utilisateur*\Application Data\Macromedia\Dreamweaver 8\Configuration

REMARQUE	Sous Windows XP, ce dossier peut se trouver dans un dossier masqué.

Les plate-formes Mac OS X utilisent l'emplacement suivant :

lecteur: Disquedur/Utilisateurs/*nom_utilisateur*/Bibliothèque/Macromedia/Dreamweaver 8/Configuration

Si Dreamweaver ne parvient pas à localiser le fichier Extensions.txt dans le dossier Configuration de l'utilisateur, Dreamweaver le recherche dans le dossier Dreamweaver Configuration.

REMARQUE	Sur les plates-formes multiutilisateurs, toute modification de la copie du fichier Extensions.txt qui réside dans le dossier Dreamweaver Configuration, et non pas celle du dossier Configuration de l'utilisateur, est transparente pour Dreamweaver, lequel analyse la copie du fichier Extensions.txt située dans le dossier Configuration de l'utilisateur, et non pas celle du dossier Dreamweaver Configuration.

Pour cela, vous avez le choix entre ajouter la nouvelle extension à un type de document existant ou créer un nouveau type de document.

Pour ajouter une nouvelle extension à un type de document existant :

1. Modifiez le fichier MMDocumentTypes.xml.
2. Ajoutez la nouvelle extension aux attributs `winfileextension` et `macfileextension` du type de document existant.

Pour ajouter un nouveau type de document :

1. Créez une copie de sauvegarde du fichier Extensions.txt dans le dossier Configuration.
2. Ouvrez le fichier Extensions.txt dans Dreamweaver ou un éditeur de texte.
3. Ajoutez une nouvelle ligne pour chaque nouveau type de fichier. Entrez, en majuscules, les extensions de noms de fichiers compatibles avec le nouveau type de fichier et séparez-les par des virgules. Ajoutez ensuite deux points et une brève description à afficher dans le menu contextuel pour les types de fichiers qui s'affichent dans la boîte de dialogue Fichier > Ouvrir.

   Par exemple, pour les fichiers JPEG, entrez `JPG,JPEG,JFIF:Fichiers image JPEG`
4. Enregistrez le fichier Extensions.txt.
5. Redémarrez Dreamweaver.

   Pour observer les modifications, sélectionnez Fichier > Ouvrir et cliquez sur le menu contextuel répertoriant les types de fichiers.

Pour modifier le type de fichier indiqué par défaut dans Fichier › Ouvrir de Dreamweaver :

1. Créez une copie de sauvegarde du fichier Extensions.txt dans le dossier Configuration.
2. Ouvrez le fichier Extensions.txt dans Dreamweaver ou un éditeur de texte.

3. Coupez la ligne correspondant au nouveau type par défaut et collez-la en tête de fichier, sur la première ligne.
4. Enregistrez le fichier Extensions.txt.
5. Redémarrez Dreamweaver.

   Pour observer les modifications, sélectionnez Fichier > Ouvrir et cliquez sur le menu contextuel répertoriant les types de fichiers.

Chaînes localisées

Dans un fichier de définition de type de document, les sous-balises `<title>` et `<description>` désignent le titre d'affichage et la description du type de document. Vous pouvez utiliser la directive `MMString:loadstring` dans les sous-balises comme espace réservé pour les chaînes localisées de ces deux sous-balises. Ce procédé, similaire à la programmation de scripts côté serveur, permet de spécifier l'utilisation d'une chaîne particulière dans votre page en utilisant un identificateur de chaîne comme espace réservé. Cet espace réservé accepte les balises spéciales ou les attributs de balises dont la valeur est remplacée.

Pour définir des chaînes localisées, procédez comme suit :

1. Placez l'instruction suivante en tête du fichier de définition de type de document :

   ```
   <?xml version="1.0" encoding="utf-8"?>
   ```

2. Déclarez l'espace de noms `MMString` dans la balise `<documenttypes>` :

   ```
   <documenttypes
       xmlns:MMString="http://www.macromedia.com/schemes/data/string/">
   ```

3. A l'emplacement du fichier de définition de type de document auquel insérer une chaîne localisée, utilisez la directive `MMString:loadstring` pour définir un espace réservé à cette chaîne. Vous pouvez spécifier cet espace réservé en procédant comme suit, au choix :

   ```
   <description>
      <loadstring>myJSPDocType/Description</loadstring>
   </description>
   ```

   ou

   ```
   <description>
      <loadstring id="myJSPDocType/Description" />
   </description>
   ```

   Dans ces exemples, `myJSPDocType/Description` est un identificateur de chaîne unique faisant office d'espace réservé à la chaîne localisée. La chaîne localisée est définie à l'étape suivante.

4. Dans le dossier Configuration/Strings, créez un fichier XML (ou modifiez un fichier existant) définissant la chaîne localisée. Par exemple, lorsque le code suivant est inséré dans le fichier Configuration/Strings/strings.xml, il définit la chaîne `myJSPDocType/Description` :

```
<strings>
...
   <string id="myJSPDocType/Description"
        value=
        "<![CDATA[JavaServer Page with <em>special</em>
  features]]>"
   />
...
</strings>
```

REMARQUE

Les identificateurs de chaîne, tels que `myJSPDocType/Description` dans l'exemple précédent, doivent être uniques dans l'application Dreamweaver. Au lancement, Dreamweaver analyse tous les fichiers XML du dossier Configuration/Strings et charge ces chaînes uniques.

Règles d'utilisation des fichiers de définition de type de document

Dreamweaver autorise les types de documents associés à un modèle de serveur à partager des extensions de fichiers. Exemple : ASP-JS et ASP-VB peuvent adopter l'extension .asp (pour savoir quel modèle de serveur prévaut, voir *canRecognizeDocument()*, page 444).

Dreamweaver n'autorise pas les types de documents non associés à un modèle de serveur à partager des extensions de fichiers.

Si une extension de fichier est revendiquée par deux types de documents alors qu'un type est associé à un modèle de serveur et que l'autre ne l'est pas, ce dernier prévaut. Supposons que vous ayez défini un type de document appelé SAM, non associé à un modèle de serveur et portant l'extension de fichier .sam, et que vous ajoutiez cette extension au type de document ASP-JS. Lorsqu'un utilisateur ouvre un fichier portant l'extension .sam dans Dreamweaver, le programme lui affecte le type de document SAM, et non pas le type ASP-JS.

Ouverture d'un document dans Dreamweaver

Lorsqu'un utilisateur ouvre un document, Dreamweaver identifie en plusieurs étapes le type de document d'après son extension de fichier.

Si Dreamweaver détecte un type de document unique, il l'utilise et charge le modèle de serveur associé (le cas échéant) pour le document que l'utilisateur ouvre. Si cet utilisateur a opté pour l'utilisation des comportements de serveur de Dreamweaver UltraDev 4, Dreamweaver charge le modèle de serveur UltraDev 4 correspondant.

Si l'extension de fichier est commune à plusieurs types de documents, Dreamweaver effectue les opérations suivantes :

- Si un type de document statique figure dans la liste des types de documents, il prévaut.
- Si tous les types de documents sont dynamiques, Dreamweaver génère une liste alphabétique des modèles de serveur associés à ces types de documents, puis appelle la fonction `canRecognizeDocument()` pour chacun des modèles de serveur (voir *canRecognizeDocument()*, page 444). Dreamweaver collecte les valeurs de retour et identifie le modèle de serveur qui a renvoyé l'entier positif le plus grand. Le type de document dont le modèle de serveur a renvoyé l'entier le plus grand correspond au type de document assigné par Dreamweaver au document en cours d'ouverture. Si, toutefois, plusieurs modèles de serveur renvoient le même entier, Dreamweaver passe en revue la liste alphabétique de ces modèles de serveur, choisit le premier de la liste, puis l'utilise. Si, par exemple, les types ASP-JS et ASP-VB portent la même extension .asp et que leur fonction `canRecognizeDocument()` respective renvoie une valeur égale, Dreamweaver affecte au document le type ASP-JS (puisque ASP-JS apparaît en premier dans l'ordre alphabétique).

Si Dreamweaver ne peut pas faire correspondre l'extension de fichier à un type de document, Dreamweaver ouvre le document au format texte.

Personnalisation des présentations de l'espace de travail

Dreamweaver vous permet de personnaliser la présentation de l'espace de travail. Vous pouvez ainsi spécifier les panneaux inclus dans la présentation indiquée, ainsi que divers autres attributs tels que la position et la taille des panneaux, leur état (réduit ou développé), la position et la taille de la fenêtre d'application, ainsi que la position et la taille de la fenêtre de document.

La présentation de l'espace de travail est spécifiée dans les fichiers XML stockés dans le dossier Configuration/Workspace layouts. Les sections suivantes décrivent la syntaxe des balises XML. Les attributs facultatifs sont désignés par des accolades (`{}`). Tout attribut qui n'est pas entouré d'accolades est donc obligatoire.

<panelset>

Description

Balise d'extrémité, qui signale le début de la description du jeu de panneaux.

Attributs

Néant

Contenu

Cette balise peut contenir une ou plusieurs balises <application>, <document> ou <panelset>.

Contenant

Néant

Exemple

```
<panelset>
<!-- placez les balises panelset ici -->
</panelset>
```

<application>

Description

Stipule la position et la taille initiales de la fenêtre d'application.

Attributs

`rect, maximize`

- `rect` spécifie la position et la taille de la fenêtre d'application. La chaîne est exprimée au format « gauche haut droite bas », spécifié sous forme de nombres entiers.
- `maximize` est une valeur booléenne : `true` si la fenêtre d'application est agrandie au démarrage, `false` dans le cas contraire. La valeur par défaut est `true`.

Contenu

Néant

Contenant

Cette balise doit être imbriquée dans une balise `panelset`.

Exemple

```
<panelset>
  <application rect="0 0 1000 1200" maximize="false">
  </application>
</panelset>
```

<document>

Description

Stipule la position et la taille initiales de la fenêtre de document.

Attributs

`rect, maximize`

- `rect` spécifie la position et la taille de la fenêtre de document. La chaîne est exprimée au format « gauche haut droite bas », spécifié sous forme de nombres entiers. Si la valeur `maximize` est réglée sur `true`, la valeur `rect` n'est pas prise en compte.
- `maximize` est une valeur booléenne : `true` si la fenêtre de document est agrandie au démarrage, `false` dans le cas contraire. La valeur par défaut est `true`.

Contenu

Néant

Contenant

Cette balise doit être imbriquée dans une balise `panelset`.

Exemple

```
<panelset>
  <document rect="100 257 1043 1200" maximize="false">
  </document>
</panelset>
```

<panelframe>

Description

Décrit un groupe entier de panneaux.

Attributs

`x, y, {width, height}, dock, collapse`

- `x` indique la position de gauche du groupe de panneaux. Vous pouvez le régler sur un nombre entier ou sur une valeur relative à l'écran. Si le nombre entier excède les limites de l'écran, le groupe de panneaux s'affiche à l'emplacement le plus proche possible pour être visible à l'écran. Les valeurs relatives gérées sont « left (gauche) » ou « right (droite) » et identifient la bordure du groupe de panneaux alignée sur la bordure de l'écran virtuel.

- `y` indique la position supérieure du groupe de panneaux. Vous pouvez le régler sur un nombre entier ou sur une valeur relative à l'écran. Si le nombre entier excède les limites de l'écran, le groupe de panneaux s'affiche à l'emplacement le plus proche possible pour être visible à l'écran. Les valeurs relatives gérées sont « top (haut) » ou « bottom (bas) » et identifient la bordure du groupe de panneaux alignée sur la bordure de l'écran virtuel.
- `width` indique la largeur du groupe de panneaux, exprimée en pixels. Cet attribut est facultatif. Si vous ne spécifiez pas de largeur, la valeur par défaut du groupe de panneaux est utilisée.
- `height` indique la hauteur du groupe de panneaux, exprimée en pixels. Cet attribut est facultatif. Si vous ne spécifiez pas de hauteur, la valeur par défaut du groupe de panneaux est utilisée.
- `dock` est une chaîne qui identifie la bordure du cadre d'application sur laquelle est ancré le groupe de panneaux. Cet attribut n'est pas pris en compte sur Macintosh, car il est impossible d'ancrer les groupes de panneaux.
- `collapse` est une valeur booléenne : `true` indique que le groupe de panneaux est réduit, `false` indique que le groupe de panneaux est développé. Cet attribut n'est pas pris en compte sur Macintosh, car les panneaux flottent.

Contenu

Cette balise doit contenir une ou plusieurs balises `panelcontainer`.

Contenant

Cette balise doit être imbriquée dans une balise `panelset`.

Exemple

```
<panelset>
  <panelframe rect=”196 453 661 987” visible=”true” dock=”floating”>
    <!-- placez les balises panelcontainer ici -->
  </panelframe>
</panelset>
```

<panelcontainer>

Description

Décrit un groupe entier de panneaux.

Attributs

```
expanded, title,{ height,} activepanel, visible, maximize, maxRestorePanel,
  maxRestoreIndex, maxRect, tabsinheader
```

- `expanded` est une valeur booléenne : `true` si le panneau est développé, `false` s'il ne l'est pas.
- `title` est une chaîne qui indique le titre du panneau.
- `height` est un nombre entier qui indique la hauteur des panneaux, exprimée en pixels. Cet attribut est facultatif. Si vous ne spécifiez pas `height`, la valeur par défaut de chaque panneau est utilisée.

REMARQUE	La largeur est héritée du parent.

- `activepanel` est une valeur numérique correspondant à l'ID du panneau avant.
- `visible` est une valeur booléenne : `true` si le panneau est visible, `false` s'il ne l'est pas.
- `maximize` est une valeur booléenne : `true` si le panneau doit être agrandi au lancement, `false` dans le cas contraire.
- `maxRestorePanel` est une valeur numérique correspondant à l'ID du panneau à afficher suite à une restauration.
- `maxRect` est une chaîne qui indique la position et la taille du panneau agrandi. La chaîne est exprimée au format « gauche haut droite bas », spécifié sous forme de nombres entiers.
- `tabsinheader` est une valeur booléenne : `true` indique que les onglets doivent être placés dans le titre plutôt que sous la barre de titre, `false` stipule l'inverse.

Contenu

Cette balise doit contenir une ou plusieurs balises `panel`.

Contenant

Cette balise doit être imbriquée dans une balise `panelframe`.

Exemple

```
<panelset>
  <panelframe rect="196 453 661 987" visible="true" dock="floating">
    <panelcontainer title="Color" height="250" visible="true"
  expanded="true" activepanel="20">
      <!-- placez les balises panel ici --->
    </panelcontainer>
  </panelframe>
</panelset>
```

‹panel›

Description

Spécifie le panneau qui s'affiche dans le contenant.

Attributs

`id, visibleTab`

- `id` est une valeur numérique correspondant à l'ID du panneau. Le tableau ci-dessous contient une liste de valeurs.

Produit	ID	Panneau
Flash	1	Propriétés
	2	Actions
	3	Aligner
	4	Comportements
	5	Composants
	6	Inspecteur de composants
	7	Mélangeur
	8	Nuanciers
	9	Historique
	10	Info
	11	Bibliothèque
	12	Explorateur d'animations
	13	Sortie
	14	Propriétés
	15	Projet
	16	Transformer
	17	Séquence
	18	Chaînes
	19	Débogueur
	101-110	Panneaux Bibliothèque
Dreamweaver	1	Propriétés
Zorn	1	Propriétés

- `visibleTab` est une valeur booléenne : `true` pour afficher l'onglet et le panneau et `false` dans le cas contraire.

Contenu

Néant

Contenant

Cette balise doit être imbriquée dans une balise `panelcontainer`.

Exemple

```
<panelset>
  <panelframe rect="196 453 661 987" visible="true" dock="floating">
    <panelcontainer title="Color" height="250" visible="true"
  expanded="true" activepanel="20">
      <panel id="20"></panel>
    </panelcontainer>
  </panelframe>
</panelset>
```

Personnalisation de la barre d'outils Mode Code

La barre d'outils Mode Code contient initialement 15 boutons. Vous disposez cependant d'autres boutons. Pour personnaliser la barre d'outils Mode Code, vous pouvez modifier les boutons intégrés à celle-ci et leur ordre d'affichage. Vous devez pour ce faire modifier le fichier Configuration/Toolbars/Toolbars.xml. Vous pouvez également insérer vos propres boutons sur la barre d'outils par le biais d'Extension Manager.

Pour modifier l'ordre des boutons :

1. Ouvrez le fichier Configuration/Toolbars/toolbars.xml.
2. Localisez la section Code view toolbar en recherchant le commentaire suivant :

   ```
   <!-- Code view toolbar -->
   ```

3. Copiez et collez les balises associées aux boutons pour que ces derniers s'affichent dans l'ordre requis dans la barre d'outils.
4. Enregistrez le fichier.

Pour supprimer un bouton :

1. Ouvrez le fichier Configuration/Toolbars/toolbars.xml.
2. Localisez la section Code view toolbar en recherchant le commentaire suivant :

   ```
   <!-- Code view toolbar -->
   ```

3. Entourez le bouton à supprimer d'un commentaire.

L'exemple suivant illustre un bouton entouré de commentaires.

```
<!-- supprimer le bouton de la barre d'outils Mode Code
  <button id="DW_ExpandAll"
    image="Toolbars/images/MM/T_ExpandAll_Sm_N.png"
    disabledImage="Toolbars/images/MM/T_ExpandAll_Sm_D.png"
    tooltip="Expand All"
    domRequired="false"
    enabled="dw.getFocus(true) == 'textView' || dw.getFocus(true) == ¬
      'html'"
    command="if (dw.getFocus(true) == 'textView' || dw.getFocus(true) ¬
      == 'html') dw.getDocumentDOM().source.expandAllCodeFragments();"
    update="onViewChange" />
-->
```

4. Enregistrez le fichier.

Pour afficher un bouton qui n'apparaît pas sur la barre d'outils, il suffit de supprimer le commentaire qui entoure le bouton dans le fichier XML.

CHAPITRE 2

Personnalisation du mode Code

2

En mode Code, Macromedia Dreamweaver 8 permet de rédiger rapidement un code lisible et sans erreur grâce aux deux outils que sont les indicateurs de code et la coloration du code. De plus, Dreamweaver effectue un contrôle de validité du code saisi par rapport aux navigateurs cibles spécifiés et permet de modifier le formatage HTML par défaut.

Vous pouvez personnaliser les outils d'indication et de coloration du code en modifiant leurs fichiers d'implémentation au format XML. Vous pouvez ajouter des éléments aux menus Indicateurs de code en ajoutant des entrées au fichier CodeHints.xml. Vous pouvez modifier les modèles de couleur par l'intermédiaire du fichier de style de coloration du code, Colors.xml, ou encore modifier les modèles de coloration du code ou en ajouter de nouveaux par l'intermédiaire des fichiers de syntaxe de coloration du code (par exemple, CodeColoring.xml). Vous pouvez également modifier le fichier de profil CSS (feuille de style en cascade) de votre navigateur cible de manière à influer sur le système de validation des valeurs et des propriétés CSS dans Dreamweaver. Vous pouvez enfin modifier le formatage HTML par défaut de Dreamweaver par l'intermédiaire de la boîte de dialogue Préférences. Les sections suivantes décrivent les procédures de personnalisation de ces différentes fonctions.

Indicateurs de code

Les indicateurs de code sont des menus contextuels qui s'affichent dans Dreamweaver lorsque vous tapez certains caractères en mode Code. Ils vous évitent de saisir tout le texte en proposant une liste de chaînes susceptibles de compléter la chaîne que vous tapez. Si la chaîne que vous tapez apparaît dans le menu, sélectionnez-la et appuyez sur Entrée ou Retour pour compléter votre saisie. Si vous tapez <, par exemple, un menu contextuel affiche une liste des noms de balises. Plutôt que de taper le reste du nom de la balise, vous pouvez la sélectionner dans le menu pour l'inclure à votre texte.

Dreamweaver charge les menus Indicateurs de code à partir du fichier CodeHints.xml enregistré dans le dossier Configuration/CodeHints. Vous pouvez ajouter des menus Indicateurs de code dans Dreamweaver en les définissant dans le fichier CodeHints.xml. Une fois le contenu du fichier CodeHints.xml chargé, vous pouvez également ajouter dynamiquement de nouveaux menus Indicateurs de code via JavaScript. Par exemple, le code JavaScript ajoute des données à la liste des variables de session dans le panneau Liaisons. Vous pouvez utiliser le même code pour ajouter un menu Indicateurs de code. Dans ce cas, lorsqu'un utilisateur tape « `Session.` » en mode Code, un menu de variables de session s'affiche dans Dreamweaver. Pour obtenir des informations sur l'ajout ou la modification d'un menu Indicateurs de code avec JavaScript, voir Fonctions de code dans le *Guide des API de Dreamweaver*.

Certains types de menus Indicateurs de code ne s'expriment pas dans le fichier XML ou l'API JavaScript de Dreamweaver. Le fichier CodeHints.xml et l'API JavaScript contiennent un sous-ensemble utile du moteur Indicateurs de code, mais certaines fonctionnalités Dreamweaver ne sont pas accessibles. Par exemple, comme il n'existe pas d'accroche JavaScript pouvant afficher un sélecteur de couleur, Dreamweaver ne peut pas exprimer le menu Valeurs des attributs à l'aide de JavaScript. Vous pouvez uniquement afficher un menu d'éléments de texte permettant d'insérer du texte.

REMARQUE	Lorsque vous insérez du texte, le pointeur d'insertion se place après la chaîne insérée.

Fichier CodeHints.xml

Le fichier CodeHints.xml contient les entités suivantes :

- Une liste de tous les groupes de menus

  Lorsque vous sélectionnez la catégorie Indicateurs de code dans la boîte de dialogue Préférences, la liste des groupes de menus s'affiche dans Dreamweaver. Pour ouvrir la boîte de dialogue Préférences, sélectionnez Edition > Préférences. Dreamweaver MX propose les groupes ou types de menus Indicateurs de code suivants : Noms des balises, Noms des attributs, Valeurs des attributs, Arguments des fonctions, Méthodes quant aux objets et variables et Entités HTML.

- La description de chaque groupe de menus

  Cette description relative à la catégorie Indicateurs de code apparaît dans la boîte de dialogue Préférences lorsque vous sélectionnez le groupe de menus dans la liste. La description de l'entrée sélectionnée apparaît au-dessous de la liste de groupes de menus.

- Des menus Indicateurs de code

  Un menu se compose d'un modèle qui déclenche le menu Indicateurs de code et d'une liste d'éléments de menu. Par exemple, le modèle "&" peut déclencher un menu "&", ">", "<".

Le format du fichier CodeHints.xml est illustré dans l'exemple suivant :

```
<codehints>
<menugroup name="HTML Entities" enabled="true"
  id="CodeHints_HTML_Entities">
  <description>
  <![CDATA[ When you type a '&', a drop-down menu shows
    a list of HTML entities. The list of HTML entities
    is stored in Configuration/CodeHints.xml. ]]>
  </description>

  <menu pattern="&amp;">
       <menuitem value="&amp;amp;" texticon="&amp;"/>
       <menuitem value="&amp;lt;" icon="lessThan.gif"/>
  </menu>
</menugroup>

<menugroup name="Tag Names" enabled="true" id="CodeHints_Tag_Names">
  <description>
       <![CDATA[ When you type '<', a drop-down menu shows
       all possible tag names.  You can edit the list of tag
       names using the
       <a href="javascript:dw.popupTagLibraryEditor()"> Tag Library Editor
       </a>]]>
    </description>
</menugroup>

<menugroup name="Function Arguments" enabled="true"
    id="CodeHints_Function_Arguments">
  <description>
    ...
  </description>
  <function pattern="ArraySort(array, sort_type, sort_order)"
        doctypes="CFML"/>
  <function pattern="Response.addCookie(Cookie cookie)"
        doctypes="JSP"/>
</menugroup>
<codehints>
```

Balises des indicateurs de code

Le fichier CodeHints.xml contient les balises suivantes qui définissent les menus Indicateurs de code. Vous pouvez utiliser ces balises pour définir les menus Indicateurs de code.

<codehints>

Description

La balise `codehints` est la racine du fichier `CodeHints.xml`.

Attributs

Néant

Contenu

Une ou plusieurs balises `menugroup`.

Contenant

Néant

Exemple

```
<codehints>
```

<menugroup >

Description

Chaque balise `menugroup` correspond à un type de menu. Vous pouvez afficher les types de menu définis dans Dreamweaver en choisissant la catégorie Indicateurs de code dans la boîte de dialogue Préférences. Pour afficher la boîte de dialogue Préférences, choisissez Préférences dans le menu Edition.

Vous pouvez créer un groupe de menus ou ajouter des menus à un groupe existant. Les groupes de menus sont des collections logiques de menus que l'utilisateur peut activer ou désactiver dans la boîte de dialogue Préférences.

Attributs

`name, enabled, id`

- L'attribut `name` est le nom localisé qui apparaît dans la liste des groupes de menu de la catégorie Indicateurs de code de la boîte de dialogue Préférences.
- L'attribut `enabled` indique si le groupe de menus est actuellement coché ou activé. Une coche apparaît en regard d'un groupe de menus s'il est activé dans la catégorie Indicateurs de code de la boîte de dialogue Préférences. Pour activer un groupe de menus, assignez-lui la valeur `true` ; pour le désactiver, assignez-lui la valeur `false`.
- L'attribut `id` est un identificateur non localisé qui se rapporte au groupe de menus.

Contenu

Balises `description`, `menu` et `function`.

Contenant

Balise `codehints`.

Exemple

```
<menugroup name="Session Variables" enabled="true" id="Session_Code_Hints">
```

<description>

Description

La balise `description` contient du texte affiché par Dreamweaver lorsque vous sélectionnez le groupe de menus dans la boîte de dialogue Préférences. La description apparaît au-dessous de la liste de groupes de menus. Elle peut contenir une balise `a` dans laquelle l'attribut `href` doit être une URL JavaScript exécutée par Dreamweaver si l'utilisateur clique sur le lien. Utilisez la structure XML CDATA pour inclure des caractères spéciaux ou illégaux dans la chaîne afin qu'ils soient considérés comme du texte par Dreamweaver.

Attributs

Néant

Contenu

Texte de la description.

Contenant

Balise `menugroup`.

Exemple

```
<description>
<![CDATA[ To add or remove tags and attributes, use the <a
  href="javascript:dw.tagLibrary.showTagLibraryEditor()">Tag Library
  Editor</a>.
  ]]>
</description>
```

<menu>

Description

Cette balise décrit un menu contextuel. Dreamweaver affiche le menu lorsque l'utilisateur tape le dernier caractère de la chaîne dans l'attribut pattern. Par exemple, le menu qui affiche le contenu d'une variable de session peut avoir un attribut pattern égal à « `Session.` ».

Attributs

`pattern, doctypes, casesensitive`

- L'attribut `pattern` spécifie le modèle des caractères tapés entraînant l'affichage du menu Indicateurs de code dans Dreamweaver. Si le premier caractère est une lettre, un chiffre ou un trait de soulignement, Dreamweaver affiche le menu uniquement si le caractère qui précède le modèle dans le document n'est pas une lettre, un chiffre ou un trait de soulignement. Par exemple, si le modèle est « `Session.` », Dreamweaver n'affiche pas le menu lorsque l'utilisateur tape « `ma_Session.` ».
- L'attribut `doctypes` indique que le menu est actif uniquement pour les types de documents spécifiés. Cet attribut permet de spécifier différentes listes de noms de fonctions pour ASP-JavaScript (ASP-JS), Java Server Pages (JSP), Macromedia ColdFusion, etc. Vous pouvez spécifier l'attribut `doctypes` en tant que liste d'ID de types de documents séparés par des virgules. Reportez-vous au fichier MMDocumentTypes.xml dans le dossier Configuration/Documenttypes de Dreamweaver pour consulter la liste des types de documents Dreamweaver.
- L'attribut `casesensitive` indique si le modèle fait la distinction entre les majuscules et les minuscules. Les valeurs possibles pour l'attribut `casesensitive` sont `true`, `false` ou un sous-ensemble de la liste séparée par des virgules spécifiée pour l'attribut `doctypes`. La liste des types de documents vous permet de spécifier si le modèle fait la distinction entre majuscules et minuscules pour certains types de documents plutôt que d'autres. Par défaut, la valeur est `false` si vous omettez cet attribut. Si vous définissez la valeur de l'attribut `casesensitive` sur `true`, le menu Indicateurs de code s'affiche uniquement si le texte tapé par l'utilisateur correspond exactement au modèle spécifié par l'attribut pattern. Si vous définissez la valeur de l'attribut `casesensitive` sur `false`, le menu s'affiche même si le modèle est en minuscules et le texte en majuscules.

Contenu

Balise `menuitem`.

Contenant

Balise `menugroup`.

Exemple

```
<menu pattern="CGI." doctypes="ColdFusion">
```

<menuitem>

Description

Cette balise indique le texte d'un élément dans un menu contextuel Indicateurs de code. La balise `menuitem` indique également la valeur à insérer dans le texte lorsque vous sélectionnez l'élément.

Attributs

`label, value` {`icon`}, {`texticon`}

- L'attribut `label` correspond à la chaîne affichée dans le menu contextuel de Dreamweaver.
- L'attribut `value` correspond à la chaîne insérée par Dreamweaver dans le document lorsque vous sélectionnez l'élément du menu. Lorsque l'utilisateur sélectionne l'élément du menu et appuie sur Entrée ou Retour, Dreamweaver remplace tout le texte tapé depuis l'affichage du menu. L'utilisateur a tapé les caractères correspondant au modèle avant l'affichage du menu, si bien que Dreamweaver ne les insère pas une seconde fois. Par exemple, si vous souhaitez insérer `&amp`, l'entité HTML de l'esperluette (&), vous pouvez définir les balises `menu` et `menuitem` suivantes :

  ```
  <menu pattern="&amp;">
  <menuitem label="&amp;amp;" value="amp;" texticon="&amp;"/>
  ```

  L'attribut de valeur n'inclut pas l'esperluette (&), car l'utilisateur l'a tapée avant que le menu ne s'affiche.
- L'attribut `icon` (facultatif) spécifie le chemin d'accès à un fichier image affiché par Dreamweaver sous la forme d'une icône située à gauche du texte du menu. L'emplacement de ce fichier est défini par une URL (dossier Configuration).
- L'attribut `texticon` (facultatif) affiche une chaîne de texte dans la zone d'icône (et non un fichier image). Cet attribut est utilisé pour le menu Entités HTML.

Contenu

Néant

Contenant

Balise `menu`.

Exemple

```
<menuitem label="CONTENT_TYPE" value="&quot;CONTENT_TYPE&quot;)"
  icon="shared/mm/images/hintMisc.gif" />
```

<function>

Description

Cette balise remplace la balise menu pour la spécification des arguments de fonction et des méthodes d'objet d'un menu contextuel Indicateurs de code. Lorsque vous tapez un nom de fonction ou de méthode en mode Code, Dreamweaver affiche un menu de prototypes de fonction, indiquant l'argument actuel en caractère gras. Chaque fois que vous tapez une virgule, Dreamweaver met à jour le menu de manière à afficher l'argument suivant en gras. Par exemple, si vous avez tapé le nom de fonction `ArrayAppend` dans un document ColdFusion, le menu Indicateurs de code affiche `ArrayAppend(`**`array`**`, value)`. Une fois la virgule suivant `array` saisie, le menu se met à jour et affiche `ArrayAppend(array, `**`value`**`)`.

Dans le cas des méthodes d'objet, Dreamweaver affiche un menu regroupant les méthodes définies pour l'objet dont vous tapez le nom.

L'ensemble de fonctions reconnues est enregistré dans le fichier CodeHints.xml qui se trouve dans le dossier Configuration de Dreamweaver.

Attributs

`pattern, doctypes, casesensitive`

- L'attribut `pattern` spécifie le nom de la fonction et sa liste d'arguments. S'il est utilisé avec des méthodes, l'attribut pattern décrit le nom de l'objet et de la méthode, ainsi que les arguments de cette dernière. Avec un nom de fonction, le menu Indicateurs de code s'affiche lorsque l'utilisateur tape `functionname(`. Le menu affiche la liste des arguments relatifs à la fonction. S'il est utilisé avec une méthode d'objet, le menu Indicateurs de code s'affiche lorsque l'utilisateur tape `objectname.` (point final compris). Ce menu indique les méthodes qui ont été spécifiées pour l'objet. Ensuite, le menu Indicateurs de code affiche une liste des arguments de la méthode, comme il le fait avec une fonction.
- L'attribut `doctypes` indique que le menu est actif uniquement pour les types de documents spécifiés. Cet attribut permet de spécifier différentes listes de noms de fonctions pour ASP-JavaScript (ASP-JS), Java Server Pages (JSP), Macromedia ColdFusion, etc. Vous pouvez spécifier l'attribut `doctypes` en tant que liste d'ID de types de documents séparés par des virgules. Pour obtenir une liste des types de documents Dreamweaver, voir le fichier Dreamweaver Configuration/Documenttypes/MMDocumentTypes.xml.

- L'attribut `casesensitive` indique si le modèle fait la distinction entre les majuscules et les minuscules. Les valeurs possibles pour l'attribut `casesensitive` sont `true`, `false` ou un sous-ensemble de la liste séparée par des virgules spécifiée pour l'attribut `doctypes`. La liste des types de documents vous permet de spécifier si le modèle fait la distinction entre majuscules et minuscules pour certains types de documents plutôt que d'autres. Par défaut, la valeur est `false` si vous omettez cet attribut. Si vous définissez la valeur de l'attribut `casesensitive` sur `true`, le menu Indicateurs de code s'affiche uniquement si le texte tapé par l'utilisateur correspond exactement au modèle spécifié par l'attribut du modèle. Si vous définissez la valeur de l'attribut `casesensitive` sur `false`, le menu s'affiche même si le modèle est en minuscules et le texte en majuscules.

Contenu

Néant

Contenant

Balise `menugroup`.

Exemple

```
// exemple de fonction
<function pattern="CreateDate(year, month, day)" DOCTYPES="ColdFusion" />
// exemple de méthode d'objet
<function pattern="application.getAttribute(String name)" DOCTYPES="JSP" />
```

Coloration du code

Dreamweaver permet de personnaliser ou d'étendre les modèles de coloration affichés en mode Code de manière à ajouter de nouveaux mots-clés à un modèle ou à ajouter des modèles de coloration du code correspondant à de nouveaux types de documents. Par exemple, si vous développez des fonctions JavaScript dans le script côté client, vous pouvez ajouter le nom de ces fonctions à la section des mots-clés de manière à les afficher dans la couleur indiquée dans les préférences. De même, si vous développez un nouveau langage de programmation pour un serveur d'application et que vous souhaitez distribuer un nouveau type de document dans le but d'aider les utilisateurs de Dreamweaver à l'inclure dans le processus de création de pages, vous pouvez ajouter un modèle de coloration du code correspondant à ce type de document.

Dreamweaver contient la fonction JavaScript `dreamweaver.reloadCodeColoring()` permettant de recharger les fichiers XML de coloration du code susceptibles d'avoir été modifiés manuellement. Pour plus d'informations sur cette fonction, voir le *Guide des API de Dreamweaver*.

Pour mettre à jour un modèle de coloration du code ou ajouter un nouveau modèle, vous devez modifier les fichiers de définition de coloration du code.

Fichiers de coloration du code

Dreamweaver définit les styles et les modèles de coloration du code dans des fichiers XML se trouvant dans le dossier Configuration/CodeColoring. Un fichier de style de coloration du code définit le style des champs indiqués dans les définitions de syntaxe. Le nœud racine est `<codeColors>`. Un fichier de modèle de coloration du code définit la syntaxe de coloration du code ; le nœud racine est `<codeColoring>`.

Le fichier de style de coloration du code de Dreamweaver est Colors.xml. Les fichiers de syntaxe de coloration du code de Dreamweaver sont CodeColoring.xml, ASP JavaScript.xml, ASP VBScript.xml, ASP.NET CSharp.xml et ASP.NET VB.xml.

REMARQUE La coloration du code illustrée dans les exemples ci-dessous apparaît uniquement sur les impressions en couleur. Pour voir la coloration du code illustrée dans ces exemples, voir l'aide de Dreamweaver › Extensions › Extension de Dreamweaver ou consultez le fichier PDF *Extension de Dreamweaver* dans le dossier Documentation du CD d'installation.

L'extrait du fichier Colors.xml ci-dessous illustre la hiérarchie de balises dans un fichier de style de coloration du code :

```
<codeColors>
  <colorGroup>
    <syntaxColor id="CodeColor_HTMLEntity" bold="true" italic="true" />
    <syntaxColor id="CodeColor_JavascriptNative" text="#009999" />
    <syntaxColor id="CodeColor_JavascriptNumber" text="#FF0000" />

…

    <tagColor id="CodeColor_HTMLStyle" text="#990099" />
    <tagColor id="CodeColor_HTMLTable" text="#009999" />
    <syntaxColor id="CodeColor_HTMLComment" text="#999999" italic="true" /
  >

…

  </colorGroup>
</codeColors>
```

Les couleurs sont indiquées sous la forme de valeurs hexadécimales rouge-vert-bleu (RVB). Par exemple, l'instruction `text="009999"` figurant dans le code XML ci-dessus assigne une couleur bleu-vert (sarcelle) à l'ID `"CodeColor_JavascriptNative"`.

L'extrait du fichier codeColoring.xml ci-dessous illustre la hiérarchie de balises dans un fichier de modèle de coloration du code ainsi que la relation entre le fichier de styles et le fichier de modèle :

```
<codeColoring>
  <scheme name="Text" id="Text" doctypes="Text" priority="1">
    <ignoreTags>Yes</ignoreTags>
    <defaultText name="Text" id="CodeColor_TextText" />
    <sampleText doctypes="Text">
<![CDATA[Default file syntax highlighting.
The quick brown fox
jumped over the lazy dog.
]]>
    </sampleText>
  </scheme>

<scheme name="HTML" id="HTML" doctypes="ASP.NET_VB,ASP.NET_CSharp,ASP-
  JS,ASP-
  VB,ColdFusion,CFC,HTML,JSP,EDML,PHP_MySQL,DWTemplate,LibraryItem,WML"
  priority="50">
    <ignoreCase>Yes</ignoreCase>
    <ignoreTags>No</ignoreTags>
    <defaultText name="Text" id="CodeColor_HTMLText" />
    <defaultTag name="Other Tags" id="CodeColor_HTMLTag" />
    <defaultAttribute />
    <commentStart name="Comment" id="CodeColor_HTMLComment"><![CDATA[<!--
  ]]></commentStart>
. . .
  <tagGroup name="HTML Anchor Tags" id="CodeColor_HTMLAnchor"
  taglibrary="DWTagLibrary_html" tags="a" />
  <tagGroup name="HTML Form Tags" id="CodeColor_HTMLForm"
  taglibrary="DWTagLibrary_html" tags="select,form,input,option,textarea"
  />
. . .
</codeColoring>
```

Notez que les balises `syntaxColor` et `tagColor` du fichier Colors.xml assignent des valeurs de couleur et de style à une valeur de chaîne `id`. La valeur `id` permet alors d'assigner un style à une balise scheme dans le fichier codeColoring.xml. Par exemple, l'`id` de la balise `defaultTag` figurant dans l'extrait codeColoring.xml est `"CodeColor_HTMLComment"`. Dans le fichier Colors.xml, la valeur `text=` attribuée à la valeur `id` de `"CodeColor_HTMLComment"` est `"#999999"`, c'est-à-dire gris.

Dreamweaver contient les modèles de coloration du code suivants : Par défaut, HTML, JavaScript, ASP_JavaScript, ASP_VBScript, JSP et ColdFusion. La valeur `id` du modèle par défaut est égale à `"Text"`. Ce modèle est utilisé pour les types de documents dont le modèle de coloration du code n'est pas défini.

Un fichier de coloration du code contient les balises suivantes.

<scheme>

Description

La balise `scheme` indique la coloration du code assignée à un bloc de texte de code. Un fichier peut contenir plusieurs modèles spécifiant différentes colorations pour différents langages de script ou de balise. Chaque modèle possède une priorité vous permettant d'imbriquer un bloc de texte correspondant à un modèle dans un bloc de texte correspondant à un autre modèle.

Attributs

`name`, `id`, `priority`, `doctypes`

- `name="`*`nom_modèle`*`"` Chaîne attribuant un nom au modèle. Le nom du modèle s'affiche dans la boîte de dialogue Modifier le modèle de coloration de Dreamweaver. Dreamweaver affiche une combinaison de nom de modèle et de champ, par exemple `HTML Comment`. Si vous n'attribuez pas de nom, les champs du modèle n'apparaissent pas dans la boîte de dialogue Modifier le modèle de coloration. Pour plus d'informations sur la boîte de dialogue Modifier le modèle de coloration, voir *Modification des modèles*, page 93.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.
- `priority="`*`chaîne`*`"` Valeurs comprises entre "1" et "99". La priorité la plus haute est "1". Indique la priorité d'un modèle. Les blocs se trouvant au sein de blocs dont la priorité est supérieure sont ignorés ; les blocs se trouvant au sein de blocs dont la priorité est inférieure ou égale sont prioritaires. Si vous n'indiquez pas de priorité, la valeur par défaut est "50".
- `doctypes="`*`liste_doc`*`"` Facultatif. Indique une liste des types de documents, séparés par des virgules, auxquels s'applique le modèle de coloration du code. Cette valeur est indispensable pour la résolution des conflits impliquant différents blocs de début et de fin dont les extensions sont identiques.

Contenu

`blockEnd`, `blockStart`, `brackets`, `charStart`, `charEnd`, `charEsc`, `commentStart`, `commentEnd`, `cssProperty`, `cssSelector`, `cssValue`, `defaultAttribute`, `defaultText`, `endOfLineComment`, `entity`, `functionKeyword`, `idChar1`, `idCharrest`, `ignoreCase`, `ignoreMMTParam`, `ignoreTags`, `keywords`, `numbers`, `operators`, `regexp`, `sampletext`, `searchPattern`, `stringStart`, `stringEnd`, `stringEsc`, `urlProtocol`, `urlProtocols`

Contenant

Balise `codeColoring`.

Exemple

```
<scheme name="Text" id="Text" doctypes="Text" priority="1">
```

<blockEnd>

Description

Facultatif. Texte délimitant la fin du bloc de texte de ce modèle. Les balises `blockEnd` et `blockStart` doivent être équilibrées et leur combinaison doit être unique. La casse des valeurs indiquées n'est pas prise en compte. La valeur de la balise `blockEnd` peut être composée d'un seul caractère. Vous pouvez créer plusieurs instances de cette balise. Pour plus d'informations sur les chaînes `blockEnd`, voir *Caractères génériques*, page 89.

Attributs

Néant

Exemple

```
<blockEnd><![CDATA[--->]]></blockEnd>
```

<blockStart>

Description

Facultatif. Spécifiée uniquement si le modèle de coloration peut être imbriqué dans un autre modèle de coloration. Les balises `blockEnd` et `blockStart` doivent être équilibrées et leur combinaison doit être unique. La casse des valeurs indiquées n'est pas prise en compte. La valeur de la balise `blockStart` doit comporter au minimum deux caractères. Vous pouvez créer plusieurs instances de cette balise. Pour plus d'informations sur les chaînes `blockStart`, voir *Caractères génériques*, page 89. Pour plus d'informations sur l'attribut `blockStart scheme`, voir *Coloration des délimiteurs de bloc de modèle*, page 85.

Attributs

`canNest`, `doctypes`, `id`, `name`, `scheme`

- `canNest` Indique si le modèle peut s'imbriquer. Les valeurs sont `"Yes"` ou `"No"`. La valeur par défaut est `"No"`.
- `doctypes="`*`type_doc1, type_doc2,…`*`"` Obligatoire. Indique une liste des types de documents, séparés par des virgules, dans lesquels vous pouvez imbriquer ce modèle de coloration du code. Les types de documents sont définis dans le fichier Configuration/Document Types/MMDocumentTypes.xml de Dreamweaver.
- `id="`*`chaîne_id`*`"` Obligatoire lorsque `scheme="customText"`. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

- `name="nom_affiché"` Chaîne apparaissant dans la boîte de dialogue Modifier le modèle de coloration lorsque `scheme="customText"`.
- `scheme` Obligatoire. Définit la coloration des chaînes `blockStart` et `blockEnd`. Pour plus d'informations sur les valeurs d'attribut scheme possibles, voir *Coloration des délimiteurs de bloc de modèle*, page 85.

Exemple

```
<blockStart doctypes="ColdFusion,CFC" scheme="innerText"
  canNest="Yes"><![CDATA[<!---]]></blockStart>
```

<brackets>

Description

Liste des caractères représentant des crochets.

Attributs

`name`, `id`

- `name="nom_crochet"` Chaîne assignant un nom à la liste des crochets.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<brackets name="Bracket" id="CodeColor_JavaBracket"><![CDATA[{[()]}]]></
  brackets>
```

<charStart>

Description

Contient une chaîne de texte représentant le délimiteur du début d'un caractère. Les balises `charStart` et `charEnd` doivent être équilibrées. Vous pouvez indiquer plusieurs paires de balises `charStart` … `charEnd`.

Attributs

Néant

Exemple

```
<charStart><![CDATA[']]></charStart>
```

‹charEnd›

Description

Contient une chaîne de texte représentant le délimiteur de fin d'un caractère. Les balises `charStart` et `charEnd` doivent être équilibrées. Vous pouvez indiquer plusieurs paires de balises `charStart` … `charEnd`.

Attributs

Néant

Exemple

```
<charEnd><![CDATA[']]></charEnd>
```

‹charEsc›

Description

Contient une chaîne de texte représentant un caractère d'échappement. Vous pouvez insérer plusieurs balises `charEsc`.

Attributs

Néant

Exemple

```
<charEsc><![CDATA[\]]></charEsc>
```

‹commentStart›

Description

Chaîne de texte délimitant le début d'un bloc de commentaire. Les balises `commentStart` et `commentEnd` doivent être équilibrées. Vous pouvez indiquer plusieurs paires de balises `commentStart`…`/commentEnd`.

Attributs

Néant

Exemple

```
<commentStart><![CDATA[<%--]]></commentStart>
```

<commentEnd>

Description

Chaîne de texte délimitant la fin d'un bloc de commentaire. Les balises `commentStart` et `commentEnd` doivent être équilibrées. Vous pouvez indiquer plusieurs paires de balises `commentStart.../commentEnd`.

Attributs

Néant

Exemple

```
<commentEnd><![CDATA[--%>]]></commentEnd>
```

<cssImport/>

Description

Balise vide indiquant la règle de coloration du code de la fonction `@import` de l'élément `style` dans une CSS.

Attributs

`name`, `id`

`name="`*`nom_cssImport`*`"` Chaîne assignant un nom à la fonction `@import` CSS.

`id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<cssImport name="@import" id="CodeColor_CSSImport" />
```

<cssMedia/>

Description

Balise vide indiquant la règle de coloration du code de la fonction `@media` de l'élément `style` dans une CSS.

Attributs

`name`, `id`

- `name="`*`nom_cssMedia`*`"` Chaîne assignant un nom à la fonction `@media` CSS.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<cssMedia name="@media" id="CodeColor_CSSMedia" />
```

<cssProperty/>

Description

Balise vide indiquant les règles CSS et regroupant des attributs de coloration du code.

Attributs

`name`, `id`

- `name="`*`nom_cssProperty`*`"` Chaîne assignant un nom à la propriété CSS.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Préférence de couleur de code

Propriété CSS

Exemple

```
<cssProperty name="Property" id="CodeColor_CSSProperty" />
```

<cssSelector/>

Description

Balise vide indiquant les règles CSS et regroupant des attributs de coloration du code.

Attributs

`name`, `id`

- `name="`*`nom_cssSelector`*`"` Chaîne assignant un nom au sélecteur CSS.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<cssSelector name="Selector" id="CodeColor_CSSSelector" />
```

<cssValue/>

Description

Balise vide indiquant les règles CSS et regroupant des attributs de coloration du code.

Attributs

`name`, `id`

- `name="nom_cssValue"` Chaîne assignant un nom à la valeur CSS.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<cssValue name="Value" id="CodeColor_CSSValue" />
```

<defaultAttribute>

Description

Facultatif. Cette balise s'applique uniquement à la syntaxe de balise (c'est-à-dire une syntaxe pour laquelle `ignoreTags="No"`). Si cette balise figure dans le code, tous les attributs de balise sont colorés suivant le style qui lui est assigné. Si ce n'est pas le cas, la couleur des attributs est identique à celle de la balise correspondante.

Attributs

`name`

- Chaîne assignant un nom à l'attribut par défaut.

Exemple

```
<defaultAttribute name="Attribute"/>
```

<defaultTag>

Description

Cette balise permet de spécifier la couleur et le style par défaut des balises d'un modèle.

Attributs

`name`, `id`

- `name="nom_affiché"` Chaîne affichée par Dreamweaver dans l'éditeur de coloration du code.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<defaultTag name="Other Tags" id="CodeColor_HTMLTag" />
```

<defaultText/>

Description

Facultatif. Si cette balise figure dans le code, les chaînes de texte définies par toute autre balise sont colorées suivant le style assigné à cette balise. Si ce n'est pas le cas, le texte est de couleur noire.

Attributs

`name`, `id`

- `name="`*`nom_cssSelector`*`"` Chaîne assignant un nom au sélecteur CSS.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<defaultText name="Text" id="CodeColor_TextText" />
```

<endOfLineComment>

Description

Chaîne de texte délimitant le début d'un commentaire allant jusqu'à la fin de la ligne actuelle. Vous pouvez insérer plusieurs balises `endOfLineComment…/endOfLineComment`.

Attributs

Néant

Exemple

```
<endOfLineComment><![CDATA[//]]></endOfLineComment>
```

<entity/>

Description

Balise vide indiquant que les caractères spéciaux HTML doivent être reconnus et contenir des attributs de coloration.

Attributs

`name`, `id`

- `name="`*`nom_entité`*`"` Chaîne assignant un nom à l'entité.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<entity name="Special Characters" id="CodeColor_HTMLEntity" />
```

<functionKeyword>

Description

Identifie les mots-clés définissant une fonction. Ces mots-clés permettent à Dreamweaver de naviguer dans le code. Vous pouvez insérer plusieurs balises `functionKeyword`.

Attributs

`name`, `id`

- `name="`*`nom_functionKeyword`*`"` Chaîne assignant un nom au bloc `functionKeyword`.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<functionKeyword name="Function Keyword"
  id="CodeColor_JavascriptFunction">function</functionKeyword>
```

<idChar1>

Description

Liste de chacun des caractères identifiables par Dreamweaver comme premier caractère d'un identifiant.

Attributs

`name`, `id`

- `name="`*`nom_idChar1`*`"` Chaîne assignant un nom à la liste des caractères d'identifiant.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<idChar1>_$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</idChar1>
```

<idCharRest>

Description

Liste de caractères devant être identifiés comme les autres caractères de l'identificateur. Si la balise `idChar1` n'est pas spécifiée, tous les caractères de l'identifiant sont validés en fonction de cette liste.

Attributs

`name`, `id`

- `name="`*`nom_idCharRest`*`"` Chaîne assignant un nom au bloc `stringStart`.
- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<idCharRest name="Identifier"
  id="CodeColor_JavascriptIdentifier">_$abcdefghijklmnopqrstuvwxyzABCDEFGH
  IJKLMNOPQRSTUVWXYZ0123456789</idCharRest>
```

‹ignoreCase›

Description

Indique si la casse des caractères doit être ignorée lors de la comparaison des expressions et des mots-clés. Les valeurs sont `Yes` ou `No`. La valeur par défaut est `Yes`.

Attributs

Néant

Exemple

```
<ignoreCase>Yes</ignoreCase>
```

‹ignoreMMTParams›

Description

Indique si une couleur particulière doit être attribuée aux balises `MMTInstance:Param`, `<!--InstanceParam` ou <!-- `#InstanceParam`. Les valeurs sont `Yes` et `No` ; la valeur par défaut est `Yes`. Cette balise assure la coloration appropriée des pages utilisant des modèles.

Attributs

Néant

Exemple

```
<ignoreMMTParams>No</ignoreMMTParams>
```

<ignoreTags>

Description

Indique si les balises doivent être ignorées. Les valeurs sont `Yes` et `No` ; la valeur par défaut est `Yes`. Réglez la valeur sur `No` si la syntaxe correspond à un langage de balise délimité par < et >. Réglez la valeur sur `Yes` si la syntaxe correspond à un langage de programmation.

Attributs

Néant

Exemple

```
<ignoreTags>No</ignoreTags>
```

<isLocked>

Description

Indique si le texte correspondant au modèle est autorisé à être modifié en mode Code. Les valeurs sont `Yes` et `No`. La valeur par défaut est `No`.

Attributs

Néant

Exemple

```
<isLocked>Yes</isLocked>
```

<keyword>

Description

Chaîne de texte définissant un mot-clé. Vous pouvez insérer plusieurs balises `keyword`. Il n'existe aucune restriction liée au premier caractère d'un mot-clé ; en revanche, les caractères suivants doivent être `a-z`, `A-Z`, `0-9`, _, `$` ou `@`.

La couleur du code est spécifiée par les balises contenant `keyword`.

Attributs

Néant

Exemple

```
<keyword>.getdate</keyword>
```

<keywords>

Description

Liste des mots-clés correspondant au type spécifié dans l'attribut category. Vous pouvez insérer plusieurs balises `keywords`.

Attributs

`name`, `id`

- `name="nom_mots_clés"` Chaîne assignant un nom à la liste des mots-clés.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Contenu

```
<keyword></keyword>
```

Exemple

```
<keywords name="Reserved Keywords" id="CodeColor_JavascriptReserved">
    <keyword>break</keyword>
    <keyword>case</keyword>
</keywords>
```

<numbers/>

Description

Balise vide spécifiant les chiffres devant être reconnus et contenant des attributs de couleur.

Attributs

`name`, `id`

- `name="nom_chiffre"` Chaîne assignant un nom à la balise numbers.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<numbers name="Number" id="CodeColor_CFScriptNumber" />
```

<operators>

Description

Liste des caractères devant être reconnus comme opérateurs.

Attributs

`name`, `id`

- `name="nom_opérateur"` Chaîne assignant un nom à la liste des caractères opérateurs.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.

Exemple

```
<operators name="Operator" id="CodeColor_JavaOperator"><![CDATA[+-*/
  %<>!?:=&|^~]]></operators>
```

<regexp>

Description

Spécifie une liste de balises `searchPattern`.

Attributs

`name`, `id`, `delimiter`, `escape`

- `name="nom_stringStart"` Chaîne assignant un nom à la liste des chaînes de modèle de recherche.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.
- `delimiter` Caractère ou chaîne situé au début et à la fin d'une expression régulière.
- `escape` Caractère ou chaîne signalant le traitement de caractères spéciaux, également appelé caractère ou chaîne « d'échappement ».

Contenu

```
<searchPattern></searchPattern>
```

Exemple

```
<regexp name="RegExp" id="CodeColor_JavascriptRegexp" delimiter="/"
  escape="\\">
    <searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
    <searchPattern><![CDATA[=\s*/\e*\\/]]></searchPattern>
</regexp>
```

<sampleText>

Description

Texte servant à la représentation dans la fenêtre Aperçu de la boîte de dialogue Modifier le modèle de coloration. Pour plus d'informations sur la boîte de dialogue Modifier le modèle de coloration, voir *Modification des modèles*, page 93.

Attributs

`doctypes`

- `doctypes="doc_type1, doc_type2,..."` Types de documents pour lesquels cet échantillon apparaît.

Exemple

```
<sampleText doctypes="JavaScript"><![CDATA[/* JavaScript */
function displayWords(arrayWords) {
  for (i=0; i < arrayWords.length(); i++) {
    // commentaire en ligne
    alert("Word " + i + " is " + arrayWords[i]);
  }
}

var tokens = new Array("Bonjour", "Monde");
displayWords(tokens);
]]></sampleText>
```

<searchPattern>

Description

Chaîne de caractères définissant un modèle de recherche régulier à l'aide des caractères génériques pris en charge. Vous pouvez insérer plusieurs balises `searchPattern`.

Attributs

Néant

Contenant

Balise `regexp`.

Exemple

```
<searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
```

<stringStart>

Description

Ces balises contiennent une chaîne de texte représentant le délimiteur du début d'une chaîne. Les balises `stringStart` et `stringEnd` doivent être équilibrées. Vous pouvez indiquer plusieurs paires de balises `stringStart` … `stringEnd`.

Attributs

`name, id, wrap`

- `name="nom_stringStart"` Chaîne assignant un nom au bloc `stringStart`.
- `id="chaîne_id"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.
- `wrap="true"` ou `"false"`. Définit la reconnaissance ou non des chaînes de texte placées sur la ligne suivante. La valeur par défaut est `"true"`.

Exemple

```
<stringStart name="Attribute Value"
  id="CodeColor_HTMLString"><![CDATA["]]></stringStart>
```

<stringEnd>

Description

Contient une chaîne de texte représentant le délimiteur de fin d'une chaîne de code. Les balises `stringStart` et `stringEnd` doivent être équilibrées. Vous pouvez indiquer plusieurs paires de balises `stringStart` … `stringEnd`.

Attributs

Néant

Exemple

```
<stringEnd><![CDATA["]]></stringEnd>
```

<stringEsc>

Description

Contient une chaîne de texte représentant le délimiteur du caractère d'échappement d'une chaîne. Vous pouvez insérer plusieurs balises `stringEsc`.

Attributs

Néant

Exemple

```
<stringEsc><![CDATA[\]]></stringEsc>
```

‹tagGroup›

Description

Cette balise regroupe une ou plusieurs balises auxquelles vous pouvez assigner une couleur et un style uniques.

Attributs

`id`, `name`, `taglibrary`, `tags`

- `id="`*`chaîne_id`*`"` Obligatoire. Chaîne d'identificateur permettant de mapper la couleur et le style à cet élément de syntaxe.
- `name="`*`nom_affiché`*`"` Chaîne affichée par Dreamweaver dans l'éditeur de coloration du code.
- `taglibrary="`*`id_bibliothèque_balises`*`"` Identificateur de la bibliothèque de balises à laquelle le groupe appartient.
- `tags="`*`liste_balise`*`"` Balise ou liste de balises séparées par des virgules comprenant le groupe de balises.

Exemple

```
<tagGroup name="HTML Table Tags" id="CodeColor_HTMLTable"
  taglibrary="DWTagLibrary_html"
  tags="table,tbody,td,tfoot,th,thead,tr,vspec,colw,hspec" />
```

Coloration des délimiteurs de bloc de modèle

L'attribut de modèle `blockStart` détermine la coloration des chaînes de début et de fin de bloc ou des délimiteurs de bloc. Les valeurs valides pour l'attribut `blockStart` sont répertoriées ci-après.

REMARQUE	Ne pas confondre l'attribut `blockStart.scheme` et la balise `scheme`.

innerText

Cette valeur indique à Dreamweaver que les délimiteurs de bloc doivent avoir la même couleur que le texte par défaut du modèle qu'ils renferment.

Le modèle Modèle illustre l'effet de ce modèle. Le modèle Modèle correspond à des blocs de code en lecture seule. La couleur grise indique qu'il est impossible de les modifier. Les délimiteurs de bloc, à savoir les chaînes `<!-- #EndEditable -->` et `<!-- #BeginEditable "..." , -->` sont aussi de couleur grise car il est également impossible de les modifier.

Exemple de code

```
<!-- #EndEditable -->
  <p><b><font size="+2">header</font></b></p>
  <!-- #BeginEditable "test" -->
  <p>Ce texte-ci est modifiable </p>
  <p> </p>
  <!-- #EndEditable -->
```

Exemple

```
<blockStart doctypes="ASP-JS,ASP-VB, ASP.NET_CSharp, ASP.NET_VB,
  ColdFusion,CFC, HTML, JSP,LibraryItem,PHP_MySQL"
  scheme="innerText"><![CDATA[<!--\s*#BeginTemplate]]></blockStart>
```

customText

Cette valeur indique à Dreamweaver qu'une couleur personnalisée doit être appliquée aux délimiteurs.

Exemple de code

Les délimiteurs de bloc de script PHP, qui apparaissent en rouge, illustrent l'effet de la valeur `customText` :

```
<?php
  if ($loginMsg <> "")
  echo $loginMsg;
  ?>
```

Exemple

```
<blockStart name="Block Delimiter" id="CodeColor_JavaBlock" doctypes="JSP"
  scheme="customText"><![CDATA[<%]]></blockStart>
```

outerTag

La valeur `outerTag` indique que les balises `blockStart` et `blockEnd` sont des balises à part entière et que Dreamweaver doit les colorer comme des balises du modèle qui les entoure.

Le modèle JavaScript, dans lequel les chaînes `<script>` et `</script>` correspondent aux balises `blockStart` et `blockEnd`, illustre cette valeur. Ce modèle correspond aux blocs de code JavaScript, dans lequel les balises ne sont pas reconnues ; les délimiteurs doivent donc prendre la coloration du modèle qui les entoure.

Exemple de code

```
<script language="JavaScript">
  // commentaire
  if (true)
  window.alert("Bonjour, Monde");
  </script>
```

Exemple

```
<blockStart doctypes="PHP_MySQL"
  scheme="outerTag"><![CDATA[<script\s+language="php">]]></blockStart>
```

innerTag

Cette valeur est similaire à la valeur `outerTag` ; la seule différence est que la coloration provient du modèle se trouvant à l'intérieur des délimiteurs. Elle est généralement utilisée pour la balise `html`.

nameTag

Cette valeur indique que la chaîne `blockStart` correspond au début d'une balise, que la chaîne `blockEnd` correspond à la fin d'une balise et que ces délimiteurs doivent être colorés suivant les paramètres de balise du modèle.

Ce type de modèle affiche les balises pouvant être imbriquées dans d'autres balises (par exemple, la balise `cfoutput`).

Exemple de code

```
<input type="text" name="zip"
  <cfif newRecord IS "no">
  <cfoutput query="employee"> Value="#zip#" </cfoutput>
  </cfif>
  >
```

Exemple

```
<blockStart doctypes="ColdFusion,CFC"
  scheme="nameTag"><![CDATA[<cfoutput\n]]></blockStart>
```

nameTagScript

Cette valeur est identique au modèle `nameTag` ; cependant, son contenu se compose de script, par exemple, des instructions ou des expressions d'assignation, par opposition aux paires d'attributs `name=value`.

Ce type de modèle affiche uniquement les balises contenant du script (par exemple, les balises ColdFusion `cfset`, `cfif` et `cfifelse`) et pouvant être imbriquées au sein d'autres balises.

Exemple de code

Voir l'exemple illustrant nameTag.

Exemple

```
<blockStart doctypes="ColdFusion,CFC"
  scheme="nameTagScript"><![CDATA[<cfset\n]]></blockStart>
```

Traitement des modèles

Dreamweaver regroupe trois modes de base de coloration du code : le mode CSS, le mode Script et le mode Balises.

Dans chaque mode, Dreamweaver applique uniquement la coloration du code à certains champs. Le tableau ci-dessous répertorie les champs auxquels s'applique la coloration du code pour chaque mode.

Champ	CSS	Balises	Script
defaultText		X	X
defaultTag		X	
defaultAttribute		X	
comment	X	X	X
string	X	X	X
cssProperty	X		
cssSelector	X		
cssValue	X		
character		X	X
function keyword			X
identifier			X
number		X	X
operator			X
brackets		X	X
keywords		X	X

Afin de faciliter le processus de définition de modèles, Dreamweaver permet de spécifier des caractères génériques et des caractères d'échappement.

Caractères génériques

La liste ci-dessous répertorie les caractères génériques pris en charge dans Dreamweaver, les chaînes permettant de les spécifier et une description de leur utilisation.

Caractère générique	Chaîne d'échappement	Description
Caractère générique	`\*`	Ignore tous les caractères de la règle jusqu'à la détection du caractère suivant le caractère générique. Par exemple, `<MMTInstance:Editable name="\*">` permet de détecter toutes les balises de ce type pour lesquelles l'attribut name est spécifié.
Caractère générique avec caractère d'échappement	`\e*x`	`x` correspond au caractère d'échappement. Même utilisation que le caractère générique, avec possibilité de spécifier un caractère d'échappement. Le caractère suivant le caractère d'espacement est ignoré. Le caractère suivant le caractère générique peut ainsi apparaître dans une chaîne sans correspondre au critère spécifié pour l'arrêt du traitement du caractère générique. Par exemple, `/\e*\\/` permet de reconnaître une expression JavaScript régulière commençant et terminant par une barre oblique (/) et pouvant contenir des barres obliques précédées d'une barre oblique inversée (\). La barre oblique inversée correspond au caractère d'échappement de coloration du code ; dès lors, vous devez la précéder d'une barre oblique inversée lorsque vous la spécifiez dans du code XML coloré.

Caractère générique	Chaîne d'échappement	Description
Espace blanc facultatif	`\s*`	Permet de reconnaître les espaces vides (même inexistants) ou les caractères de nouvelle ligne. Par exemple, `<!--\s*#include` permet de reconnaître des directives d'inclusion ASP, que l'expression `#include` soit précédée ou non d'espace blanc, car les deux cas sont valides. Les caractères génériques espace blanc correspondent à toutes les combinaisons d'espace blanc et de caractères de nouvelle ligne.
Espace blanc obligatoire	`\s+`	Permet de reconnaître un ou plusieurs espaces vides ou les caractères de nouvelle ligne. Par exemple, `<!--#include\s+virtual` permet de reconnaître des directives d'inclusion ASP contenant toutes les combinaisons d'espace blanc entre les expressions `#include` et `virtual`. Un espace blanc doit obligatoirement séparer ces deux expressions, mais il peut correspondre à toutes les combinaisons de caractères d'espace blanc valides. Les caractères génériques espace blanc correspondent à toutes les combinaisons d'espace blanc et de caractères de nouvelle ligne.

Caractères d'échappement

La liste ci-dessous répertorie les caractères d'échappement pris en charge dans Dreamweaver, les chaînes permettant de les spécifier et une description de leur utilisation.

Caractère d'échappement	Chaîne d'échappement	Description
Barre oblique inversée	`\\`	La barre oblique inversée (\) correspond au caractère d'échappement de coloration du code ; elle doit donc être ignorée pour être spécifiée dans une règle de coloration du code.
Espace blanc	`\s`	Ce caractère d'échappement correspond à tous les caractères non visibles, à l'exception de ceux correspondant au caractère de nouvelle ligne, comme les espaces et les tabulations. L'espace blanc facultatif et l'espace blanc obligatoire correspondent à la fois aux espaces blancs et aux caractères de nouvelle ligne.
Nouvelle ligne	`\n`	Ce caractère d'échappement correspond aux caractères de nouvelle ligne (également appelés sauts de ligne) et aux retours chariot.

Longueur maximale de chaîne

La longueur maximale autorisée pour les chaînes de données est de 100 caractères. Par exemple, la balise `blockEnd` suivante contient un caractère générique.

```
<blockEnd><![CDATA[<!--\s*#BeginEditable\s*"\*"\s*-->]]></blockEnd>
```

En partant du principe que les chaînes de caractère générique espace blanc facultatif (`\s*`) correspondent à un seul espace, généré automatiquement par Dreamweaver, la chaîne de données comprend 26 caractères, plus une chaîne de caractère générique (`\*`) correspondant au nom.

```
<!-- #BeginEditable "\*" -->
```

Le nom de région modifiable peut donc comporter jusqu'à 74 caractères, c'est-à-dire 100 caractères (longueur maximale) moins 26.

Priorité de modèle

Dreamweaver applique la coloration de la syntaxe de texte en mode Code en utilisant l'algorithme suivant :

1. Dreamweaver détermine le modèle de syntaxe initial en fonction du type de document du fichier actuel. Le type de document du fichier est déterminé en fonction de l'attribut `scheme.documentType`. Si aucun type n'est détecté, le modèle pour lequel `scheme.documentType = "Text"` est utilisé.
2. Les modèles peuvent être imbriqués s'ils comportent des paires `blockStart...blockEnd`. Tous les modèles pouvant être imbriqués et pour lesquels l'extension du fichier actuel est répertoriée dans l'un des attributs `blockStart.doctypes` sont activés pour le fichier actuel ; les autres modèles sont désactivés.

   REMARQUE Toutes les combinaisons `blockStart/blockEnd` doivent être uniques.

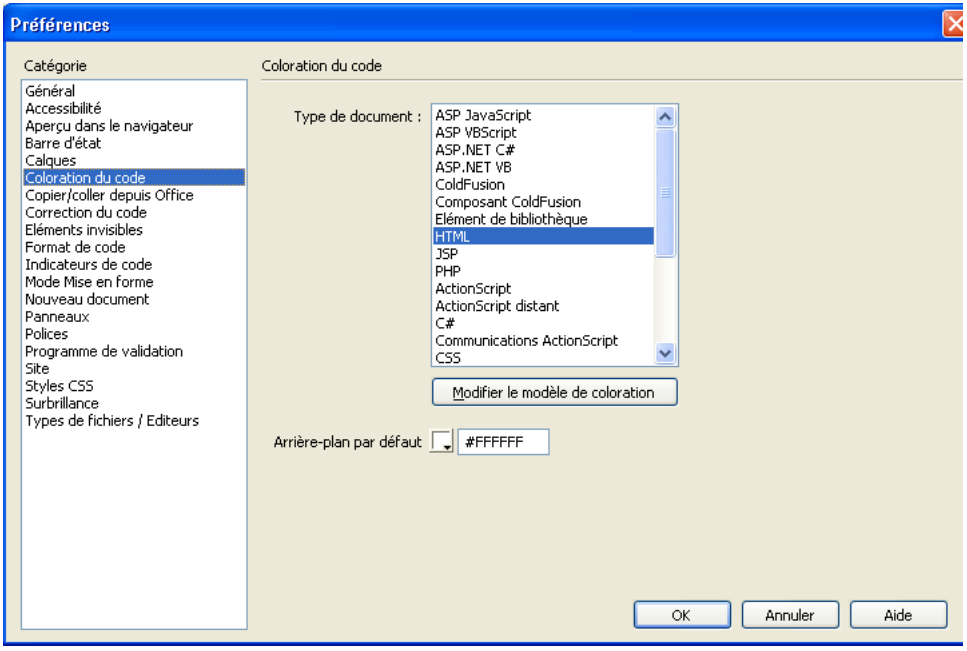
   Les modèles peuvent s'imbriquer au sein d'un autre modèle à condition que `scheme.priority` soit supérieure ou égale au modèle extérieur. Si la priorité est la même, le modèle s'imbrique uniquement au niveau du corps du modèle extérieur. Par exemple, le bloc `<script>...</script>` peut uniquement s'imbriquer dans le bloc `<html>...</html>` contenant des balises légales ; il ne peut donc pas s'imbriquer au sein d'une balise, d'un attribut, d'une chaîne, d'un commentaire, etc.

   Les modèles dont la priorité est supérieure au modèle extérieur peuvent s'imbriquer à quasiment tous les niveaux du modèle extérieur. Par exemple, le bloc `<%...%>` peut non seulement s'imbriquer au niveau du corps du bloc `<html>...</html>`, mais également au sein d'une balise, d'un attribut, d'une chaîne, d'un commentaire, etc.

   Le niveau d'imbrication maximum est de 4.
3. Lorsqu'il recherche des chaînes `blockStart`, Dreamweaver utilise toujours la correspondance la plus longue.
4. Une fois la chaîne `blockEnd` atteinte pour le modèle actuel, la coloration de syntaxe revient au niveau où la chaîne `blockStart` est détectée. Par exemple, si un bloc `<%...%>` est détecté au sein d'une chaîne HTML, la coloration reprend avec la couleur de la chaîne HTML.

Modification des modèles

Vous pouvez modifier les styles d'un modèle de coloration du code soit en modifiant le fichier de coloration du code ou en sélectionnant la catégorie Coloration du code dans la boîte de dialogue Préférences de Dreamweaver (voir illustration ci-dessous) :

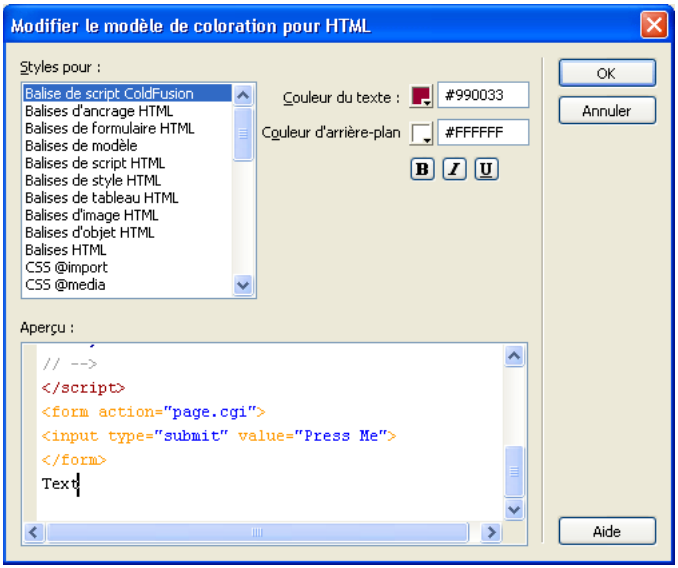


Les paramètres de couleur et de style des champs susceptibles d'apparaître plusieurs fois (par exemple, stringStart) doivent être indiqués uniquement pour la première balise. Si vous répartissez les paramètres de couleur et de style sur plusieurs balises et que vous les modifiez ultérieurement dans la boîte de dialogue Préférences, vous perdrez vos données.

REMARQUE

Macromedia recommande de créer des copies de sauvegarde de tous les fichiers XML avant d'apporter vos modifications. Vérifiez toutes les modifications apportées manuellement avant de modifier les paramètres de style et de couleur dans la boîte de dialogue Préférences. Si vous modifiez un fichier XML non valide dans la boîte de dialogue Préférences, vous perdrez vos données.

Pour modifier les styles d'un modèle dans la catégorie Coloration du code de la boîte de dialogue Préférences, double-cliquez sur un type de document ou cliquez sur le bouton Modifier le modèle de coloration ; la boîte de dialogue Modifier le modèle de coloration s'affiche.



Pour modifier le style d'un élément en particulier, sélectionnez-le dans la liste Styles pour. Le volet Styles pour répertorie les champs du modèle modifié ainsi que les modèles susceptibles d'apparaître sous forme de bloc au sein de ce modèle. Par exemple, si vous modifiez le modèle HTML, les champs des blocs CSS et JavaScript apparaissent également.

Les champs de modèle répertoriés correspondent aux champs définis dans le fichier XML. La valeur de l'attribut `scheme.name` précède chacun des champs répertoriés dans le volet Styles pour. Les champs sans nom ne sont pas répertoriés.

Outre la coloration du code, le style d'un élément désigne la mise en caractère gras, la mise en italique, le soulignement et la couleur d'arrière-plan. Une fois l'élément sélectionné dans le volet Styles pour, vous pouvez modifier toutes ces caractéristiques de style.

La zone Aperçu affiche un échantillon de texte auquel s'appliquent les paramètres indiqués. L'échantillon est défini dans le paramètre `sampleText` du modèle.

Sélectionnez un élément dans la zone Aperçu pour modifier la sélection de la liste Style pour.

Si vous modifiez les paramètres d'un élément d'un modèle, Dreamweaver consigne la valeur dans le fichier de coloration et remplace le paramètre d'origine. Lorsque vous cliquez sur OK, Dreamweaver actualise automatiquement toutes les modifications de coloration du code.

Exemples de coloration du code

Les exemples de coloration du code ci-dessous illustrent les modèles de coloration du code d'un document de style en cascade et d'un document JavaScript. Par souci de concision, les listes de mots-clés de l'exemple JavaScript sont abrégées.

Coloration du code CSS

```
  <scheme name="CSS" id="CSS" doctypes="CSS" priority="50">
    <ignoreCase>Yes</ignoreCase>
    <ignoreTags>Yes</ignoreTags>
    <blockStart doctypes="ASP-JS,ASP-
  VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,CFC,HTML,JSP,LibraryItem,DWTempl
  ate,PHP_MySQL" scheme="outerTag"><![CDATA[<style>]]></blockStart>
    <blockEnd><![CDATA[</style>]]></blockEnd>
    <blockStart doctypes="ASP-JS,ASP-
  VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,CFC,HTML,JSP,LibraryItem,DWTempl
  ate,PHP_MySQL" scheme="outerTag"><![CDATA[<style\s+\*>]]></blockStart>
    <blockEnd><![CDATA[</style>]]></blockEnd>
    <commentStart name="Comment" id="CodeColor_CSSComment"><![CDATA[/
  *]]></commentStart>
    <commentEnd><![CDATA[*/]]></commentEnd>
    <endOfLineComment><![CDATA[<!--]]></endOfLineComment>
    <endOfLineComment><![CDATA[-->]]></endOfLineComment>
    <stringStart name="String" id="CodeColor_CSSString"><![CDATA["]]></
  stringStart>
    <stringEnd><![CDATA["]]></stringEnd>
    <stringStart><![CDATA[']]></stringStart>
    <stringEnd><![CDATA[']]></stringEnd>
    <stringEsc><![CDATA[\]]></stringEsc>
    <cssSelector name="Selector" id="CodeColor_CSSSelector" />
    <cssProperty name="Property" id="CodeColor_CSSProperty" />
    <cssValue name="Value" id="CodeColor_CSSValue" />
    <sampleText doctypes="CSS"><![CDATA[/* Comment */
H2, .head2      {
            font-family : 'Sans-Serif';
            font-weight : bold;
            color : #339999;
      }]]>
  </sampleText>
  </scheme>
```

Echantillon CSS

L'échantillon du modèle CSS suivant illustre le modèle de coloration du code CSS :

```
/* Comment */
H2, .head2      {
            font-family : 'Sans-Serif';
            font-weight : bold;
            color : #339999;

   }
```

Les lignes suivantes, extraites du fichier Colors.xml, fournissent les valeurs de couleur et de style affichées dans l'échantillon et ont été assignées par le modèle de coloration du code :

```
<syntaxColor id="CodeColor_CSSSelector" text="#FF00FF" />
<syntaxColor id="CodeColor_CSSProperty" text="#000099" />
<syntaxColor id="CodeColor_CSSValue" text="#0000FF" />
```

Coloration du code JavaScript

```
<scheme name="JavaScript" id="JavaScript" doctypes="JavaScript"
  priority="50">
    <ignoreCase>No</ignoreCase>
    <ignoreTags>Yes</ignoreTags>
    <blockStart doctypes="ASP-JS,ASP-
  VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,CFC,HTML,JSP,LibraryItem,DWTempl
  ate,PHP_MySQL" scheme="outerTag"><![CDATA[<script>]]></blockStart>
    <blockEnd><![CDATA[</script>]]></blockEnd>
    <blockStart doctypes="ASP-JS,ASP-
  VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,CFC,HTML,JSP,LibraryItem,DWTempl
  ate,PHP_MySQL" scheme="outerTag"><![CDATA[<script\s+\*>]]></blockStart>
    <blockEnd><![CDATA[</script>]]></blockEnd>
    <commentStart name="Comment"
  id="CodeColor_JavascriptComment"><![CDATA[/*]]></commentStart>
    <commentEnd><![CDATA[*/]]></commentEnd>
    <endOfLineComment><![CDATA[//]]></endOfLineComment>
    <endOfLineComment><![CDATA[<!--]]></endOfLineComment>
    <endOfLineComment><![CDATA[-->]]></endOfLineComment>
    <stringStart name="String"
  id="CodeColor_JavascriptString"><![CDATA["]]></stringStart>
    <stringEnd><![CDATA["]]></stringEnd>
    <stringStart><![CDATA[']]></stringStart>
    <stringEnd><![CDATA[']]></stringEnd>
    <stringEsc><![CDATA[\]]></stringEsc>
    <brackets name="Bracket"
  id="CodeColor_JavascriptBracket"><![CDATA[{[()]}]]></brackets>
    <operators name="Operator"
  id="CodeColor_JavascriptOperator"><![CDATA[+-*/%<>!?:=&|^]]></operators>
    <numbers name="Number" id="CodeColor_JavascriptNumber" />
    <regexp name="RegExp" id="CodeColor_JavascriptRegexp" delimiter="/"
  escape="\\">
      <searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
```

```
      <searchPattern><![CDATA[=\s*/\e*\\/]]></searchPattern>
    </regexp>
    <idChar1>_$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</
  idChar1>
    <idCharRest name="Identifier"
  id="CodeColor_JavascriptIdentifier">_$abcdefghijklmnopqrstuvwxyzABCDEFGH
  IJKLMNOPQRSTUVWXYZ0123456789</idCharRest>
    <functionKeyword name="Function Keyword"
  id="CodeColor_JavascriptFunction">function</functionKeyword>
    <keywords name="Reserved Keywords" id="CodeColor_JavascriptReserved">
      <keyword>break</keyword>
. . .
    </keywords>
    <keywords name="Native Keywords" id="CodeColor_JavascriptNative">
      <keyword>abs</keyword>
. . .
    </keywords>
    <keywords id="CodeColor_JavascriptNumber">
      <keyword>Infinity</keyword>
      <keyword>Nan</keyword>
    </keywords>
    <keywords name="Client Keywords" id="CodeColor_JavascriptClient">
      <keyword>alert</keyword>
. . .
    </keywords>
    <sampleText><![CDATA[/* JavaScript */
function displayWords(arrayWords) {
  for (i=0; i < arrayWords.length(); i++) {
    // commentaire en ligne
    alert("Word " + i + " is " + arrayWords[i]);
  }
}

var tokens = new Array("Hello", "world");
displayWords(tokens);
]]></sampleText>
  </scheme>
```

Echantillon JavaScript

L'échantillon du modèle JavaScript suivant illustre le modèle de coloration du code JavaScript :

```
* JavaScript */
function displayWords(arrayWords) {
  for (i=0; i < arrayWords.length(); i++) {
    // commentaire en ligne
    alert("Word " + i + " is " + arrayWords[i]);
  }
}
```

```
var tokens = new Array("Hello", "world");
displayWords(tokens);
```

Les lignes suivantes, extraites du fichier Colors.xml, fournissent les valeurs de couleur et de style affichées dans l'échantillon et ont été assignées par le modèle de coloration du code :

```
<syntaxColor id="CodeColor_JavascriptComment" text="#999999" italic="true"
  />
<syntaxColor id="CodeColor_JavascriptFunction" text="#000000" bold="true" /
  >
<syntaxColor id="CodeColor_JavascriptBracket" text="#000099" bold="true" />
<syntaxColor id="CodeColor_JavascriptNumber" text="#FF0000" />
<syntaxColor id="CodeColor_JavascriptClient" text="#990099" />
<syntaxColor id="CodeColor_JavascriptNative" text="#009999" />
```

Validation du code

Lorsque vous ouvrez un document en mode Code, Dreamweaver vérifie automatiquement la validité des balises, attributs, propriétés ou valeurs CSS en fonction des navigateurs cibles spécifiés par l'utilisateur. Dreamweaver souligne les erreurs détectées par une ligne ondulée de couleur rouge.

Les profils des navigateurs sont stockés dans le dossier Browser Profile du dossier Configuration de Dreamweaver. Chaque profil de navigateur se présente sous la forme d'un fichier texte portant le nom du navigateur. Par exemple, le profil du navigateur Internet Explorer version 6.0 correspond au fichier Internet_Explorer_6.0.txt. Pour prendre en charge la vérification des navigateurs cibles pour CSS, Dreamweaver stocke les informations de profil CSS d'un navigateur dans un fichier XML dont le nom correspond au profil du navigateur, auquel a été ajouté le suffixe _CSS.xml. Par exemple, le profil CSS du navigateur Internet Explorer 6.0 se trouve dans le fichier Internet_Explorer_6.0_CSS.xml. Si vous souhaitez ignorer une erreur signalée par Dreamweaver, vous pouvez modifier le fichier de profil CSS.

Le fichier de profil CSS comprend trois balises XML : `css-support`, `property` et `value`. Les sections ci-après fournissent une description de chacune de ces balises.

<css-support>

Description

Cette balise correspond au nœud racine d'un ensemble de balises `property` et `value` prises en charge par un navigateur donné.

Attributs

Néant

Contenu

Balises `property` et `value`.

Contenant

Néant

Exemple

```
<css-support>
...
</css-support>
```

<property>

Description

Définit une propriété CSS prise en charge par un profil de navigateur.

Attributs

`name, names, supportlevel, message`

- `name="`*`nom_propriété`*`"` Nom de la propriété à prendre en charge.
- `names="`*`nom_propriété`*`, `*`nom_propriété, ...`*`"` Liste des noms de propriété à prendre en charge. Les différents noms sont séparés par des virgules.

  L'attribut `names` peut être considéré comme une forme abrégée. Par exemple, l'attribut `names` ci-dessous correspond à une méthode abrégée de définition de l'attribut `name` qui suit :

  ```
  <property names="foo,bar">
    <value type="named" name="top"/>
    <value type="named" name="bottom"/>
  </property>

  <property name="foo">
    <value type="named" name="top"/>
    <value type="named" name="bottom"/>
  </property>
  <property name="bar">
    <value type="named" name="top"/>
    <value type="named" name="bottom"/>
  </property>
  ```

- `supportlevel="error", "warning", "info"` ou `"supported"` Indique le niveau de prise en charge de la propriété. Si aucune valeur n'est indiquée, la valeur attribuée est `"supported"`. Si vous indiquez un niveau de prise en charge différent de `"supported"` et que vous ne spécifiez pas l'attribut message, Dreamweaver utilise le message par défaut, « La propriété CSS *nom_propriété* n'est pas prise en charge. »
- `message="chaîne_message"` L'attribut `message` définit la chaîne du message affiché dans Dreamweaver lorsqu'il détecte la propriété d'un document. La chaîne du message décrit les limites ou les contraintes susceptibles d'apparaître en fonction de la valeur de la propriété.

Contenu

`value`

Contenant

`css-support`

Exemple

<property name="background-color" supportLevel="supported">

‹value›

Description

Définit une liste de valeurs prises en charge par la propriété actuelle.

Attributs

`type, name, names, supportlevel, message,`

- `type="any", "named", "units", "color", "string"` ou `"function"` Indique le type de valeur. Si vous spécifiez `"named", "units"` ou `"color"`, l'attribut `name` ou `names` doit spécifier les identifiants de valeur correspondant à l'élément. La valeur `"units"` correspond à une valeur numérique, suivie de l'une des valeurs d'unités spécifiées dans l'attribut name.
- `name="nom_valeur"` Identificateur de valeur CSS. Les espaces et la ponctuation, à l'exception du tiret (-), sont interdits. Nom de l'une des valeurs valides pour la propriété CSS nommée dans le nœud de propriété parent. Ce nom peut désigner une valeur spécifique ou un spécificateur d'unité.
- `names="nom1, nom2, . . ."` Spécifie une liste d'`ID` de valeur séparés par des virgules.
- `supportlevel="error", "warning", "info"` ou `"supported"` Indique le niveau de prise en charge de cette valeur dans le navigateur. Si aucune valeur n'est indiquée, la valeur attribuée est `"supported"`.

- `message="`*`chaîne_message`*`"` L'attribut `message` définit la chaîne du message affiché dans Dreamweaver lorsqu'il détecte la valeur de propriété d'un document. Si vous ne spécifiez pas l'attribut `message`, Dreamweaver affiche la chaîne de message suivante : « *`nom_valeur`* n'est pas prise en charge ».

Contenu

Néant

Contenant

`property`

Exemple

```
<property name="margin">
  <value type="units" name="ex" supportLevel="warning"
    message="The implementation of ex units is buggy in Safari 1.0."/>
  <value type="units" names="%,em,px,in,cm,mm,pt,pc”/>
  <value type="named" name="auto"/>
  <value type="named" name="inherit"/>
</property>
```

Modification du formatage HTML par défaut

La catégorie Format de code de la boîte de dialogue Préférences permet de modifier les préférences générales de formatage du code. L'éditeur de la bibliothèque de balises (Edition > Bibliothèques de balises) permet de modifier le format de balises ou d'attributs spécifiques. Pour plus d'informations, voir *Utilisation de Dreamweaver* dans le menu Aide de Dreamweaver.

Vous pouvez également modifier le formatage d'une balise en modifiant le fichier VTM correspondant à cette balise (dans un sous-dossier du dossier de configuration Tag Libraries) ; il est toutefois préférable de modifier le formatage directement dans l'application Dreamweaver.

Si vous ajoutez ou que vous supprimez un fichier VTM, vous devez modifier le fichier TagLibraries.vtm ; Dreamweaver ignore les fichiers VTM non répertoriés dans le fichier TagLibraries.vtm.

REMARQUE	Modifiez ce fichier dans un éditeur de texte et non dans Dreamweaver.

PARTIE 2

Présentation de Extension de Dreamweaver

2

Maîtrisez les fondements de l'interface Macromedia Dreamweaver 8 et apprenez à étendre Dreamweaver selon vos besoins de développement Web. Ces fondements comprennent les dossiers Dreamweaver, les API d'extension, les composants d'interface de Dreamweaver, le modèle d'objet de document (DOM) et les types de documents de Dreamweaver.

CHAPITRE 3

Extension de Dreamweaver

3

Vous créez en règle générale une extension Dreamweaver pour exécuter l'un des types de tâche suivants :

- automatisation des modifications apportées au document actif de l'utilisateur, telles que l'insertion de code HTML, CFML ou JavaScript, la modification des propriétés du texte et des images ou le tri des tableaux ;
- interaction avec l'application pour l'ouverture ou la fermeture automatique des fenêtres ou des documents, le changement des raccourcis clavier, etc. ;
- connexion aux sources de données, permettant aux utilisateurs de Dreamweaver de créer dynamiquement des pages adaptées aux données ;
- insertion et gestion de blocs de code de serveur dans le document actif.

Il peut se révéler judicieux de rédiger une extension pour le traitement d'une tâche fréquente et donc répétitive. Il se peut également qu'un problème donné ne puisse être résolu qu'en rédigeant une extension spécifique à cette situation. Quoi qu'il en soit, l'ensemble complet d'outils proposés par Dreamweaver vous permettra d'étendre ou de personnaliser ses fonctionnalités.

Pour créer une extension Dreamweaver, exécutez la procédure décrite à la section *Création d'une extension*, page 10.

Les fonctions suivantes de Macromedia Dreamweaver 8 vous permettent de créer des extensions :

- Un analyseur HTML (également appelé *module de rendu*), qui permet de concevoir des interfaces utilisateur pour les extensions en utilisant des champs de formulaires, des calques, des images et d'autres éléments HTML. Dreamweaver dispose de son propre analyseur HTML.
- Une arborescence de dossiers qui organise et stocke les fichiers qui implémentent et configurent les éléments et extensions de Dreamweaver.
- Une série d'API (interfaces de programmation d'applications), qui donnent accès à la fonctionnalité de Dreamweaver au moyen du langage JavaScript.

- Un interpréteur JavaScript, qui exécute les instructions de code JavaScript dans les fichiers d'extension. Dreamweaver utilise l'interpréteur Netscape JavaScript version 1.5. Pour plus d'informations sur les différences qui existent entre cette version de l'interpréteur et les précédentes, voir *Traitement de JavaScript dans les extensions par Dreamweaver*, page 113.

Types d'extensions Dreamweaver

La liste ci-dessous décrit les types d'extensions Dreamweaver présentés dans ce guide.

Les extensions d'**objet de la barre Insérer** sont responsables des modifications dans la barre Insérer. Un objet sert généralement à automatiser l'insertion de code dans un document. Il peut également contenir un formulaire qui regroupe les données fournies par les utilisateurs et un code JavaScript qui traite ces données. Les fichiers d'objet résident dans le dossier Configuration/Objects.

Les extensions de **commande** peuvent se charger de la plupart des tâches, avec ou sans l'intervention de l'utilisateur. Les fichiers de commandes sont généralement appelés à partir du menu Commandes ; ils peuvent l'être également depuis d'autres extensions. Les fichiers de commandes résident dans le dossier Configuration/Commands.

Les extensions de **commande de menu** développent l'API de commandes pour effectuer des tâches relatives à l'appel d'une commande depuis un menu. Les API des commandes de menu vous permettent également de créer un sous-menu dynamique.

Les extensions de **barre d'outils** permettent d'ajouter des éléments aux barres d'outils existantes ou de créer de nouvelles barres d'outils dans l'interface utilisateur de Dreamweaver. Les nouvelles barres d'outils s'affichent sous la barre d'outils par défaut. Les fichiers de barre d'outils résident dans le dossier Configuration/Toolbars.

Les extensions de **rapport** peuvent ajouter des rapports personnalisés sur le site ou modifier le jeu de rapports prédéfinis fournis avec Dreamweaver. Vous pouvez également utiliser l'API de fenêtre Résultats pour créer un rapport autonome.

Les extensions de **bibliothèque et d'éditeur de balises** fonctionnent avec les fichiers de bibliothèque de balises associés. Les extensions de bibliothèque et d'éditeur de balises permettent de modifier les attributs des boîtes de dialogue de balises existantes, de créer de nouvelles boîtes de dialogue de balises et d'ajouter des balises à la bibliothèque de balises. Les fichiers d'extension d'éditeur et de bibliothèques de balises résident dans le dossier Configuration/TagLibraries.

Les extensions d'**inspecteur de propriétés** apparaissent dans le panneau d'inspecteur Propriétés. La plupart des inspecteurs de Dreamweaver relèvent du code principal du produit et ne sont donc pas modifiables ; en revanche, les fichiers d'inspecteurs de propriétés personnalisés peuvent remplacer les interfaces d'inspecteurs de propriétés intégrées de Dreamweaver ou en créer afin d'inspecter les balises personnalisées. Les fichiers d'inspecteur résident dans le dossier Configuration/Inspectors.

Les extensions de **panneau flottant** permettent d'insérer des panneaux flottants dans l'interface utilisateur de Dreamweaver. Les panneaux flottants peuvent interagir avec la sélection, le document ou l'action en cours. Ils peuvent également afficher des informations utiles. Les fichiers de panneau résident dans le dossier Configuration/Floaters.

Les extensions de **comportement** permettent aux utilisateurs d'ajouter du code JavaScript dans leurs documents. Le code JavaScript exécute une tâche spécifique en réponse à un événement lorsque le document est affiché dans un navigateur. Les extensions de comportement s'affichent dans le menu plus (+) du panneau Comportements de Dreamweaver. Les fichiers de comportement résident dans le dossier Configuration/Behaviors/Actions.

Les extensions de **comportement de serveur** permettent d'ajouter des blocs de code côté serveur (ASP, JSP ou ColdFusion) au document. Le code côté serveur exécute des tâches sur le serveur lorsque le document est affiché dans un navigateur. Les extensions de comportement de serveur s'affichent dans le menu plus (+) du panneau Comportements de serveur de Dreamweaver. Les fichiers de comportement de serveur résident dans le dossier Configuration/Server Behaviors.

Les extensions de **source de données** vous permettent d'établir une connexion avec des données dynamiques stockées dans une base de données. Les extensions de source de données apparaissent dans le menu plus (+) du panneau Liaisons. Les extensions de source de données résident dans le dossier Configuration/DataSources.

Les extensions de **format de serveur** vous permettent de définir la mise en forme des données dynamiques.

Les extensions de **composant** vous permettent d'ajouter de nouveaux types de composants au panneau Composants. Composants est le terme utilisé dans Dreamweaver pour définir une des stratégies les plus récentes et populaires d'encapsulation, y compris les services Web, JavaBeans, et les composants ColdFusion (CFC).

Les extensions de **modèle de serveur** permettent de prendre en charge de nouveaux modèles de serveurs. Dreamweaver prend en charge les modèles de serveurs les plus courants (ASP, JSP, ColdFusion, PHP et ASP.NET). Les extensions de modèle de serveur sont nécessaires uniquement pour les solutions de serveur personnalisées, des langues différentes ou un serveur personnalisé. Les fichiers de modèle de serveur résident dans le dossier Configuration/ServerModels.

Les extensions de **traducteur de données** convertissent le code non HTML en code HTML qui s'affiche ensuite dans le mode Création de la fenêtre de document. Ces extensions bloquent également le code non HTML pour empêcher toute analyse par Dreamweaver. Les fichiers de traducteur résident dans le dossier Configuration/Translators.

Autres méthodes d'extension pour Dreamweaver

Vous pouvez également étendre les éléments suivants de Dreamweaver pour développer ses capacités ou l'adapter à votre utilisation.

Les **types de documents** définissent le fonctionnement de Dreamweaver avec différents modèles de serveur. Les informations relatives aux types de documents pour les modèles de serveurs résident dans le dossier Configuration/DocumentTypes. Pour plus d'informations, consultez la section *Types de documents extensibles dans Dreamweaver*, page 37.

Les **fragments de code** sont des blocs de code réutilisables qui sont stockés sous forme de fichiers CSN dans le dossier Configuration/Snippets de Dreamweaver. Ils sont accessibles dans le panneau Fragments de code. Vous pouvez créer de nouveaux fichiers de fragment de code et les installer dans le dossier Snippets afin qu'ils soient disponibles.

Les **indicateurs de code** sont des menus qui vous évitent de saisir tout le texte en proposant une liste de chaînes susceptibles de compléter la chaîne que vous tapez. Si une des chaînes du menu correspond à celle que vous avez commencé à entrer, vous pouvez la sélectionner pour l'insérer à la place de la chaîne dont vous aviez commencé la saisie. Des menus d'indicateurs de code sont définis dans le fichier codehints.xml dans le dossier Configuration/CodeHints. Vous pouvez en ajouter d'autres pour de nouvelles balises et fonctions que vous avez définies.

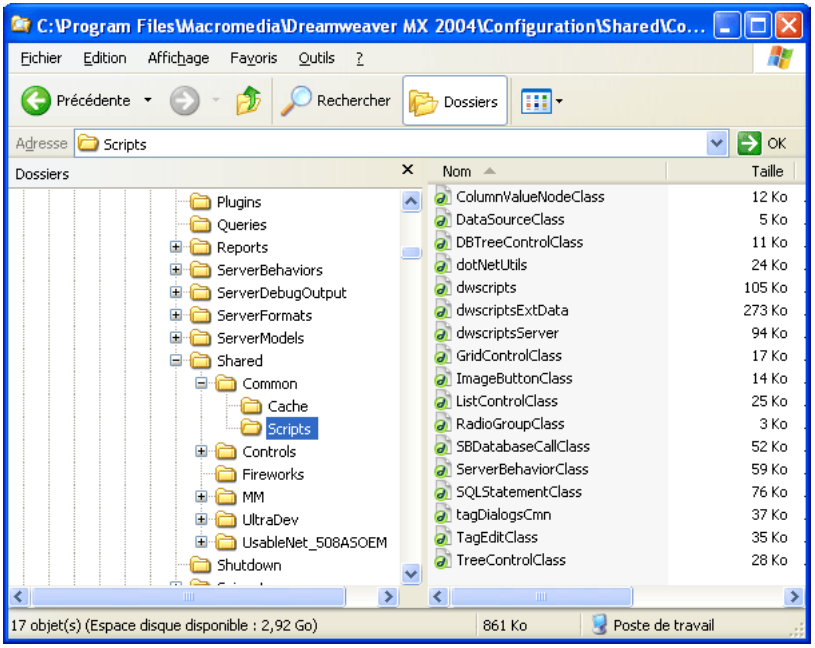
Les **menus** sont définis dans le fichier menus.xml du dossier Configuration/Menus. Vous pouvez ajouter de nouveaux menus Dreamweaver pour vos extensions en ajoutant leurs balises de menu dans le fichier menus.xml. Pour plus d'informations, consultez la section Chapitre 8, “Menus et commandes de menu,” on page 191.

Extensions et dossiers de configuration

Les dossiers et les fichiers stockés dans le dossier Configuration de Dreamweaver contiennent les extensions livrées avec Dreamweaver. Lorsque vous rédigez une extension, vous devez enregistrer les fichiers dans le dossier approprié afin que Dreamweaver puisse les reconnaître. Ainsi, si vous créez une extension d'inspecteur de propriétés, vous enregistrez les fichiers dans le dossier Configuration/Inspectors. Si vous téléchargez puis installez une extension depuis le site Web de Macromedia Exchange (www.macromedia.com/go/exchange_fr), Extension Manager enregistre automatiquement les fichiers d'extension dans les dossiers appropriés.

Vous pouvez utiliser les fichiers du dossier Configuration de Dreamweaver comme exemple ; notez toutefois que ces fichiers sont généralement plus complexes que l'extension typique disponible sur le site Web de Macromedia Exchange. Pour plus d'informations sur le contenu respectif des sous-dossiers du dossier Configuration, voir le fichier Configuration_ReadMe.htm.

Le dossier Configuration/Shared ne correspond à aucun type d'extension particulier. Il s'agit d'un lieu de stockage central pour les fonctions utilitaires, les classes et les images utilisées par plusieurs extensions. Les fichiers résidant dans le dossier Configuration/Shared/Common sont censés être utiles à un large éventail d'extensions. Ces fichiers constituent à la fois de bons exemples des techniques JavaScript et des utilitaires pratiques. Recherchez en priorité dans ces fichiers les fonctions exécutant des tâches spécifiques, par exemple créer une référence DOM (modèle d'objet de document) valide pour un objet, vérifier si la sélection active se trouve à l'intérieur d'une balise particulière, ignorer les caractères dans les chaînes, etc. Il est préférable de créer un sous-dossier distinct dans le dossier Configuration/Shared/Common, comme indiqué ci-après, lorsque vous créez des fichiers communs afin de les stocker.



Structure de dossier Configuration/Shared/Common/Scripts

Pour plus d'informations sur le dossier partagé, voir Annexe, “Dossier Shared,” on page 505.

Dossiers de configuration multiutilisateur

Pour les systèmes d'exploitation multiutilisateurs Windows XP, Windows 2000 et Macintosh OS X, Dreamweaver crée un dossier de configuration distinct pour chaque utilisateur, en complément du dossier Configuration Dreamweaver. Chaque fois que Dreamweaver ou qu'une extension JavaScript écrit dans le dossier Configuration, Dreamweaver écrit systématiquement dans le dossier de configuration utilisateur. Ceci permet à chaque utilisateur de définir ses paramètres de personnalisation de Dreamweaver sans modifier les paramètres d'autres utilisateurs. Pour plus d'informations, voir *Personnalisation de Dreamweaver dans un environnement multi-utilisateurs*, page 29 et API de configuration multiutilisateur et d'accès aux fichiers dans le *Guide des API de Dreamweaver*.

Exécution des scripts au démarrage ou à la fermeture

Si vous placez un fichier de commandes dans le dossier Configuration/Startup, la commande s'exécute au démarrage de Dreamweaver. Les commandes de démarrage se chargent avant le fichier menus.xml, avant les fichiers du dossier ThirdPartyTags et avant les autres commandes, objets, comportements, inspecteurs, panneaux flottants ou traducteurs. Par conséquent, vous pouvez utiliser les commandes de démarrage pour modifier le fichier menus.xml ou d'autres fichiers d'extension. Vous pouvez également afficher des avertissements, inviter l'utilisateur à donner des informations ou appeler la fonction `dreamweaver.runCommand()`. Toutefois, vous ne pouvez pas appeler une commande qui nécessite un DOM (Document Object Model) valide à partir du dossier Startup. Pour plus d'informations sur le DOM Dreamweaver, voir le Chapitre 5, “Modèle d'objet de document (DOM) Dreamweaver,” on page 133.

De même, si vous placez un fichier de commandes dans le dossier Configuration/Shutdown, la commande s'exécute à l'arrêt de Dreamweaver. A partir des commandes de fermeture, vous pouvez appeler la fonction `dreamweaver.runCommand()`, afficher des avertissements ou inviter l'utilisateur à entrer des informations, mais vous ne pouvez pas arrêter le processus de fermeture.

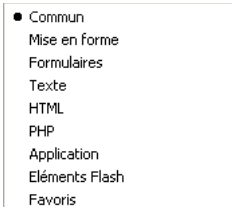
Pour plus d'informations sur les commandes, voir Chapitre 7, “Commandes,” on page 177. Pour plus d'informations sur la fonction `dreamweaver.runCommand()`, voir le *Guide des API de Dreamweaver*.

Rechargez les extensions.

Si vous modifiez une extension pendant que vous utilisez Dreamweaver, vous pouvez la recharger pour permettre à Dreamweaver d'identifier la modification.

Pour recharger les extensions :

1. Appuyez sur la touche Contrôle (Windows) ou Option (Macintosh) tout en cliquant avec la souris sur le menu affichant les catégories dans la barre de titre de la barre Insérer.



2. Sélectionnez Recharger extensions.

REMARQUE

Notez que sur un système d'exploitation multi-utilisateurs, vous devez modifier les copies des fichiers de configuration stockées dans votre dossier Configuration et non les fichiers de configuration principaux. Pour plus d'informations, consultez la section *Extensions et dossiers de configuration*, page 109.

API d'extension

Les API d'extension vous fournissent les fonctions appelées par Dreamweaver pour implémenter chaque type d'extension. Le corps de ces fonctions doit être rédigé conformément aux descriptions de chaque type d'extension. Il faut également spécifier les valeurs renvoyées attendues par Dreamweaver.

Si vous êtes développeur et souhaitez travailler directement en langage de programmation C, vous disposez d'une API d'extensibilité C qui vous permet de créer des DLL (bibliothèques de liens dynamiques). La fonctionnalité fournie dans ces API enveloppe vos DLL C de code JavaScript afin que votre extension puisse s'exécuter en toute transparence dans Dreamweaver.

La documentation des API d'extension souligne ce qu'apporte chaque fonction quand Dreamweaver l'appelle et les valeurs renvoyées attendues par Dreamweaver.

Consultez le *Guide des API de Dreamweaver* pour plus d'informations sur l'API d'utilitaire et l'API JavaScript, qui proposent des fonctions permettant d'effectuer des tâches spécifiques dans vos extensions.

Traitement de JavaScript dans les extensions par Dreamweaver

Dreamweaver vérifie le dossier Configuration/*extension_type* au démarrage. Si Dreamweaver y détecte un fichier d'extension, il traite le code JavaScript selon la procédure suivante :

- Compilation de tous les éléments compris entre les balises d'ouverture et de fermeture `SCRIPT`
- Exécution du code situé dans les balises `SCRIPT` ne faisant pas partie d'une déclaration de fonction

REMARQUE Cette étape s'impose pendant le démarrage dans la mesure où certaines extensions peuvent nécessiter l'initialisation des variables globales.

Dans le cas des fichiers JavaScript externes spécifiés dans les attributs `SRC` des balises `SCRIPT`, Dreamweaver se charge des opérations suivantes :

- lecture du fichier ;
- compilation du code ;
- exécution des procédures.

REMARQUE Si un code JavaScript intégré à votre fichier d'extension contient la chaîne "`</SCRIPT>`", l'interpréteur JavaScript la lit comme une balise `SCRIPT` de fermeture et signale une erreur littérale de chaîne non terminée. Pour éviter ce problème, divisez la chaîne en parties concaténées comme suit : `"<' + '/SCRIPT>"`.

Dreamweaver exécute le code contenu dans le gestionnaire d'événements `onLoad` (s'il est présent dans la balise `BODY`) lorsque l'utilisateur choisit la commande ou l'action dans un menu pour les types d'extensions de commande ou de comportement.

Dreamweaver exécute le code contenu dans le gestionnaire d'événements `onLoad` dans la balise `BODY` si le corps du document contient un formulaire pour les extensions d'objet.

Dreamweaver ignore le gestionnaire `onLoad` de la balise `BODY` dans les extensions suivantes :

- traducteur de données ;
- Inspecteur de propriétés
- Panneau flottant

Pour toutes les extensions, Dreamweaver exécute le code contenu dans d'autres gestionnaires d'événements (par exemple, `onBlur="alert('Ceci est un champ obligatoire.')"`) lorsque l'utilisateur utilise les champs de formulaire auxquels ils sont rattachés.

Dreamweaver prend en charge l'utilisation de gestionnaires d'événement dans des liens. Les gestionnaires d'événements des liens doivent respecter la syntaxe suivante :

Les `<a href=”#” onMouseDown=alert(‘hi’)>link text</a>`

plug-ins (réglés sur `play` en permanence) sont pris en charge dans la balise `BODY` des extensions. En revanche, l'instruction `document.write()`, les applets Java et les commandes ActiveX ne sont pas gérées.

Affichage de l'aide

La fonction `displayHelp()`, qui fait partie de nombreux API d'extension, entraîne les deux actions suivantes de Dreamweaver lorsque vous l'incluez dans votre extension :

- Ajout d'un bouton d'aide à l'interface ;
- Appelle `displayHelp()` lorsque l'utilisateur clique sur le bouton d'aide.

Vous devez rédiger le corps de la fonction `displayHelp()` pour afficher l'aide. La façon dont vous codez la fonction `displayHelp()` détermine l'affichage de l'aide par votre extension. Vous pouvez demander à la fonction `dreamweaver.browseDocument()` d'ouvrir un fichier dans un navigateur ou de définir une méthode personnalisée d'affichage de l'aide, comme l'affichage de messages sur un autre calque dans des fenêtres d'alerte.

L'exemple suivant utilise la fonction `displayHelp()` pour afficher l'aide en appelant `dreamweaver.browseDocument()`:

```
// Dans l'exemple suivant, la fonction displayHelp() ouvre dans un
  navigateur un fichier
// expliquant comment utiliser l'extension.
function displayHelp() {

  var myHelpFile = dw.getConfigurationPath() + "ExtensionsHelp/
  myExtHelp.htm";
  dw.browseDocument(myHelpFile);
}
```

Localisation d'une extension

Utilisez les techniques suivantes pour faciliter la traduction de vos extensions dans d'autres langues.

- Séparez les extensions entre fichiers HTML et JavaScript. Les fichiers HTML peuvent être dupliqués et localisés, alors que les fichiers JavaScript ne peuvent pas être localisés.
- Ne définissez pas de chaînes dans les fichiers JavaScript (vérifiez les alertes et le code de l'interface utilisateur). Vous devez extraire toutes les chaînes localisables en fichiers XML séparés dans le dossier Configuration/Strings de Dreamweaver.

- N'insérez pas de code JavaScript dans les fichiers HTML excepté pour les gestionnaires d'événement nécessaires. Ceci évite d'avoir à réparer plusieurs fois une même erreur pour chaque traduction une fois les fichiers HTML répliqués et traduits dans d'autres langues.

Fichiers de chaîne XML

Conservez toutes les chaînes dans des fichiers XML dans le dossier Configuration/Strings de Dreamweaver. Si vous installez plusieurs fichiers d'extension liés, cela vous permet de partager toutes les chaînes dans un seul fichier XML. Le cas échéant, cela vous permet également de faire référence à une même chaîne depuis des extensions C++ et JavaScript.

Vous pouvez créer un fichier nommé myExtensionStrings.xml. L'exemple suivant montre le format du fichier :

```
<strings>

  <!-- errors for feature X -->
  <string id="featureX/subProblemY" value="There was a with X when you did
  Y. Try not to do Y!"/>
  <string id="featureX/subProblemZ" value="There was another problem with
  X, regarding Z. Don't ever do Z!"/>

</strings>
```

Vos fichiers JavaScript peuvent maintenant se référer à ces chaînes traduisibles en appelant la fonction `dw.loadString()`, ainsi que l'illustre l'exemple suivant :

```
function initializeUI()
{
   ...
   if (problemYhasOccured)
   {
     alert(dw.loadString("featureX/subProblemY");
   }
}
```

Vous pouvez utiliser des barres obliques (/), mais pas d'espaces dans vos identificateurs de chaînes. Les barres obliques vous permettent d'établir une hiérarchie et d'inclure l'ensemble des chaînes dans un fichier XML unique.

REMARQUE Les fichiers commençant par cc dans le dossier Configuration/Strings sont des fichiers Contribute. C'est le cas, par exemple, du fichier ccSiteStrings.xml.

Chaînes localisables avec valeurs intégrées

Certaines chaînes d'affichage comportent des valeurs intégrées. Vous pouvez utiliser la fonction `errMsg()` pour afficher ces chaînes. La fonction `errMsg()`, similaire à la fonction `printf()` dans C, se trouve dans le fichier string.js du dossier Configuration/Shared/MM/Scripts/CMN. Utilisez le signe `%` et la lettre `s` (les caractères de l'espace réservé) pour indiquer l'emplacement où les valeurs doivent apparaître dans la chaîne, puis transmettez la chaîne et les variables (en tant qu'arguments) à `errMsg()`. Exemple :

```
<string id="featureX/fileNotFoundInFolder" value="File %s could not be
  found in folder %s."/>
```

L'exemple suivant indique comment la chaîne, ainsi que toute variable à imbriquer, est transmise à la fonction `alert()`.

```
if (fileMissing)
{
  alert( errMsg(dw.loadString("featureX/fileNotFoundInFolder"),fileName,
    folderName) );
}
```

Utilisation de Extension Manager

Si vous créez des extensions destinées à d'autres utilisateurs, il convient de les conditionner conformément aux indications disponibles sur le site Web de Macromedia Exchange (www.macromedia.com/go/exchange_fr), sous la rubrique Help > How to Create an Extension (Aide, Comment créer une extension). Après avoir rédigé puis testé une extension dans Extension Manager, sélectionnez Fichier > Empaqueter une extension. Une fois l'extension conditionnée, vous pouvez l'envoyer sur Exchange à partir de Extension Manager ; pour cela, sélectionnez Fichier > Envoyer une extension.

Extension Manager est livré avec Dreamweaver. Vous trouverez des détails sur son utilisation dans ses fichiers d'aide ou sur le site Web Macromedia Exchange.

CHAPITRE 4

Interfaces utilisateur destinées aux extensions

4

La plupart des extensions sont conçues pour recevoir des informations de l'utilisateur par le biais d'une interface utilisateur (IU). Ainsi, si vous créez une extension d'inspecteur de propriétés associée à la balise `marquee`, vous devez permettre à l'utilisateur de spécifier des attributs tels que la direction et la hauteur. Si vous envisagez de demander la certification Macromedia pour votre extension, veillez à suivre les consignes disponibles dans les fichiers de Extension Manager sur le site Web de Macromedia Exchange (www.macromedia.com/go/exchange_fr). Ces consignes n'ont pas pour objet de limiter votre créativité, mais d'assurer le bon fonctionnement des extensions certifiées dans l'interface utilisateur de Macromedia Dreamweaver 8 et de vérifier que la conception de l'interface utilisateur de l'extension n'entrave pas sa fonctionnalité.

Conception d'une interface utilisateur d'extension

En général, une extension est créée pour effectuer une tâche que l'utilisateur rencontre fréquemment. Certaines parties de la tâche étant répétitives, une extension permet de les automatiser. Plusieurs étapes de cette tâche ou attributs spécifiques du code traité par l'extension peuvent être modifiés. Pour recevoir les entrées utilisateur de ces valeurs variables, vous devez créer une interface utilisateur.

Vous pouvez par exemple créer une extension de mise à jour d'un catalogue de vente sur le Web. Les utilisateurs doivent régulièrement modifier les valeurs des sources d'image, des descriptions d'articles et des prix. Bien que les valeurs évoluent, les procédures d'extraction de ces dernières et de mise en forme des informations à afficher sur le site Web demeurent identiques. Une extension simple peut automatiser la mise en forme, tout en laissant aux utilisateurs le soin d'entrer manuellement les valeurs actualisées des sources d'image, des descriptions d'articles et des prix. Une extension plus complexe peut extraire régulièrement ces valeurs d'une base de données.

L'interface utilisateur de votre extension a pour objet de recevoir les informations entrées par l'utilisateur. Ces informations gèrent les aspects variables d'une tâche répétitive exécutés par l'extension. Dreamweaver prend en charge les éléments de formulaires HTML et JavaScript comme modules de base de la structure des commandes d'interface utilisateur d'extension et affiche l'interface au moyen de son outil de rendu HTML. Ainsi, une interface utilisateur d'extension peut se présenter sous la forme d'un simple fichier HTML contenant un tableau à deux colonnes, composé de textes descriptifs et de champs de saisie de formulaire.

Lorsque vous concevez une extension, vous devez déterminer les variables requises et les éléments de formulaire appropriés à ces dernières.

Veillez à tenir compte des observations suivantes pendant la conception d'une interface utilisateur d'extension :

- Pour attribuer un nom à votre extension, indiquez-le dans la balise `title` de votre fichier HTML. Dreamweaver affiche ce nom dans la barre de titre Extension.
- Alignez à droite les étiquettes de texte dans la partie gauche de l'interface utilisateur, et alignez à gauche les zones de texte dans la partie droite. Cette disposition permet à l'utilisateur de repérer plus facilement le début d'une zone de texte. La zone de texte peut être suivie d'un texte concis, par exemple une explication ou une unité de mesure.
- Alignez à gauche le libellé des cases à cocher et des boutons radio dans la partie droite de l'interface utilisateur.
- Dans le cas de code lisible, attribuez un nom logique aux zones de texte. Si vous créez votre interface utilisateur d'extension à l'aide de Dreamweaver, vous pouvez utiliser l'inspecteur des propriétés ou Quick Tag Editor pour nommer les champs.

Le scénario typique consiste, une fois l'interface utilisateur créée, à vérifier que le code de l'extension exécute correctement les tâches suivantes impliquant l'interface :

- Récupération des valeurs des zones de texte
- Définition des valeurs par défaut des zones de texte ou collecte des valeurs à partir de la sélection
- Application des modifications au document de l'utilisateur

Commande de rendu HTML de Dreamweaver

Jusqu'à la version 4, Dreamweaver restituait plus d'espace autour des commandes de formulaire que Microsoft Internet Explorer et Netscape Navigator. Dreamweaver ayant recours à son moteur de rendu HTML pour afficher les interfaces utilisateur d'extension, les commandes de formulaire de celles-ci sont plus spacieuses.

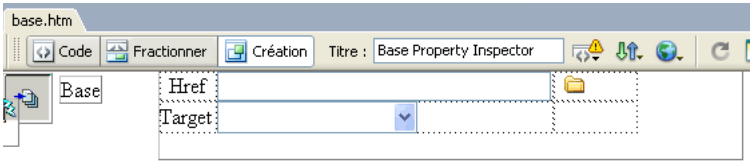
Macromedia a amélioré le rendu des commandes de formulaire afin qu'il corresponde plus précisément à celui des navigateurs. Pour bénéficier des améliorations du procédé de rendu, il convient d'utiliser l'une des trois nouvelles instructions `DOCTYPE` dans les fichiers d'extension, comme indiqué dans l'exemple suivant :

```
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine 5.0//
  dialog">

<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine5.0//
  floater">

<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine5.0//pi">
```

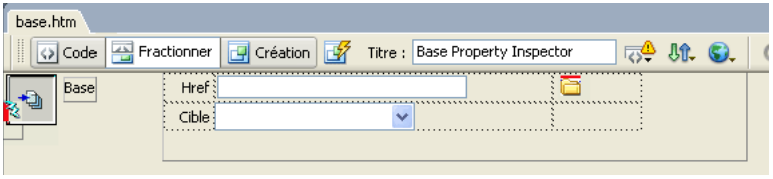
En règle générale, les instructions `DOCTYPE` doivent figurer sur la première ligne d'un document. Néanmoins, afin d'éviter des conflits avec les directives spécifiques aux extensions qui, dans les versions antérieures, devaient se trouver sur la première ligne d'un fichier (par exemple, le commentaire en haut d'un fichier de l'inspecteur des propriétés ou la directive `MENU-LOCATION=NONE` d'une commande), l'ordre des directives et les instructions `DOCTYPE` importent peu, tant qu'elles figurent avant la balise `html` d'ouverture.

Les instructions `DOCTYPE` inédites permettent non seulement de créer des interfaces utilisateur d'extension mieux assorties aux boîtes de dialogue et aux panneaux intégrés, mais également d'afficher vos extensions dans le mode Création de Dreamweaver, de façon à ce que vous puissiez les voir comme le pourrait un utilisateur.

Les exemples suivants montrent l'inspecteur des propriétés de base *sans* l'instruction `DOCTYPE`, qui améliore le rendu des commandes de formulaire, puis *avec* l'instruction `DOCTYPE`.



L'inspecteur des propriétés de base tel qu'il apparaît en mode Création sans l'instruction DOCTYPE.



L'inspecteur des propriétés de base tel qu'il apparaît en mode Création avec l'instruction DOCTYPE (à la suite de quelques réglages pour l'adapter au nouveau rendu).

Utilisation de commandes d'interface utilisateur personnalisées dans les extensions

Outre les éléments de formulaire HTML standard, Dreamweaver prend en charge des commandes personnalisées qui facilitent la création d'interfaces flexibles et professionnelles, comme suit :

- listes de sélection modifiables (également appelées des zones de liste modifiables), pour associer la fonctionnalité d'une liste à celle d'une zone de texte ;
- commandes de base de données, pour simplifier l'affichage des hiérarchies et champs de données ;
- commandes d'arborescence qui organisent les informations en nœuds extensibles et réductibles ;
- commandes de bouton couleur, pour ajouter une interface de sélecteur de couleurs à vos extensions.

Listes de sélection modifiables

Nombre d'interfaces utilisateur d'extension sont constituées de listes déroulantes définies à l'aide de la balise `select`. Dans Dreamweaver, vous pouvez autoriser la modification de ces listes dans les extensions en ajoutant l'instruction `editable="true"` à la balise `select`. Pour définir une valeur par défaut, il convient de définir l'attribut `editText` ainsi que la valeur que vous voulez voir apparaître dans la liste de sélection.

L'exemple suivant illustre les paramètres d'une liste de sélection modifiable :

```
<select name="travelOptions" style="width:250px" editable="true"
   editText="other (please specify)">
<option value="plane">plane</option>
<option value="car">car</option>
<option value="bus">bus</option>
</select>
```

Lorsque vous utilisez des listes de sélection dans vos extensions, vérifiez la présence d'un attribut et, le cas échéant, sa valeur. En l'absence d'une valeur, la liste de sélection renvoie la valeur par défaut `false`, laquelle indique que la liste de sélection n'est pas modifiable.

A l'instar des listes de sélection standard non modifiables, les listes de sélection modifiables contiennent la propriété `selectedIndex` (voir *Objets, propriétés et méthodes du DOM Dreamweaver*, page 135), laquelle renvoie la valeur -1 lorsque la zone de texte est sélectionnée.

Pour lire la valeur d'une zone de texte modifiable active dans une extension, il suffit de lire la valeur de la propriété `editText`. La propriété `editText` renvoie la chaîne saisie par l'utilisateur dans la zone de texte modifiable, la valeur de l'attribut `editText` ou une chaîne vide si aucun texte n'a été entré et aucune valeur n'a été spécifiée pour la propriété `editText`.

Dreamweaver ajoute les attributs personnalisés suivants à la balise `select` afin de contrôler les listes déroulantes modifiables :

Nom d'attribut	Description	Valeurs admises
`editable`	Déclare que la liste déroulante contient une zone de texte modifiable.	Une valeur booléenne `true` ou `false`
`editText`	Conserve ou définit le texte dans une zone de texte modifiable.	Chaîne d'une valeur quelconque

REMARQUE Des listes de sélection modifiables sont disponibles dans Dreamweaver.

L'exemple suivant crée une extension de commande contenant une liste de sélection modifiable à l'aide de fonctions JavaScript standard :

Procédez comme suit pour créer l'exemple d'extension :

1. Créez un fichier vide dans un éditeur de texte.
2. Entrez le code suivant :

```
<html>
<head>
  <title>Editable Dropdown Test</title>
  <script language="javascript">
  function getAlert()
  {

    var i=document.myForm.mySelect.selectedIndex;
    if (i>=0)
    {
      alert("Selected index: " + i + "\n" + "Selected text " +
document.myForm.mySelect.options[i].text);
    }
    else
    {
      alert("Nothing is selected" + "\n" + "or you entered a value");
    }
  }
  function commandButtons()
```

```
    {
      return new Array("OK", "getAlert()", "Cancel", "window.close()");
    }
    </script>
  </head>

  <body>
  <div name="test">
  <form name="myForm">
  <table>
    <tr>
      <td colspan="2">
      <h4>Select your favorite</h4>
      </td>
    </tr>
    <tr>
      <td>Sport:</td>
      <td>
      <select name="mySelect" editable="true" style="width:150px"
        editText="Editable Text">
        <option> Baseball</option>
        <option> Football </option>
        <option> Soccer </option>
      </select>
      </td>
    </tr>
  </table>
  </form>
  </div>
  </body>
  </html>
```

3. Enregistrez le fichier sous le nom EditableSelectTest.htm dans le dossier Dreamweaver Configuration/Commands.

Procédez comme suit pour tester l'exemple d'extension :

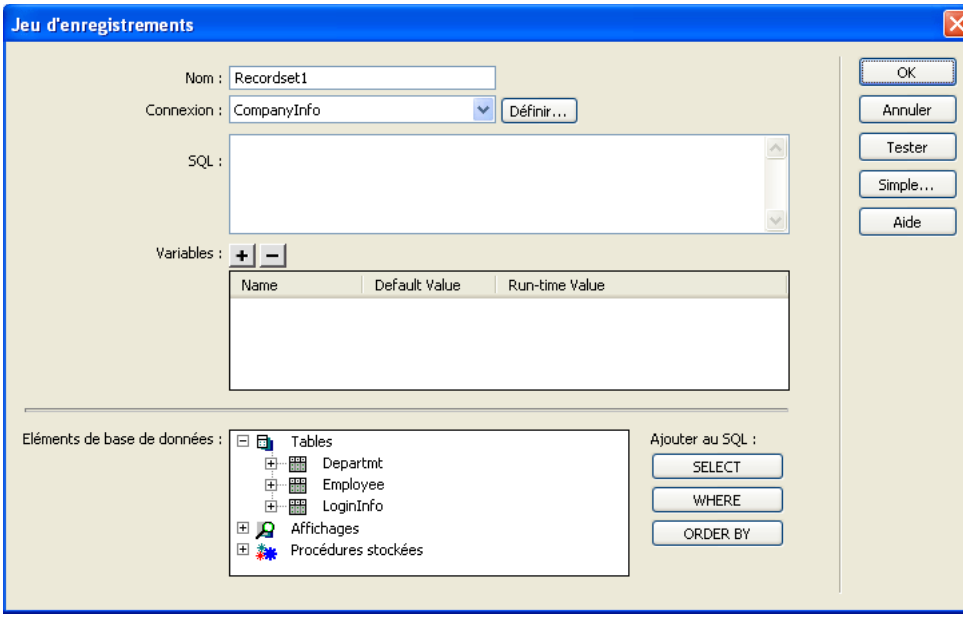
1. Redémarrez Dreamweaver.
2. Sélectionnez Commandes > EditableSelectTest.

   Lorsque vous sélectionnez une valeur dans la liste, un message d'avertissement affiche l'index de la valeur et le texte. Si vous entrez une valeur, un message d'avertissement indique qu'aucun élément n'est sélectionné.

Commandes de base de données

Dreamweaver vous permet d'étendre la balise HTML `select` afin de créer une commande d'arborescence de base de données. Vous pouvez également ajouter une commande de grille de variables. La commande d'arborescence de base de données affiche le schéma des bases de données et la commande de grille de variables affiche les informations tabulaires.

La figure suivante présente une boîte de dialogue Jeu d'enregistrements avancée utilisant une commande d'arborescence de base de données et une commande de grille de variables :



Ajout d'une commande d'arborescence de base de données

La commande d'arborescence de base de données possède les attributs suivants :

Nom d'attribut	Description
`name`	Nom de la commande d'arborescence de base de données
`control.style`	Largeur et hauteur mesurées en pixels.
`type`	Type de commande.
`connection`	Nom de la connexion à la base de données définie dans le Gestionnaire de connexions ; si aucun nom n'est spécifié, la commande est vide.

Nom d'attribut	Description
noexpandbuttons	Lorsque cet attribut est spécifié, la commande d'arborescence ne dessine ni les indicateurs de développement (plus, +) ou de réduction (moins, -), ni les flèches correspondantes sur un ordinateur Macintosh. Cet attribut est très utile pour dessiner des commandes de liste à plusieurs colonnes.
showheaders	Lorsque cet attribut est défini, la commande d'arborescence affiche un en-tête en haut, recensant le nom de chaque colonne.

Toutes les balises d'options placées dans la balise `select` sont ignorées.

Pour ajouter une commande d'arborescence à une boîte de dialogue, vous pouvez utiliser l'échantillon de code suivant en effectuant les substitutions appropriées pour les variables citées :

```
<select name="DBTree" style="width:400px;height:110px" ¬
type="mmdatabasetree" connection="connectionName" noexpandbuttons
  showHeaders></select>
```

La modification de l'attribut `connection` vous permet de récupérer les données sélectionnées et de les afficher dans l'arborescence. Utilisez l'attribut `DBTreeControl` comme objet JavaScript d'enveloppement de la nouvelle balise. Pour obtenir d'autres exemples, consultez le fichier DBTreeControlClass.js du dossier Configuration/Shared/Common/Scripts.

Ajout d'une commande de grille de variables

La commande de grille de variables possède les attributs suivants :

Nom d'attribut	Description
name	Nom de la commande de grille de variables.
style	Largeur de la colonne, exprimée en pixels.
type	Type de commande.
columns	Chaque colonne doit disposer d'un nom, séparée par une virgule.
columnWidth	Largeur de chaque colonne, séparée par une virgule. Les colonnes sont de largeur égale si aucune largeur n'est spécifiée.

L'exemple suivant ajoute une commande de grille de variables à une boîte de dialogue :

```
<select name="ParamList" style="width:515px;" ¬
type="mmparameterlist columns"="Name,SQL Data ¬
Type,Direction,Default Value,Run-time Value" size=6></select>
```

L'exemple suivant crée une commande de grille de variables de 500 pixels de large et comportant cinq colonnes de largeur variable :

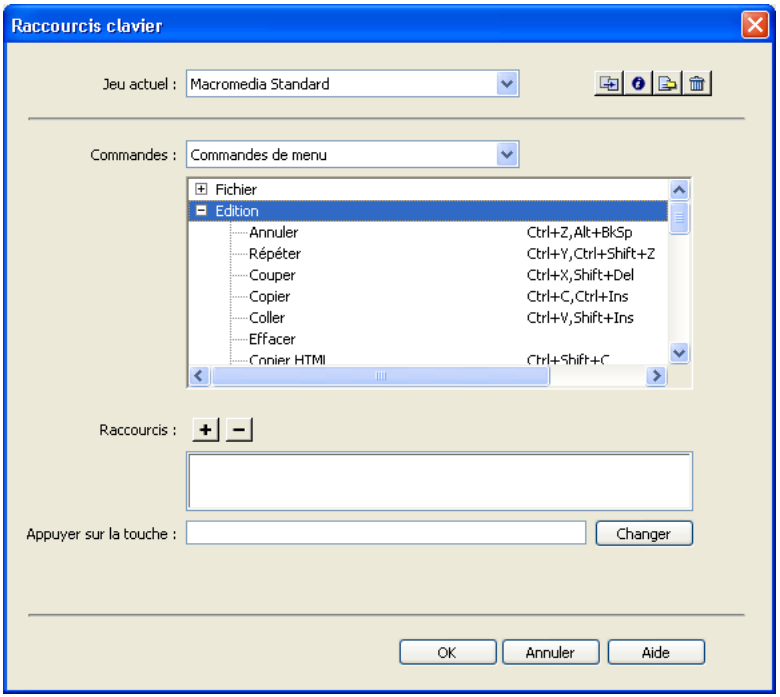
```
<select
  name="ParamList"
  style="width:800px;"
  type="mmparameterlist"
  columns="Name,SQL Data Type,Direction, Default Value,Run-time Value"
  columnWidth="100,25,11,"
  size=6>
```

Cet exemple crée deux colonnes vierges de 182 pixels de large chacune. Le calcul est le suivant : le total des colonnes définies est de 136. La largeur totale de la commande de grille de variables est de 500. L'espace restant une fois les trois premières colonnes insérées est 364. Il reste deux colonnes à insérer : 364 divisé par 2 est égal à 182.

Cette commande de grille de variables comporte également un objet d'enveloppement JavaScript devant être employé pour accéder aux données de la commande et les manipuler. L'implémentation de cet objet se trouve dans le fichier GridControlClass.js du dossier Configuration\Shared\MM\Scripts\Class.

Ajout de commandes d'arborescence

La commande d'arborescence affiche les données dans un format hiérarchisé et permet aux utilisateurs d'étendre et de réduire les nœuds de l'arborescence. La balise `MM:TREECONTROL` permet de créer des commandes d'arborescence pour tout type d'informations ; à la différence de la commande d'arborescence de base de données décrite à la section *Ajout d'une commande d'arborescence de base de données*, page 123, aucune association à une base de données n'est requise. Dans Dreamweaver, l'Editeur de raccourcis clavier utilise la commande d'arborescence comme représenté dans la figure suivante :



Création d'une commande d'arborescence

La balise `MM:TREECONTROL` crée une commande d'arborescence et peut utiliser d'autres balises afin de compléter la structure, comme indiqué dans la liste suivante :

- `MM:TREECOLUMN` est une balise facultative vide qui définit une colonne dans la commande d'arborescence.
- `MM:TREENODE` est une balise facultative qui définit un nœud dans l'arborescence. Il s'agit d'une balise nonempty qui ne peut contenir que d'autres balises `MM:TREENODE`.

Les attributs des balises MM:TREECONTROL sont les suivants :

Nom d'attribut	Description
name	Nom de la commande d'arborescence.
size	Facultatif. Nombre de lignes visibles dans la commande ; la valeur par défaut est de 5 lignes.
theControl	Facultatif. Si le nombre de nœuds de l'attribut theControl est supérieur à la valeur de l'attribut size, des barres de défilement s'affichent.
multiple	Facultatif. Autorise plusieurs sélections ; la valeur par défaut est une sélection unique.
style	Facultatif. Définition de style pour la hauteur et la largeur de la commande d'arborescence ; si cet attribut est précisé, il est prépondérant sur l'attribut size.
noheaders	Facultatif. Indique que les en-têtes de la colonne doivent être masqués.

Les attributs des balises MM:TREECOLUMN sont les suivants :

Nom d'attribut	Description
name	Nom de la colonne.
value	Chaîne qui doit s'afficher dans l'en-tête de la colonne.
width	Largeur de la colonne exprimée en pixels (les pourcentages ne sont pas pris en charge) ; la valeur par défaut est 100.
align	Facultatif. Indique l'alignement du texte de la colonne : à gauche, à droite ou centré ; la valeur par défaut est à gauche.
state	Indique si la colonne est visible ou masquée.

Pour une meilleure visibilité, les balises TREECOLUMN doivent suivre immédiatement la balise MM:TreeControl, comme indiqué dans l'exemple suivant :

```
<MM:TREECONTROL name="tree1">
<MM:TREECOLUMN name="Column1" width="100" state="visible">
<MM:TREECOLUMN name="Column2" width="80" state="visible">
...
</MM:TREECONTROL>
```

Les attributs MM:TREENODE sont décrits dans le tableau suivant :

Nom d'attribut	Description
name	Nom du nœud.
value	Comporte le contenu du nœud donné. Pour plusieurs colonnes, il s'agit d'une chaîne délimitée par un trait vertical. Pour définir une colonne vide, placez un seul caractère d'espace avant le trait vertical (|).
state	Spécifie que le nœud est développé ou réduit avec les chaînes expanded ou collapsed.
selected	Vous pouvez sélectionner plusieurs nœuds en définissant cet attribut sur plusieurs nœuds d'arborescence, si cette dernière comporte un attribut MULTIPLE.
icon	Facultatif. Index de l'icône intégrée à utiliser, en commençant par 0 : 0 = aucune icône ; 1 = icône de document DW ; 2 = icône multi-document

Par exemple, tous les nœuds de la commande d'arborescence suivante sont développés :

```
<mm:treecontrol name="test" style="height:300px;width:300px">

<mm:treenode value="rootnode1" state="expanded">
<mm:treenode value="node1" state="expanded"></mm:treenode>
<mm:treenode value="node3" state="expanded"></mm:treenode>
</mm:treenode>

<mm:treenode value="rootnode2"  state="expanded">
<mm:treenode value="node2" state="expanded"></mm:treenode>
<mm:treenode value="node4" state="expanded"></mm:treenode>
</mm:treenode>

</mm:treecontrol>
```

Manipulation du contenu d'une commande d'arborescence

Les commandes et les nœuds d'arborescence sous-jacents sont implémentés comme des balises HTML. Ils sont par conséquent analysés par Dreamweaver et stockés dans l'arborescence du document. Ces balises peuvent ensuite être manipulées comme tout autre nœud de document. Pour plus d'informations sur les fonctions et les méthodes DOM, voir Chapitre 5, “Modèle d'objet de document (DOM) Dreamweaver,” on page 133.

Ajout de nœuds Pour programmer l'ajout d'un nœud à une commande d'arborescence existante, définissez la propriété `innerHTML` de la balise `MM:TREECONTROL` ou de l'une des balises `MM:TREENODE` existantes. La définition de la balise `inner HTML` d'un nœud d'arborescence crée un nœud imbriqué.

L'exemple suivant ajoute un nœud tout en haut de l'arborescence :

```
var tree = document.myTreeControl;
//add a top-level node to the bottom of the tree
tree.innerHTML = tree.innerHTML + '<mm:treenode name="node3"¬
  value="node3">';
```

Ajout d'un nœud de niveau supérieur Pour ajouter un nœud enfant à un nœud parent sélectionné, définissez la propriété `innerHTML` du nœud sélectionné.

L'exemple suivant ajoute un nœud enfant à un nœud parent sélectionné :

```
 var tree = document.myTreeControl;
var selNode = tree.selectedNodes[0];
//deselect the node, so we can select the new one
selnode.removeAttribute("selected");
//add the new node to the top of the selected node's children
selNode.innerHTML = '<mm:treenode name="item10" value="New item11" ¬
  expanded selected>' + selNode.innerHTML;
```

Suppression de nœuds Pour supprimer le nœud sélectionné de la structure du document, utilisez la propriété `innerHTML` ou `outerHTML`.

L'exemple suivant supprime le nœud parent sélectionné entier ainsi que ses nœuds enfants :

```
var tree = document.myTreeControl;
var selNode = tree.selectedNodes[0];
selNode.outerHTML = "";
```

Commande de bouton couleur pour les extensions

Outre les types d'entrées standard tels que le texte, la case à cocher et le bouton, Dreamweaver prend en charge `mmcolorbutton`, un autre type d'entrées présent dans les extensions.

Si vous spécifiez `<input type="mmcolorbutton">` dans votre code, un sélecteur de couleurs s'affiche dans l'interface utilisateur. Vous pouvez définir la couleur par défaut du sélecteur en définissant un attribut de valeur dans la balise d'entrée. Si aucune valeur n'est spécifiée, le sélecteur de couleurs s'affiche par défaut en gris, et la propriété value de l'objet d'entrée renvoie une chaîne vide.

L'exemple suivant représente une balise `mmcolorbutton` valide :

```
<input type="mmcolorbutton" name="colorbutton" value="#FF0000">
<input type="mmcolorbutton" name="colorbutton" value="teal">
```

Un bouton couleur comporte un événement, `onChange`, qui est déclenché lorsque la couleur est modifiée.

Il peut se révéler judicieux de synchroniser une zone de texte avec un sélecteur de couleurs. Les exemples suivants créent une zone de texte qui synchronise la couleur de la zone de texte avec celle du sélecteur de couleurs :

```
<input type = "mmcolorbutton" name="fgcolorPicker"
  onChange="document.fgcolorText.value=this.value">
<input type = "test" name="fgcolorText"
  onBlur="document.fgColorPicker.value=this.value">
```

Dans cet exemple, si l'utilisateur modifie la valeur de la zone de texte, puis se déplace par tabulation ou clique ailleurs, le sélecteur de couleurs adopte la couleur spécifiée dans la zone de texte. Lorsque l'utilisateur sélectionne une nouvelle couleur dans le sélecteur, la zone de texte affiche la valeur hexadécimale de cette couleur.

Ajout de contenu Flash à Dreamweaver

Le contenu Flash (fichiers SWF) peut s'afficher dans l'interface de Dreamweaver comme partie d'un objet ou d'une commande. Cette prise en charge de Flash est très utile si vous créez des extensions qui emploient des formulaires, animations, ActionScript ou autre contenu Flash.

Pour résumer, vous permettez aux objets et commandes Dreamweaver d'afficher des boîtes de dialogue (voir Chapitre 6, “Objets de la barre Insérer,” on page 147 pour plus d'informations sur la création d'objets et Chapitre 7, “Commandes,” on page 177 pour plus d'informations sur les commandes) à l'aide de la balise `form` avec la balise `object` pour incorporer votre contenu Flash dans une boîte de dialogue Dreamweaver.

Exemple d'une boîte de dialogue Flash simple

Dans cet exemple, vous utilisez Dreamweaver pour créer une nouvelle commande qui affiche un fichier SWF nommé myFlash.swf lorsque l'utilisateur clique sur cette commande dans le menu Commandes. Pour plus d'informations sur la création de commandes avant d'essayer cet exemple, voir les informations liées aux commandes dans *Extension de Dreamweaver*.

REMARQUE

Cet exemple suppose qu'un fichier SWF nommé myFlash.swf se trouve déjà dans le dossier Configuration/Commands du dossier d'installation de l'application Dreamweaver. Pour tester cet exemple avec votre propre fichier SWF, enregistrez ce fichier SWF dans le dossier Commands de l'application et remplacez toutes les occurrences de myFlash.swf par le nom de votre fichier.

Dans Dreamweaver, ouvrez un nouveau document HTML de base (ce document constituera votre fichier de définition de commande). Entre les balises `title` d'ouverture et de fermeture, entrez `My Flash Movie` afin que le début de votre page indique :

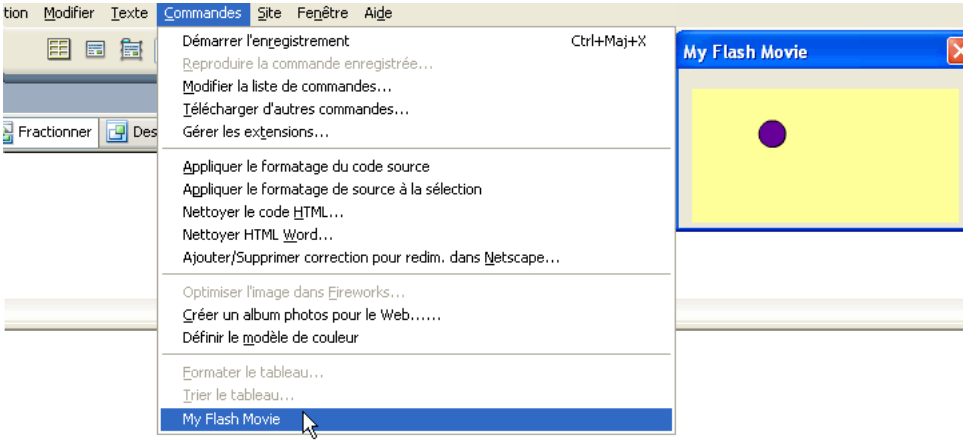
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>My Flash Movie</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
```

Maintenant, enregistrez le fichier sous le nom My Flash Movie.htm dans le dossier Configuration/Commands de l'application (le fichier doit néanmoins rester ouvert). Le fichier doit être enregistré à ce point de la procédure afin que le contenu Flash puisse être incorporé avec un lien relatif. Dans le cas contraire, Dreamweaver tente d'utiliser un chemin absolu.

Revenez au document HTML. Entre les balises `body` d'ouverture et de fermeture, ajoutez des balises `form` d'ouverture et de fermeture. Ensuite, entre ces balises `form`, utilisez l'option de menu Insertion > Médias > Flash pour ajouter votre contenu Flash au fichier de définition de commande. Lorsque vous y êtes invité, sélectionnez le fichier SWF dans le dossier Commands, puis cliquez sur OK. Votre fichier de définition de commande doit maintenant être semblable à l'exemple suivant (bien entendu, les attributs `width` et `height` peuvent être différents, selon les propriétés de votre fichier SWF) :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>My Flash Movie</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<body>
<form>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
  codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=6,0,29,0" width="200" height="100">
  <param name="movie" value="myFlash.swf">
  <param name="quality" value="high">
  <embed src="myFlash.swf" quality="high" pluginspage="http://
  www.macromedia.com/go/getflashplayer" type="application/x-shockwave-
  flash" width="200" height="100"></embed>
</object>
</form>
</body>
</html>
```

Enregistrez à nouveau le fichier. Ensuite, quittez et redémarrez Dreamweaver. Sélectionnez l'option de menu Commandes > My Flash Movie et votre contenu Flash s'affiche dans une boîte de dialogue Dreamweaver, comme indiqué dans la figure suivante :



Cet exemple illustre une mise en œuvre simple de la prise en charge de contenu Flash par Dreamweaver. Lorsque vous serez familiarisé avec la création d'objets et de commandes, ou encore de formulaires plus compliqués, vous pourrez intégrer du contenu Flash dans vos extensions Dreamweaver pour une expérience utilisateur encore plus dynamique. Pour plus d'informations, voir Chapitre 7, “Commandes,” on page 177 sur l'écriture d'une fonction `commandButtons()` permettant d'ajouter des boutons à la boîte de dialogue affichant votre contenu Flash.

CHAPITRE 5

Modèle d'objet de document (DOM) Dreamweaver

5

Dans Macromedia Dreamweaver 8, le modèle d'objet de document (DOM) est une structure essentielle pour les créateurs d'extensions. Il permet l'accès à divers éléments et la manipulation de ces derniers dans le document d'un utilisateur et dans le fichier d'extension.

Un DOM définit la composition des documents créés dans un langage de balisage. En représentant les balises et attributs sous forme sous forme d'objets et de propriétés, le DOM rend les documents et leurs composants accessibles et exploitables par les langages de programmation.

La structure d'un document HTML est révélée par son arborescence. La racine est la balise `HTML` et les deux parties principales sont les balises `HEAD` et `BODY`. Parmi les ramifications de `HEAD` figurent les balises `TITLE`, `STYLE`, `SCRIPT`, `ISINDEX`, `BASE`, `META` et `LINK`. Les ramifications de `BODY` comprennent les titres (`H1`, `H2`, etc.), les éléments de niveau bloc (`P`, `DIV`, `FORM`, etc.), les éléments de niveau texte (`FONT`, `BR`, `IMG`, etc.), ainsi que d'autres types d'éléments. Les éléments émergeant de ces ramifications correspondent, entre autres, à des attributs tels que `WIDTH`, `HEIGHT`, `ALT`, etc.

Dans un DOM, la structure de l'arborescence est maintenue et présentée sous la forme d'une hiérarchie de nœuds parents et de nœuds enfants. Le nœud racine est orphelin, et les nœuds terminaux sont dépourvus de nœuds enfants. A chaque niveau de cette structure HTML, l'élément HTML peut revêtir la forme d'un nœud dans JavaScript. Cette structure vous permet ainsi d'accéder au document et à tous les éléments qui le composent.

Dans JavaScript, vous pouvez référencer un objet de document par son nom ou par son index, comme décrit dans la liste suivante :

- Par nom, comme `document.myForm.myButton`
- Par index, comme `document.forms[0].elements[1]`

Les objets ayant le même nom forment un tableau. Vous pouvez accéder à un objet particulier d'un tableau en incrémentant l'index de zéro comme origine (par exemple, le premier bouton radio intitulé `myRadioGroup` dans le document `myForm` serait référencé par `document.myForm.myRadioGroup[0]`).

De quel DOM de document parlons-nous ?

Il convient de faire la différence entre le DOM du document de l'utilisateur et le DOM de l'extension. Les informations présentées dans ce chapitre s'appliquent aux deux types de documents Dreamweaver, mais la manière de référencer chacun des DOM diffère.

Si vous êtes familier avec l'utilisation de JavaScript dans les navigateurs, vous pouvez désigner les objets du document par `document`. Par exemple, `document.forms[0]`, de la même manière que vous référencez les objets dans les fichiers d'extension. Pour référencer les objets dans le document de l'utilisateur, vous devez toutefois appeler `dw.getDocumentDOM()`, `dw.createDocument()` ou toute autre fonction qui renvoie un objet de document utilisateur.

Ainsi, pour désigner la première image du document actif, vous pouvez écrire `dw.getDocumentDOM().images[0]`. Vous pouvez également stocker l'objet de document dans une variable et vous y référer ultérieurement, comme indiqué dans l'exemple suivant :

```
var dom = dw.getDocumentDOM(); //get the dom of the current document
var firstImg = dom.images[0];
firstImg.src = “myImages.gif”;
```

Ce type d'annotation est commun dans tous les fichiers du dossier Configuration, particulièrement dans les fichiers de commandes. Pour plus d'informations sur la méthode `dw.getDocumentDOM()`, voir la fonction `dreamweaver.getDocumentDOM()` dans le *Guide des API de Dreamweaver*.

DOM Dreamweaver

Le DOM Dreamweaver associe un sous-ensemble d'objets, de propriétés et de méthodes du DOM Niveau 1 du World Wide Web Consortium (W3C) (http://www.w3.org/TR/REC-DOM-Level-1/), combinés à certaines propriétés du DOM de Microsoft Internet Explorer 4.0.

Objets, propriétés et méthodes du DOM Dreamweaver

Le tableau suivant répertorie les objets, les propriétés, les méthodes et les événements pris en charge par le DOM Dreamweaver. Certaines propriétés sont en lecture seule lorsqu'elles sont accessibles en tant que propriétés d'un objet spécifique. Les propriétés en lecture seule, lorsqu'elles sont utilisées dans le contexte recensé, sont marquées d'une puce (•).

Objet	Propriétés	Méthodes	Evénements
window	navigator • document • innerWidth • innerHeight • screenX • screenY •	alert() confirm() escape() unescape() close() setTimeout() clearTimeout() setInterval() clearInterval() resizeTo()	onResize
navigator	platform •	None	None
document	forms • (tableau d'objets de formulaire) images • (tableau d'objets image) layers • (tableau d'objets LAYER, ILAYER et d'objets DIV et SPAN à positionnement absolu) Objets child par nom • nodeType • parentNode • childNodes • documentElement • body • URL • parentWindow •	getElementsBy TagName() hasChildNodes()	onLoad
all tags/ elements	nodeType • parentNode • childNodes • tagName • attributes by name innerHTML outerHTML	getAttribute() setAttribute() removeAttribute() getElementsByTagName() hasChildNodes()	

Objet	Propriétés	Méthodes	Evénements
form	En complément des propriétés disponibles pour toutes les balises : tags:elements • (tableau d'objets button, checkbox, password, radio, reset, select, submit, text, file, hidden, image et textarea) mmcolorbutton *objets child par nom* •	Uniquement les méthodes disponibles pour toutes les balises.	Aucun
layer	En complément des propriétés disponibles pour toutes les balises : visibility left top width height zIndex	Uniquement les méthodes disponibles pour toutes les balises.	Aucun
image	En complément des propriétés disponibles pour toutes les balises : src	Uniquement les méthodes disponibles pour toutes les balises.	onMouseOver onMouseOut onMouseDown onMouseUp
button reset submit	En complément des propriétés disponibles pour toutes les balises : form •	En complément des méthodes disponibles pour toutes les balises : blur() focus()	onClick
checkbox radio	En complément des propriétés disponibles pour toutes les balises : checked form •	En complément des méthodes disponibles pour toutes les balises : blur() focus()	onClick
password text file hidden image (field) textarea	En complément des propriétés disponibles pour toutes les balises : form • value	En complément des méthodes disponibles pour toutes les balises : blur() focus() select()	onBlur onFocus

Objet	Propriétés	Méthodes	Evénements
select	En complément des propriétés disponibles pour toutes les balises : form • options • (an array of option objects) selectedIndex	En complément des méthodes disponibles pour toutes les balises : blur() (Windows uniquement) focus() (Windows uniquement)	onBlur (Windows uniquement) onChange onFocus (Windows uniquement)
option	En complément des propriétés disponibles pour toutes les balises : text	Uniquement les méthodes disponibles pour toutes les balises.	Aucun
mmcolorbutton	En complément des propriétés disponibles pour toutes les balises : name value	Aucun	onChange
array boolean date function math number object string regexp	Correspond à Netscape Navigator 4.0	Correspond à Netscape Navigator 4.0	Aucun
text	nodeType • parentNode • childNodes • data	hasChildNodes()	Aucun
comment	nodeType • parentNode • childNodes • data	hasChildNodes()	Aucun
NodeList	length •	item()	Aucun
NamedNodeMap	length •	item()	Aucun

Propriétés et méthodes de l'objet document

Le tableau suivant décrit en détail les propriétés et les méthodes de l'objet `document` empruntées au DOM Niveau 1 et employées dans Dreamweaver. Les propriétés en lecture seule sont marquées d'une puce (•).

Propriété ou méthode	Valeur renvoyée
`nodeType` •	`Node.DOCUMENT_NODE`
`parentNode` •	`null`
`parentWindow` •	L'objet JavaScript correspond à la fenêtre parente du document (cette propriété n'est pas incluse dans le DOM Niveau 1 ; elle est toutefois prise en charge par Microsoft Internet Explorer 4.0).
`childNodes` •	Une `NodeList` (liste de nœuds) comprend tous les enfants immédiats de l'objet `document`. En règle générale, le document comporte un seul enfant : l'objet `HTML`.
`documentElement` •	L'objet JavaScript correspond à la balise `HTML`. Cette propriété est une forme courte permettant d'obtenir la valeur de `document.childNodes` et d'extraire la balise `HTML` de la `NodeList`.
`body` •	L'objet JavaScript correspond à la balise `BODY`. Cette propriété est une forme courte permettant d'appeler `document.documentElement.childNodes` et d'extraire la balise `BODY` de la `NodeList`. Pour les documents de jeu de cadres, cette propriété renvoie le nœud correspondant au jeu de cadres situé le plus à l'extérieur.
`URL` •	L'URL de type *`file`*`://` du document ou, si le fichier n'a pas été enregistré, une chaîne vide.
`getElementsByTagName(`*`tagName`*`)`	Une `NodeList` pouvant être utilisée pour parcourir les balises de type *`tagName`* (par exemple, `IMG`, `DIV`, etc.). Si l'argument *`tag`* est `LAYER`, la fonction renvoie toutes les balises `LAYER` et `ILAYER` et toutes les balises `DIV` et `SPAN` à positionnement absolu. Si l'argument *`tag`* est `INPUT`, la fonction renvoie tous les éléments du formulaire (si un attribut de nom est spécifié pour un ou plusieurs objets `tagName`, il doit commencer par une lettre conformément à la spécification HTML 4.01 ; à défaut, la longueur du tableau renvoyée par cette fonction sera incorrecte).
`hasChildNodes()`	`true`

Propriétés et méthodes des objets de balise HTML

Chaque balise HTML est représentée par un objet JavaScript. Les balises sont organisées selon une hiérarchie en arborescence dans laquelle la balise `x` est parente de la balise `y` si `y` est entièrement définie entre les balises de début et de fin de `x` (`<x>le contenu de x <y>le contenu de y</y> plus le contenu de x.</x>`). Par conséquent, votre code doit être bien structuré.

Le tableau ci-dessous répertorie les propriétés et les méthodes des objets de balise dans Dreamweaver, ainsi que leurs valeurs de retour ou leurs explications. Les propriétés en lecture seule sont marquées d'une puce (•).

Propriété ou méthode	Valeur renvoyée
`nodeType` •	`Node.ELEMENT_NODE`
`parentNode` •	Balise parente. S'il s'agit de la balise `HTML`, l'objet document est renvoyé.
`childNodes` •	Une `NodeList` (liste de nœuds) comprend tous les enfants immédiats de la balise.
`tagName` •	Nom HTML de la balise, tel que `IMG`, `A` ou `BLINK`. Cette valeur est toujours renvoyée en majuscules.
`attrName`	Chaîne de caractères contenant la valeur de l'attribut de balise spécifié. `tag.`*`attrName`* ne peut pas être utilisé si l'attribut *`attrName`* est un mot réservé dans le langage JavaScript (par exemple, `class`). Dans ce cas, utilisez `getAttribute()` et `setAttribute()`.
`innerHTML`	Code source contenu entre la balise de début et la balise de fin. Par exemple, dans le code `<p><b>Hello</b>, World!</p>`, `p.innerHTML` renvoie `<b>Hello</b>, World!`. Si vous écrivez dans cette propriété, l'arborescence DOM est immédiatement mise à jour de façon à refléter la nouvelle structure du document (cette propriété n'est pas incluse dans le DOM Niveau 1, mais elle est prise en charge par Internet Explorer 4.0).
`outerHTML`	Code source de cette balise, y compris la balise elle-même. Pour l'exemple de code précédent, `p.outerHTML` renvoie `<p><b>Hello</b>, World!</p>`. Si vous écrivez dans cette propriété, l'arborescence DOM est immédiatement mise à jour de façon à refléter la nouvelle structure du document (cette propriété n'est pas incluse dans le DOM Niveau 1, mais elle est prise en charge par Internet Explorer 4.0).

Propriété ou méthode	Valeur renvoyée
`getAttribute(attrName)`	Valeur de l'attribut spécifié, si celui-ci est spécifié explicitement ; sinon, `null`.
`getTranslatedAttribute(attrName)`	Valeur traduite de l'attribut spécifié ou la même valeur que celle renvoyée par `getAttribute()` si la valeur de l'attribut n'est pas traduite (cette propriété n'est pas incluse dans le DOM Niveau 1 ; elle a été ajoutée à Dreamweaver 3 pour prendre en charge la traduction d'attributs).
`setAttribute(attrName, attrValue)`	Ne renvoie aucune valeur. Affecte la valeur spécifiée à l'attribut défini : par exemple, `img.setAttribute("src", "image/roses.gif")`.
`removeAttribute(attrName)`	Ne renvoie aucune valeur. Supprime l'attribut défini et sa valeur du code HTML de la balise.
`getElementsByTagName(tagName)`	Une `NodeList` pouvant être utilisée pour parcourir les balises enfants de type *`tagName`* (par exemple, `IMG`, `DIV`, etc.). Si l'argument *`tag`* est `LAYER`, la fonction renvoie toutes les balises `LAYER` et `ILAYER` et toutes les balises `DIV` et `SPAN` à positionnement absolu. Si l'argument *`tag`* est `INPUT`, la fonction renvoie tous les éléments du formulaire (si un attribut de nom est spécifié pour un ou plusieurs objets `tagName`, il doit commencer par une lettre conformément à la spécification HTML 4.01 ; à défaut, la longueur du tableau renvoyée par cette fonction sera incorrecte).
`hasChildNodes()`	Valeur booléenne indiquant si la balise a des enfants.
`hasTranslatedAttributes()`	Valeur booléenne indiquant si la balise a des attributs traduits (cette propriété n'est pas incluse dans le DOM Niveau 1 ; elle a été ajoutée à Dreamweaver 3 pour prendre en charge la traduction d'attributs).

Propriétés et méthodes des objets texte

Chaque bloc de texte contigu dans un document HTML (par exemple, le texte compris à l'intérieur d'une balise `P`) est représenté par un objet JavaScript. Les objets texte n'ont jamais d'enfants. Le tableau suivant décrit les propriétés et les méthodes d'objets texte empruntées au DOM Niveau 1 et employées dans Dreamweaver. Les propriétés en lecture seule sont marquées d'une puce (•).

Propriété ou méthode	Valeur renvoyée
`nodeType` •	`Node.TEXT_NODE`
`parentNode` •	Balise parente
`childNodes` •	Une `NodeList` vide
`data`	La chaîne de texte réelle. Les entités du texte sont représentées par des caractères uniques (par exemple, le texte `Joseph & I` est renvoyé sous la forme `Joseph & I`).
`hasChildNodes()`	`false`

Propriétés et méthodes des objets de commentaire

Un objet JavaScript représente chaque commentaire HTML. Le tableau suivant décrit en détail les propriétés et les méthodes d'objets de commentaires empruntées au DOM Niveau 1 et employées dans Dreamweaver. Les propriétés en lecture seule sont marquées d'une puce (•).

Propriété ou méthode	Valeur renvoyée
`nodeType` •	`Node.COMMENT_NODE`
`parentNode` •	Balise parente
`childNodes` •	Un tableau `NodeList` vide
`data`	Chaîne de texte comprise entre les marqueurs de commentaires (`<!-- et -->`)
`hasChildNodes()`	`false`

Objets dreamweaver et site

Dreamweaver implémente les objets standard accessibles au moyen du DOM et ajoute deux objets personnalisés : `dreamweaver` et `site`. Ces objets personnalisés sont couramment utilisés dans les API et pour la rédaction d'extensions. Pour plus d'informations sur les méthodes des objets `dreamweaver` et `site`, voir le *Guide des API de Dreamweaver*.

Propriétés de l'objet dreamweaver

L'objet `dreamweaver` possède deux propriétés en lecture seule, comme indiqué dans la liste suivante :

- La propriété `appName` prend la valeur `Dreamweaver`.
- La propriété `appVersion` prend une valeur ayant le format "*versionNumber.releaseNumber.buildNumber* [*languageCode*] (*platform*)".

Par exemple, la valeur de la propriété `appVersion` pour la version Windows suédoise de Dreamweaver 8 correspond à `"8.0.XXXX [se] (Win32)"` ; pour la version Macintosh anglaise, cette valeur est `"8.0.XXXX [en] (MacPPC)"`.

REMARQUE	Vous trouverez la version et le numéro de compilation en sélectionnant l'élément de menu Aide > A propos de.

Les propriétés `appName` et `appVersion` ont été implémentées dans Dreamweaver 3 et ne sont pas disponibles dans les versions antérieures de Dreamweaver. Si vous souhaitez vérifier que l'utilisateur de votre extension dispose de la version 3 ou ultérieure de Dreamweaver, vous pouvez contrôler l'existence de la propriété `appVersion` ou `appName`.

Pour connaître la version spécifique de Dreamweaver, vérifiez en premier lieu l'existence de `appVersion`. Ensuite, pour le numéro de version, procédez comme dans l'exemple suivant :

```
if (dreamweaver.appVersion && ¬
dreamweaver.appVersion.indexOf('3.01') != -1){
  // execute code
}
```

L'objet `dreamweaver` comprend une propriété appelée `systemScript`, qui vous permet de vérifier la langue du système d'exploitation de l'utilisateur. Elle vous sera utile pour inclure dans le code de votre extension des spécificités des systèmes d'exploitation localisés, comme indiqué dans l'exemple suivant :

```
if (dreamweaver,systemScript && (dreamweaver.systemScript.indexOf('ja')!=-
  1){
```

```
SpecialCase
}
```

La propriété `systemScript` renvoie les valeurs suivantes pour les systèmes d'exploitation localisés :

Langue	Valeur
Japonais	`jaja`
Coréen	`koko`
Chinois traditionnel	`zh_twzh_tw`
Chinois simplifié	`zh_cnzh_cn`

Les systèmes d'exploitation localisés dans les langues européennes renvoient la valeur `en`.

Objet site

L'objet `site` n'a aucune propriété. Pour plus d'informations sur les méthodes de l'objet `site`, voir le *Guide des API de Dreamweaver*.

PARTIE 3

API d'extension

3

Etudiez les fonctions que vous devez rédiger lorsque vous créez des objets, barres d'outils, éditeurs de balises, panneaux flottants, comportements de serveurs, composants ou modèles de serveur.

CHAPITRE 6

Objets de la barre Insérer

6

Dans Macromedia Dreamweaver, les objets insèrent des chaînes de code spécifiques dans le document de l'utilisateur. Ils résident en règle générale dans la barre Insérer, dans le menu Insertion ou dans les deux à la fois. Les objets permettent à l'utilisateur d'ajouter un contenu (images, calques, tableaux, etc.) en cliquant sur des icônes ou en sélectionnant des options de menu. Vous pouvez ajouter des éléments à la barre Insérer pour permettre aux utilisateurs d'effectuer automatiquement certaines tâches répétitives, voire créer des boîtes de dialogue pour leur permettre de définir des attributs spécifiques.

Les objets sont stockés dans le dossier Configuration/Objects du dossier de l'application Dreamweaver. Les sous-dossiers du dossier Objects sont regroupés en fonction de l'emplacement des objets dans la barre Insérer ; vous pouvez les ouvrir pour visualiser la construction des objets actuels. Par exemple, vous pouvez ouvrir le fichier Configuration/Objects/Common/Hyperlink.htm pour afficher le code correspondant au bouton permettant d'insérer l'objet Hyperlien de la barre Insérer.

Le tableau ci-dessous recense les fichiers utilisés pour créer un objet.

Chemin	Fichier	Description
Configuration/Objects/*type_objet*/	*nom_objet*.htm	Indique l'élément à insérer dans le document.
Configuration/Objects/*type_objet*/	*nom_objet*.js	Contient les fonctions à exécuter.
Configuration/Objects/*type_objet*/	*nom_objet*.gif	Contient l'image qui s'affiche sur la barre Insérer.
Configuration/Objects	insertbar.xml	Désigne les objets qui s'affichent sur la barre Insérer, ainsi que l'ordre d'affichage correspondant.

Fonctionnement des fichiers d'objet

Les objets se composent des éléments suivants :

- Le fichier HTML définissant l'élément inséré dans le document.

  La section `HEAD` d'un fichier d'objet contient des fonctions JavaScript (ou fait référence à des fichiers JavaScript externes) traitant les entrées de formulaire de la section `BODY` et contrôlant le contenu ajouté dans le document de l'utilisateur. La section `BODY` d'un fichier d'objet peut contenir un formulaire HTML acceptant les paramètres de l'objet (le nombre de lignes et de colonnes à insérer dans un tableau, par exemple) et activant une boîte de dialogue permettant aux utilisateurs de saisir différents attributs.

  REMARQUE Les objets les plus simples contiennent uniquement le code HTML à insérer, sans les balises `BODY` et `HEAD`. Pour plus d'informations, voir la section Customizing Dreamweaver (Personnalisation de Dreamweaver) du centre de support Macromedia.

- L'image figurant dans la barre Insérer (18 x 18 pixels).
- Additions au fichier insertbar.xml. Le fichier insertbar.xml définit l'endroit où apparaît l'objet dans la barre Insérer.

Lorsqu'un utilisateur sélectionne un objet en cliquant sur une icône de la barre Insérer ou en choisissant un élément du menu Insertion, les événements suivants se produisent :

1. Dreamweaver appelle la fonction `canInsertObject()` pour déterminer s'il doit afficher une boîte de dialogue.
2. La balise `FORM` est recherchée dans le fichier d'objet. Si un formulaire a été défini et que l'option Afficher la boîte de dialogue lors de l'insertion d'objets est activée dans la catégorie Général de la boîte de dialogue Préférences, Dreamweaver appelle la fonction `windowDimensions()`, si elle est définie, pour déterminer la taille de la boîte de dialogue dans laquelle doit s'afficher le formulaire. Si le fichier d'objet ne contient aucun formulaire, Dreamweaver n'affiche pas de boîte de dialogue et ignore l'étape 2.
3. Si Dreamweaver affiche une boîte de dialogue à l'étape 1, l'utilisateur saisit les paramètres de l'objet (tels que le nombre de lignes et de colonnes d'un tableau) dans les champs de texte de la boîte de dialogue, puis clique sur OK.
4. Dreamweaver appelle la fonction `objectTag()` et insère la valeur renvoyée correspondante dans le document après la sélection en cours (celle-ci n'est pas remplacée).
5. Si Dreamweaver ne trouve pas la fonction `objectTag()`, il recherche une fonction `insertObject()` et l'appelle.

Fichier de définition de la barre Insérer

Le fichier Configuration/Objects/insertbar.xml définit les propriétés de la barre Insérer. Ce fichier XML contient la définition de chacun des objets, dans leur ordre d'apparition.

La première fois qu'un utilisateur démarre Dreamweaver, la barre Insérer s'affiche horizontalement au-dessus du document. Sa visibilité et sa position sont ensuite enregistrées dans le Registre.

Hiérarchie des balises du fichier Insertbar.xml

L'exemple ci-dessous affiche le format et la hiérarchie des balises imbriquées du fichier insertbar.xml :

```
<?xml version="1.0" ?>
<!DOCTYPE insertbarset SYSTEM "-//Macromedia//DWExtension insertbar 5.0">

<insertbar xmlns:MMString="http://www.macromedia.com/schemes/data/string/">

<category id="DW_Insertbar_Common" MMString:name="insertbar/categorycommon"
  folder="Common">
    <button id="DW_Hyperlink" image="Common\Hyperlink.png"
    MMString:name="insertbar/hyperlink" file="Common\Hyperlink.htm" />
    <button id="DW_Email" image="Common\E-Mail Link.png"
    MMString:name="insertbar/email" file="Common\E-Mail Link.htm" />
    <separator />
  <menubutton id="DW_Images" MMString:name="insertbar/images"
  image="Common\Image.png">
    <button id="DW_Image" image="Common\Image.png"
    MMString:name="insertbar/image" file="Common\Image.htm" />
  ...
    </menubutton>
    <separator />
      <button id="DW_TagChooser" MMString:name="insertbar/tagChooser"
      image="Common\Tag Chooser.gif" command="dw.showTagChooser()"
      codeOnly="TRUE"/>
  </category>
    ...
  </insertbar>
```

REMARQUE La fin du contenu des balises `insertbar` et `category` est indiquée par les balises de fermeture `</insertbar>` et `</category>`. En revanche, les balises button, checkbutton et separator ne sont associées à aucune balise de fermeture. La fin des attributs et du contenu des balises button, checkbutton et separator est indiquée par une barre oblique (/) insérée avant le crochet de fermeture.

Balises de définition de la barre Insérer

Le fichier insertbar.xml contient les balises et les attributs suivants :

<insertbar>

Description

Cette balise désigne le contenu du fichier de définition de la barre Insérer. La balise de fermeture `</insertbar>` indique la fin du contenu.

Attributs

Aucun.

Exemple

```
<insertbar>
   <category id="DW_Insertbar_Common" folder="Common">
     <button id="DW_Hyperlink" image="Common\Hyperlink.gif"
     file="Common\Hyperlink.htm"/>
...
</insertbar>
```

<category>

Description

Cette balise définit une catégorie de la barre Insérer (par exemple, Commun, Formulaires ou HTML). La balise de fermeture `</category>` indique la fin du contenu.

REMARQUE	Par défaut, la barre Insérer regroupe différentes catégories (par exemple, Commun, Formulaires ou HTML). Dans les versions précédentes de Dreamweaver, la barre Insérer se composait de différents onglets. Les utilisateurs peuvent définir leurs préférences quant à l'organisation des objets de la barre Insérer (par catégorie ou par onglet). Si l'utilisateur choisit l'organisation par onglet, la balise category définit chacun d'eux.

Attributs

`id, {folder}, {showIf}`

Exemple

```
<category id="DW_Insertbar_Common" folder="Common">
  <button id="DW_Hyperlink" image="Common\Hyperlink.gif"
  file="Common\Hyperlink.htm"/>
</category>
```

<menubutton>

Description

Cette balise définit un menu déroulant dans la barre Insérer.

Attributs

`id, image, {showIf}, {name}, {folder}`

Exemple

```
<menubutton
   id="DW_ImageMenu"
   name="Images"
   image="Common\imagemenu.gif"
   folder="Images">

   <button id="DW_Image"
      image="Common\Image.gif"
      enabled=""
      showIf=""
      file="Common\Image.htm" />
</menubutton>
```

<button />

Description

Cette balise définit un bouton de la barre Insérer permettant à l'utilisateur d'exécuter le code spécifié par les attributs `command` ou `file`.

Attributs

`id, image, name, {canDrag}, {showIf}, {enabled}, {command}, {file}, {tag}, {codeOnly}`

Exemple

```
<button id="DW_Object"
image="Common\Object.gif"
name=”Object”
enabled="true"
showIf=""
file="Common\Obect.htm"
  />
```

<checkbutton />

Description

Ce type de bouton peut se voir attribuer l'état activé ou désactivé. Lorsque vous cliquez dessus, ce bouton apparaît en surbrillance comme étant enfoncé. Lorsqu'il est désactivé, il s'affiche sous forme plate. Dreamweaver inclut les états suivants : au passage de la souris ; enfoncé ; au passage de la souris si enfoncé ; désactivé si enfoncé. La commande doit faire en sorte qu'un clic sur le bouton d'activation provoque un changement de son état.

Attributs

```
id, image, checked, {showIf}, {enabled}, {command}, {file}, {tag}, {name},
{codeOnly}
```

Exemple

```
<checkbutton id="DW_StandardView"
name = "Standard View"
image="Tools\Standard View.gif"
checked="_View_Standard"
command="dw.getDocumentDOM().setShowLayoutView(false)"/>
```

<separator/>

Description

Cette balise affiche une ligne verticale sur la barre Insérer.

Attributs

```
{showIf}
```

Exemple

```
<separator showIf="_VIEW_CODE"/>
```

Attributs des balises de définition de la barre Insérer

La signification des attributs correspondant aux balises de définition de la barre Insérer est la suivante :

id="unique id"

Description

L'attribut `id` correspond à l'identificateur des boutons affichés dans la barre Insérer. Chaque élément du fichier doit posséder un attribut `id` unique.

Exemple

```
id="DW_Anchor"
```

image="image_path”

Description

Cet attribut indique le chemin d'accès, par rapport au dossier Configuration de Dreamweaver, du fichier de l'icône affichée dans la barre Insérer. Le format de cette icône doit figurer parmi les formats pris en charge par Dreamweaver, mais il s'agit en principe d'un fichier au format GIF ou JPEG, d'une taille de 18 x 18 pixels.

Exemple

```
image="Common/table.gif"
```

canDrag="Boolean”

Description

Cet attribut spécifie si l'utilisateur a la possibilité de faire glisser l'icône dans le code ou dans l'espace de travail pour insérer l'objet correspondant dans le document. Si vous ne spécifiez pas cet attribut, la valeur par défaut est `true`.

Exemple

```
canDrag="false"
```

showIf="enabler"

Description

Cet attribut indique que ce bouton doit s'afficher dans la barre Insérer uniquement si la valeur de l'activateur Dreamweaver désigné est réglée sur `true`. Si vous ne spécifiez pas l'attribut `showIf`, le bouton s'affiche systématiquement. Les activateurs gérés sont `_SERVERMODEL_ASP`, `_SERVERMODEL_ASPNET`, `_SERVERMODEL_JSP`, `_SERVERMODEL_CFML` (pour toutes les versions de Macromedia ColdFusion), `_SERVERMODEL_CFML_UD4` (réservé à Dev version 4 de ColdFusion), `_SERVERMODEL_PHP`, `_FILE_TEMPLATE`, `_VIEW_CODE`, `_VIEW_DESIGN`, `_VIEW_LAYOUT`, `_VIEW_EXPANDED_TABLES` et `_VIEW_STANDARD`.

Pour spécifier plusieurs activateurs, séparez-les par des virgules (chaque virgule signifie ET). Pour exclure des activateurs (NOT), utilisez `un` point d'exclamation (`!`).

Exemple

Si vous souhaitez qu'un bouton apparaisse uniquement en mode Code dans une page ASP, définissez les activateurs comme suit :

```
showIf="_VIEW_CODE, _SERVERMODEL_ASP"
```

enabled="enabler"

Description

Cet attribut indique que l'utilisateur peut accéder à l'élément si la valeur de *activateur_DW* est réglée sur `true`. Si vous ne spécifiez pas la fonction `enabled`, l'élément est systématiquement activé par défaut. Les activateurs possibles sont : `_SERVERMODEL_ASP`, `_SERVERMODEL_ASPNET`, `_SERVERMODEL_JSP`, `_SERVERMODEL_CFML` (pour toutes les versions de ColdFusion), `_SERVERMODEL_CFML_UD4` (applicable uniquement à UltraDev version 4 de ColdFusion), `_SERVERMODEL_PHP`, `_FILE_TEMPLATE`, `_VIEW_CODE`, `_VIEW_DESIGN`, `_VIEW_LAYOUT`, `_VIEW_EXPANDED_TABLES` et `_VIEW_STANDARD`.

Pour spécifier plusieurs activateurs, séparez-les par des virgules (chaque virgule signifie ET). Pour exclure des activateurs (NOT), utilisez un point d'exclamation (`!`).

Exemple

Pour que le bouton soit disponible en mode Code uniquement, indiquez :

```
enabled="_VIEW_CODE"
```

Le bouton s'affiche alors en grisé dans les autres modes.

checked="enabler"

Description

L'attribut checked est obligatoire si vous insérez la balise checkbutton.

L'élément est activé si la valeur de *DW_activateur* est définie sur `true`. Les activateurs possibles sont : `_SERVERMODEL_ASP`, `_SERVERMODEL_ASPNET`, `_SERVERMODEL_JSP`, `_SERVERMODEL_CFML` (pour toutes les versions de ColdFusion), `_SERVERMODEL_CFML_UD4` (applicable uniquement à UltraDev version 4 de ColdFusion), `_SERVERMODEL_PHP`, `_FILE_TEMPLATE`, `_VIEW_CODE`, `_VIEW_DESIGN`, `_VIEW_LAYOUT`, `_VIEW_EXPANDED_TABLES` et `_VIEW_STANDARD`.

Pour spécifier plusieurs activateurs, séparez-les par des virgules (chaque virgule signifie ET). Pour exclure des activateurs (NOT), utilisez un point d'exclamation (`!`).

Exemple

```
checked="_View_Layout"
```

command="API_function"

Description

Au lieu d'indiquer à Dreamweaver de se référer à un fichier HTML comportant le code à insérer, vous pouvez associer une commande à cette balise. Lorsque l'utilisateur clique sur le bouton auquel se rapporte cette balise, Dreamweaver effectue alors la commande spécifiée.

Exemple

```
command="dw.showTagChooser()"
```

file="chemin_fichier"

Description

L'attribut `file` indique le chemin d'accès au fichier d'un objet par rapport au dossier Configuration de Dreamweaver. Dreamweaver extrait le libellé de l'info-bulle associée à l'icône de l'objet du titre du fichier d'objet, à moins que vous ne spécifiez l'attribut `name`.

Exemple

```
file="Templates/Editable.htm"
```

tag="editor"

Description

Cet attribut ordonne à Dreamweaver d'appeler un éditeur de balises. En mode Code, si l'attribut `tag` est défini et que l'utilisateur clique sur l'objet, Dreamweaver appelle la boîte de dialogue Balise. En mode Code, si vous spécifiez les attributs `tag` et `command`, Dreamweaver appelle l'éditeur de balises. En mode Création, si `codeOnly="TRUE"` et que l'attribut `file` n'est pas spécifié, Dreamweaver MX appelle l'affichage à deux volets, place le focus dans la partie code et appelle l'éditeur de balises.

Exemple

```
tag = "form"
```

name="texte_infobulle"

Description

L'attribut `name` définit l'info-bulle qui s'affiche lorsque le curseur de la souris pointe sur l'objet concerné. Si vous spécifiez un fichier d'objet sans spécifier l'attribut `name`, Dreamweaver utilise le nom du fichier comme info-bulle.

REMARQUE	Si vous ne spécifiez pas l'attribut name, l'objet ne pourra pas être regroupé dans la catégorie Favoris de l'interface de la barre Insérer.

Certains objets de la barre Insérer utilisent une variante de l'attribut name comportant le préfixe `MMString`. Le préfixe MMString indique une chaîne localisée ; ces valeurs sont décrites à la section *Localisation d'une extension*, page 114.

Exemple

```
name = "cfoutput"
```

Modification de la barre Insérer

Vous pouvez déplacer les objets d'une catégorie à une autre, renommer les différentes catégories et supprimer de manière définitive certains objets du panneau. Pour que vos modifications soient prises en compte et apparaissent dans la barre Insérer, vous devez redémarrer Dreamweaver ou recharger les extensions. Pour plus d'informations sur le rechargement des extensions, voir *Rechargez les extensions.*, page 111.

Pour déplacer ou copier un objet répertorié dans une catégorie de la barre Insérer vers une autre catégorie ou vers un emplacement différent de la même catégorie :

1. Enregistrez une copie de sauvegarde du fichier insertbar.xml (nommez-la par exemple insertbar.sauvegarde.xml).
2. Ouvrez le fichier insertbar.xml d'origine.
3. Recherchez la balise `button` représentant l'objet à déplacer ou à copier. Par exemple, pour déplacer l'objet Image de son emplacement au sein de la catégorie Commun, recherchez la balise `button` dont l'attribut `id` est `"DW_Image"`.
4. Coupez ou copiez l'ensemble de la balise `button`.
5. Recherchez la balise `category` représentant la catégorie dans laquelle vous souhaitez déplacer ou copier l'objet.

6. Recherchez l'emplacement de la catégorie dans lequel vous souhaitez faire apparaître l'objet.
7. Collez la balise `button`.
8. Enregistrez le fichier insertbar.xml.
9. Rechargez les extensions.

Pour supprimer un objet de la barre Insérer :

1. Enregistrez une copie de sauvegarde du fichier insertbar.xml (nommez-la par exemple insertbar.sauvegarde.xml).
2. Ouvrez le fichier insertbar.xml d'origine.
3. Recherchez la balise `button` représentant l'objet à supprimer.
4. Supprimez l'intégralité de la balise `button`.
5. Enregistrez le fichier insertbar.xml.
6. Sur votre disque, retirez les fichiers HTML, GIF et JavaScript du dossier dans lequel ils se trouvent pour les placer dans un dossier non répertorié dans le fichier insertbar.xml. Par exemple, vous pouvez créer un nouveau dossier nommé « Inutilisé » dans le dossier Configuration/Objects et y placer les fichiers de l'objet (si vous êtes certain de vouloir supprimer cet objet, vous pouvez supprimer les fichiers de manière définitive. Il peut néanmoins s'avérer utile de conserver une copie de sauvegarde de ces fichiers en vue de pouvoir les restaurer, le cas échéant).
7. Rechargez les extensions.

Pour modifier l'ordre des catégories de la barre Insérer :

1. Enregistrez une copie de sauvegarde du fichier insertbar.xml (nommez-la par exemple insertbar.sauvegarde.xml).
2. Ouvrez le fichier insertbar.xml d'origine.
3. Recherchez la balise `category` correspondant à la catégorie à déplacer, puis sélectionnez-la ainsi que l'ensemble de son contenu.
4. Coupez cette balise.
5. Collez la balise dans son nouvel emplacement. Veillez à ne pas coller la balise au sein d'une autre balise `category`.
6. Enregistrez le fichier insertbar.xml.
7. Rechargez les extensions.

Pour créer une catégorie :

1. Enregistrez une copie de sauvegarde du fichier insertbar.xml (nommez-la par exemple « insertbar.sauvegarde.xml »).
2. Ouvrez le fichier insertbar.xml d'origine.
3. Créez une balise category et indiquez le dossier par défaut ainsi que l'ensemble des objets contenus dans cette catégorie.
4. Pour plus d'informations sur la syntaxe des balises du fichier insertbar.xml, voir *Balises de définition de la barre Insérer*, page 150.
5. Enregistrez le fichier insertbar.xml.
6. Rechargez les extensions.

Ajout d'objets à la barre Insérer

Vous pouvez ajouter des objets à la barre Insérer. Pour que vos modifications soient prises en compte et apparaissent dans la barre Insérer, vous devez redémarrer Dreamweaver ou recharger les extensions. Pour plus d'informations sur le rechargement des extensions, voir *Rechargez les extensions.*, page 111.

Pour ajouter un nouvel objet à la barre Insérer, procédez comme suit :

1. Définissez la chaîne spécifique de code destinée au document de l'utilisateur par le biais de HTML et, en option, JavaScript.
2. Identifiez ou créez l'image (18 x 18 pixels) correspondant au bouton à intégrer à l'interface de Dreamweaver.

   Si vous créez une image de taille supérieure, Dreamweaver la redimensionne pour lui attribuer une taille de 18 x 18 pixels. Si vous ne créez pas d'image pour un objet, une icône d'objet par défaut représentée par un point d'interrogation (?) apparaît dans la barre Insérer.
3. Ajoutez les nouveaux fichiers au dossier Configuration/Objects.
4. Modifiez le fichier insertbar.xml pour indiquer l'emplacement des nouveaux fichiers et spécifiez les attributs (voir *Fichier de définition de la barre Insérer*, page 149) définissant l'aspect du bouton.
5. Redémarrez Dreamweaver ou rechargez les extensions.

Le nouvel objet apparaît dans la barre Insérer à l'emplacement indiqué.

REMARQUE	Même si les fichiers d'objet peuvent être stockés dans des dossiers séparés, il est important que le nom de chaque fichier soit unique. La fonction `dom.insertObject()`, par exemple, recherche les fichiers dans l'ensemble du dossier Objects sans tenir compte des sous-dossiers (pour plus d'informations sur la fonction `dom.insertObject()`, voir le *Guide des API de Dreamweaver)*. Si le dossier Forms contient un fichier nommé Button.htm et qu'un autre fichier nommé Button.htm se trouve également dans le dossier MyObjects, Dreamweaver ne parvient pas à les différencier. S'il existe deux instances du fichier Button.htm, `dom.insertObject()` affiche deux objets nommés Button ; il est dès lors difficile pour l'utilisateur de les différencier.

Ajout d'objets au menu Insertion

Pour ajouter ou contrôler la position d'un objet du menu Insérer (ou de tout autre menu), modifiez le fichier menus.xml. Ce fichier contrôle l'intégralité de la structure de menus pour Dreamweaver. Pour plus d'informations sur la modification du fichier menus.xml, voir Chapitre 8, “Menus et commandes de menu,” on page 191.

Si vous souhaitez distribuer l'extension à d'autres utilisateurs de Dreamweaver, voir *Utilisation de Extension Manager*, page 116 pour en savoir plus sur le conditionnement des extensions.

Exemple simple d'insertion d'un objet

Cet exemple ajoute à la barre Insérer un objet permettant aux utilisateurs de rayer d'une ligne (barrer) le texte sélectionné en cliquant sur un bouton. Cet objet s'utilise lors de l'insertion de commentaires concernant l'édition d'un document.

Cet exemple implique des manipulations du texte ; vous pouvez par conséquent découvrir certains des objets existants dans le menu déroulant Texte de la catégorie HTML de la barre Insérer et vous en servir comme d'un modèle. Par exemple, vous pouvez examiner les fichiers d'objet Bold, Emphasis et Heading pour observer des fonctionnalités similaires, dont le principe consiste à insérer une balise de part et d'autre du texte sélectionné dans Dreamweaver.

La procédure de création de l'objet Strikethrough est la suivante :

- Création du fichier HTML
- Ajout de fonctions JavaScript
- Création de l'image
- Modification du fichier insertbar.xml
- Ajout d'une boîte de dialogue

Création du fichier HTML

Le titre de l'objet est indiqué entre les balises `title` d'ouverture et de fermeture. Vous spécifiez également le langage de script JavaScript.

Pour créer le fichier HTML :

1. Créez un document vierge.
2. Ajoutez le code suivant :

```
<html>
<head>
<title>Strikethrough</title>
<script language="javascript">
</script>
</head>
<body>
</body>
</html>
```

3. Enregistrez le fichier sous le nom Strikethrough.htm dans le dossier Configuration/Objects/Text.

Ajout de fonctions JavaScript

Dans cet exemple, les fonctions JavaScript définissent le comportement de l'objet Strikethrough et insèrent le code correspondant. Vous devez placer toutes les fonctions API dans la section `HEAD` du fichier. Les fichiers d'objet déjà créés, par exemple, Configuration/Objects/Text/Em.htm, sont basés sur le même modèle de fonctions et de commentaires.

La première fonction utilisée par le fichier de définition d'objet est la fonction `isDOMRequired()`, qui indique si le mode Création doit être synchronisé avec le mode Code existant avant de poursuivre l'exécution. Cependant, l'objet Superscript peut s'utiliser avec de nombreux autres objets en mode Code ; une synchronisation imposée n'est donc pas nécessaire.

Pour ajouter la fonction isDOMRequired() :

1. Dans la section `HEAD` du fichier Strikethrough.htm, ajoutez la fonction ci-dessous entre les balises `script` d'ouverture et de fermeture :

```
<script language="javascript">
  function isDOMRequired() {
    // Return false, indicating that this object is available in Code
  view.
```

```
    return false;
}
</script>
```

2. Enregistrez le fichier.

Ensuite, indiquez si vous souhaitez utiliser la fonction `objectTag()` ou `insertObject()`. L'opération effectuée par l'objet Barré consiste simplement à insérer la balise `s` de part et d'autre du texte sélectionné. Il ne remplit donc pas les conditions nécessaires à l'utilisation de la fonction `insertObject()` (voir *insertObject()*, page 172).

Dans la fonction `objectTag()`, indiquez `dw.getFocus()` pour déterminer si le mode actuel correspond au mode Code. Si le mode Code est actif, la fonction insère la balise appropriée (majuscule ou minuscule) de part et d'autre du texte sélectionné. Si le mode Création est actif, la fonction utilise `dom.applyCharacterMarkup()` pour assigner le format du texte sélectionné. Notez que cette fonction s'utilise uniquement avec les balises prises en charge (pour plus d'informations, voir `dom.applyCharacterMarkup()` dans le *Guide des API de Dreamweaver*). Pour utiliser d'autres balises ou opérations, il sera peut-être nécessaire de faire appel à d'autres fonctions API. Une fois le format appliqué par Dreamweaver, le point d'insertion (curseur) réapparaît dans le document, sans message ni invite préalable. La procédure ci-dessous illustre la forme de la fonction `objectTag()` une fois ces informations renseignées :

Pour ajouter la fonction objectTag :

1. Dans la section `HEAD` du fichier Strikethrough.htm, ajoutez la fonction ci-dessous après la fonction `isDOMRequired()` :

```
function objectTag() {
  // Determine if the user is in Code view.
  var dom = dw.getDocumentDOM();
  if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
    var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper
  Case", "") == 'TRUE');
  // Manually wrap tags around selection.
    if (upCaseTag){
      dom.source.wrapSelection('<S>','</S>');
    }else{
     dom.source.wrapSelection('<s>','</s>');
    }
  // If the user is not in Code view, apply the formatting in Design
  view.
  }else if (dw.getFocus() == 'document'){
    dom.applyCharacterMarkup("s");
  }

  // Just return--don't do anything else.
```

```
  return;
}
```

2. Enregistrez le fichier sous le nom Strikethrough.htm dans le dossier Configuration/Objects/Text.

Au lieu d'inclure les fonctions JavaScript dans la section `HEAD` du fichier HTML, vous pouvez créer un fichier JavaScript distinct. Cette organisation à part s'utilise notamment pour les objets contenant plusieurs fonctions ou des fonctions susceptibles d'être partagées par d'autres objets.

Pour séparer le fichier de définition d'objet HTML des fonctions JavaScript de prise en charge :

1. Créez un document vierge.
2. Collez toutes les fonctions JavaScript dans le fichier.
3. Supprimez ces fonctions du fichier Strikethrough.htm, puis ajoutez le nom du fichier JavaScript à l'attribut `src` de la balise script, comme indiqué dans l'exemple suivant :

```
<html>
<head>
<title>Strikethrough</title>
<script language="javascript" src="Strikethrough.js">
</script>
</head>
<body>
</body>
</html>
```

4. Enregistrez le fichier Strikethrough.htm.
5. Enregistrez le fichier qui contient à présent les fonctions JavaScript sous le nom Strikethrough.js dans le dossier Configuration/Objects/Text.

Création de l'image

Pour créer l'image utilisée dans la barre Insérer :

1. Créez une image GIF de 18 x 18 pixels, ainsi que l'indique la figure suivante :

2. Enregistrez le fichier sous le nom Strikethrough.gif dans le dossier Configuration/Objects/Text.

Modification du fichier insertbar.xml

Vous devez à présent modifier le fichier insertbar.xml file, afin que Dreamweaver puisse associer ces deux éléments à l'interface de la barre Insérer.

REMARQUE	Avant de modifier le fichier insertbar.xml file, vous pouvez l'enregistrer sous l'extension insertbar.xml.bak de manière à conserver une copie de sauvegarde.

Le code contenu dans le fichier insertbar.xml désigne tous les objets figurant dans la barre Insérer.

- Chaque balise `category` du fichier XML crée une catégorie dans l'interface.
- Chaque balise `menubutton` crée un menu déroulant dans la barre Insérer.
- Chaque balise `button` du fichier XML affiche une icône dans la barre Insérer et la relie au fichier HTML ou à la fonction correspondants.

Pour ajouter le nouvel objet à la barre Insérer :

1. Recherchez la ligne suivante située au début du fichier inserbar.xml :

   ```
   <category id="DW_Insertbar_Common" MMString:name="insertbar/category/
     common" folder="Common">
   ```

   Cette ligne indique le début de la catégorie Commun de la barre Insérer.

2. Créez une nouvelle ligne à la suite de la balise category, puis insérez la balise `button` en lui assignant les attributs `id`, `image` et `file` correspondant à l'objet Strikethrough.

   L'ID doit correspondre au nom unique du bouton. Conformément aux conventions de dénomination, utilisez `DW_Text_Strikethrough` pour cet objet. Les attributs `image` et `file` indiquent simplement à Dreamweaver l'emplacement des fichiers en rapport avec l'objet, comme indiqué ci-dessous :

   ```
   <button id="DW_Text_Strikethrough"
   image="Text\Strikethrough.gif"
   file="Text\Strikethrough.htm"/>
   ```

3. Enregistrez le fichier insertbar.xml.
4. Rechargez les extensions (voir *Rechargez les extensions.*, page 111).

Le nouvel objet apparaît au début de la catégorie Commun de la barre Insérer, comme indiqué ci-dessous :

Ajout d'une boîte de dialogue

Vous pouvez ajouter un formulaire à votre objet de manière à ce que l'utilisateur puisse indiquer des commentaires avant que Dreamweaver n'insère le code spécifié (par exemple, l'objet Hyperlien nécessite la saisie des valeurs texte, lien, cible, index de catégorie, titre et clé d'accès). Dans cet exemple, vous ajoutez un formulaire à l'objet Barré créé dans l'exemple précédent. Le formulaire entraîne l'ouverture d'une boîte de dialogue permettant à l'utilisateur de colorer le texte en rouge et d'ajouter la balise Strikethrough.

Cet exemple suppose que vous avez déjà créé un fichier JavaScript distinct nommé Strikethrough.js.

Tout d'abord, dans le fichier Strikethrough.js, ajoutez la fonction appelée par le formulaire si l'utilisateur modifie la couleur du texte. Cette fonction est similaire à la fonction `objectTag()` de l'objet Strikethrough, mais elle est facultative.

Pour créer la fonction :

1. Après la fonction `objectTag()` dans Strikethrough.js, créez la fonction `fontColorRed()` en entrant le code suivant :

```
function fontColorRed(){
  var dom = dw.getDocumentDOM();
    if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){

      var upCaseTag = (dw.getPreferenceString("Source Format", "Tags
  Upper Case", "") == 'TRUE');


      // Manually wrap tags around selection.
      if (upCaseTag){
          dom.source.wrapSelection('<FONT COLOR="#FF0000">','</FONT>');
      }else{
          dom.source.wrapSelection('<font color="#FF0000">','</font>');
          }
    }else if (dw.getFocus() == 'document'){
       dom.applyFontMarkup("color", "#FF0000");
      }

    // Just return -- don't do anything else.
    return;
}
```

REMARQUE Parce que `dom.applyCharacterMarkup()` ne gère pas les modifications de couleur de police, vous devez identifier la fonction API correspondant à ce type d'opération. (Pour plus d'informations, voir `dom.applyFontMarkup()` dans le *Guide des API de Dreamweaver*).

2. Enregistrez le fichier sous le nom Strikethrough.js.

Ensuite, ajoutez le formulaire au fichier Strikethrough.htm. Le formulaire illustré dans l'exemple ci-dessous est une simple case à cocher permettant d'appeler la fonction `fontColorRed()` lorsqu'elle est activée. La balise `form` permet de définir votre formulaire, la balise table assure la définition de sa mise en forme (si vous n'effectuez pas cette étape, la taille de la boîte de dialogue ou la disposition des champs risque d'être incorrecte).

Pour ajouter le formulaire :

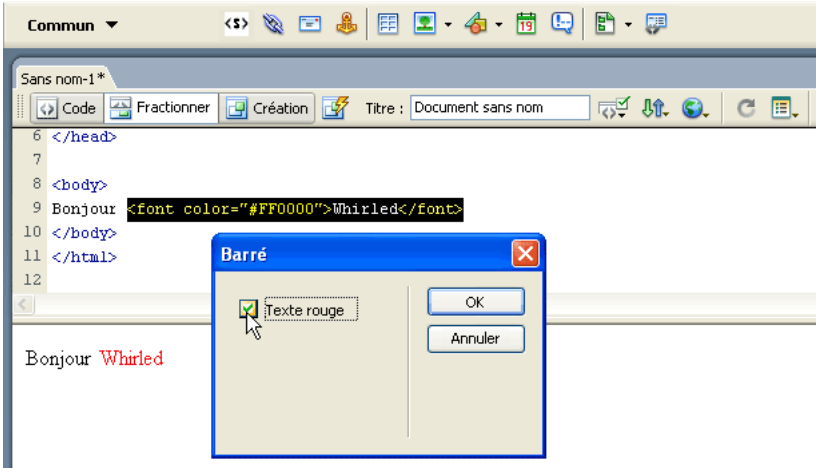
1. Ajoutez le code ci-dessous après la balise `body` :

```
<form>
<table border="0" height="100" width="100">
  <tr valign="baseline">
  <td align="left" nowrap>
  <input type="checkbox" name="red" onClick=fontColorRed()>Red text</
  input>
  </td>
  </tr>
</table>
</form>
```

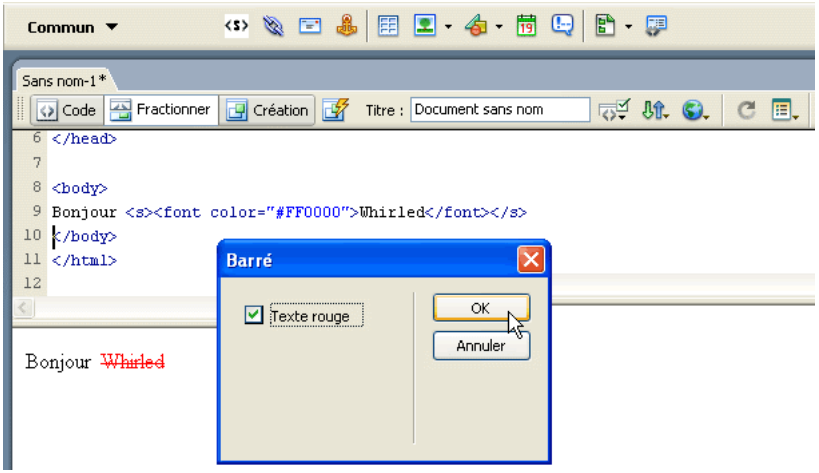
2. Enregistrez le fichier sous le nom Strikethrough.htm.
3. Rechargez les extensions (voir *Rechargez les extensions.*, page 111).

Pour tester la boîte de dialogue :

1. Cliquez sur la case à cocher Texte rouge.

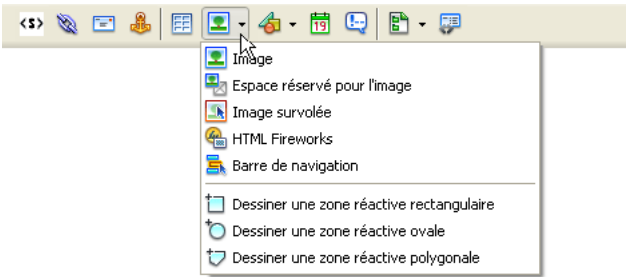


2. Cliquez sur OK pour exécuter la fonction `objectTag()`, qui barre le texte :



Création d'un menu déroulant dans la barre Insérer

L'organisation de la barre Insérer de Dreamweaver se présente sous une nouvelle forme et prend désormais en charge les menus déroulants regroupant des sous-ensembles d'objets, comme indiqué dans la figure suivante.



L'exemple ci-dessous crée une nouvelle catégorie, Editorial, dans la barre Insérer, puis ajoute un menu déroulant à cette catégorie. Le menu déroulant regroupe l'objet Strikethrough, ainsi que l'objet Blue Text créé dans la procédure ci-dessous. Les objets de la catégorie Editorial permettent aux utilisateurs d'insérer des commentaires d'édition dans un fichier et de barrer le contenu à supprimer ou d'afficher en bleu un nouveau contenu.

Pour organiser les fichiers :

1. Créez le dossier Configuration/Objects/Editorial dans le dossier d'installation de Dreamweaver.

2. Copiez les fichiers correspondant à l'exemple d'objet Strikethrough créé plus haut dans le dossier Editorial (Strikethrough.htm, Strikethrough.js et Strikethrough.gif).

Pour créer l'objet Blue Text :

1. Créez un fichier HTML.
2. Ajoutez le code suivant :

```
<html>
<head>
<title>Blue Text</title>
<script language="javascript">

//---------------     API FUNCTIONS    ---------------

function isDOMRequired() {
  // Return false, indicating that this object is available in Code
  view.
  return false;
}

function objectTag() {
  // Manually wrap tags around selection.
  var dom = dw.getDocumentDOM();
  if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){

    var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper
  Case", "") == 'TRUE');


    // Manually wrap tags around selection.
    if (upCaseTag){
        dom.source.wrapSelection('<FONT COLOR="#0000FF">','</FONT>');
    }else{
        dom.source.wrapSelection('<font color="#0000FF">','</font>');
        }
  }else if (dw.getFocus() == 'document'){
     dom.applyFontMarkup("color", "#0000FF");
    }

  // Just return -- don't do anything else.
  return;
}

</script>
</head>
<body>
</body>
</html>
```

3. Enregistrez le fichier sous le nom AjoutBleu.htm dans le dossier Editorial.

Vous pouvez maintenant créer une image pour l'objet Blue Text.

Pour créer l'image :

1. Créez un fichier GIF de 18 x 18 pixels, similaire à la figure ci-dessous :

2. Enregistrez l'image sous le nom AjoutBleu.gif dans le dossier Editorial.

Modifiez ensuite le fichier insertbar.xml. Ce fichier définit les objets de la barre Insérer ainsi que leur emplacement. Notez les nombreuses balises `menubutton` et leurs attributs dans les balises `category` ; ces balises `menubutton` définissent chacune des menus déroulants de la catégorie HTML. Dans les balises `menubutton`, chaque balise `button` définit l'un des éléments du menu déroulant.

Pour modifier insertbar.xml :

1. Recherchez la ligne de code suivante située au début du fichier :

```
<insertbar xmlns:MMString="http://www.macromedia.com/schemes/data/
  string/">
```

   La balise `insertbar` définit le début de chacun des éléments de la barre Insérer.

2. A la suite de cette ligne, insérez une balise `category` pour la catégorie Editorial, en lui assignant les attributs ID (unique), name et folder, puis insérez une balise de fermeture `category` comme indiqué dans l'exemple suivant :

```
<category id="DW_Insertbar_Editorial" name="Editorial"
  folder="Editorial">
</category>
```

3. Rechargez les extensions. Pour plus d'informations sur le rechargement des extensions, voir *Rechargez les extensions.*, page 111.

La catégorie Editorial apparaît dans la barre Insérer :

4. Entre les balises d'ouverture et de fermeture `category`, ajoutez le menu déroulant en insérant la balise `menubutton` ainsi que les attributs suivants, y compris un ID unique.

```
<menubutton id="DW_Insertbar_Markup" name="markup"
  image="Editorial\Strikethrough.gif" folder="Editorial">
```

Pour plus d'informations sur les attributs, voir *Attributs des balises de définition de la barre Insérer*, page 152.

5. Ajoutez les objets du nouveau menu déroulant à l'aide de la balise `button`, comme suit :

```
<button id="DW_Editorial_Strikethrough"
  image="Editorial\Strikethrough.gif"
  file="Editorial\Strikethrough.htm"/>
```

6. A la suite de l'objet Strikethrough, insérez l'objet Blue Text, comme suit :

```
<button id="DW_Blue_Text" image="Editorial\AddBlue.gif name="Blue Text"
  file="Editorial\AddBlue.htm"/>
```

REMARQUE La balise button ne possède pas de balise de fermeture séparée, elle se termine simplement par "`/>`".

7. Terminez l'insertion dans le menu déroulant en spécifiant la balise de fermeture `</menubutton>`.

Le code ci-dessous représente l'ensemble de la catégorie regroupant le menu déroulant et les deux objets :

```
<category id="DW_Insertbar_Editorial" name="Editorial" folder="Editorial">
  <menubutton id="DW_Insertbar_Markup" name="markup"
  image="Editorial\Strikethrough.gif" folder="Editorial">
    <button id="DW_Editorial_Strikethrough"
    image="Editorial\Strikethrough.gif"
  file="Editorial\Strikethrough.htm"/>
```

```
    <button id="DW_Blue_Text" image="Editorial\AddBlue.gif" name="Blue
  Text"
    file="Editorial\AddBlue.htm"/>
  </menubutton>
</category>
```

Pour tester le nouveau menu déroulant :

1. Rechargez les extensions. Pour plus d'informations sur le rechargement des extensions, voir *Rechargez les extensions.*, page 111.
2. Cliquez sur le menu Editorial.

Le menu déroulant ci-dessous s'affiche :

API des objets

Cette section décrit les fonctions de l'API des objets. Vous devez définir soit la fonction `insertObject()`, soit la fonction `objectTag()`. Pour plus d'informations sur ces fonctions, voir *insertObject()*, page 172. Les autres fonctions sont facultatives.

canInsertObject()

Disponibilité

Dreamweaver MX.

Description

Cette fonction détermine si la boîte de dialogue de l'objet doit être affichée.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne.

Exemple

Le code ci-dessous indique à Dreamweaver de vérifier la présence d'une chaîne spécifique dans un document avant de permettre à l'utilisateur d'insérer l'objet sélectionné :

```
function canInsertObject(){
   var docStr = dw.getDocumentDOM().documentElement.outerHTML;
   var patt = /hava/;
   var found = ( docStr.search(patt) != -1 );
   var insertionIsValid = true;

   if (!found){
    insertionIsValid = false;
    alert("the document must contain a 'hava' string to use this object.");}
   return insertionIsValid;}
```

displayHelp()

Description

Si vous définissez cette fonction, un bouton Aide apparaît sous les boutons OK et Annuler dans la boîte de dialogue Paramètres. Cette fonction est appelée lorsque l'utilisateur clique sur le bouton Aide.

Arguments

Néant

Valeurs renvoyées

Dreamweaver n'attend rien.

Exemple

L'exemple suivant permet d'afficher le fichier monObjetAide.htm dans un navigateur ; ce fichier explique l'utilisation de l'extension :

```
function displayHelp(){
  var myHelpFile = dw.getConfigurationPath() +
    '/ExtensionsHelp/myObjectHelp.htm';
  dw.browseDocument(myHelpFile);
}
```

isDomRequired()

Description

Cette fonction détermine si l'objet requiert un DOM valide pour fonctionner. Si cette fonction renvoie la valeur `true` ou si la fonction n'est pas définie, Dreamweaver suppose que la commande nécessite un DOM valide et synchronise les modes Code et Création du document avant de l'exécuter. La synchronisation met à jour toutes les modifications effectuées en mode Affichage de code dans le mode Création.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend la valeur `true` si une commande nécessite un DOM valide pour fonctionner et `false` dans le cas contraire.

insertObject()

Disponibilité

Dreamweaver MX.

Description

Cette fonction est obligatoire si la fonction `objectTag()` n'est pas définie. Elle est appelée si l'utilisateur clique sur OK. Elle insère du code dans le document de l'utilisateur et ferme la boîte de dialogue ou affiche un message d'erreur sans fermer la boîte de dialogue. Il s'agit d'une fonction de substitution à la fonction `objectTag()`. Elle ne suppose pas que l'utilisateur tape son texte au niveau du point d'insertion actuel et permet la validation des données lorsque l'utilisateur clique sur OK. Utilisez la fonction `insertObject()` dans l'une des conditions suivantes :

- Vous devez insérer un code à plusieurs endroits.
- Vous devez insérer un code à un endroit autre que le point d'insertion.
- Vous devez valider une entrée avant d'insérer le code.

Si aucune de ces conditions ne s'applique, utilisez la fonction `objectTag()`.

Arguments

Aucun.

Valeurs renvoyées

Dreamweaver attend soit une chaîne contenant un message d'erreur, soit une chaîne vide. Dans ce dernier cas, la boîte de dialogue de l'objet se ferme lorsque l'utilisateur clique sur OK. Sinon, Dreamweaver affiche le message d'erreur et la boîte de dialogue reste à l'écran.

Activateur

`canInsertObject()`

Exemple

Dans l'exemple ci-dessous, la fonction `insertObject()` est utilisée car une entrée doit être validée avant d'insérer le code :

```
function insertObject() {
   var theForm = document.forms[0];
   var nameVal = theForm.firstField.value;
   var passwordVal = theForm.secondField.value;
   var errMsg = "",
   var isValid = true;

   // ensure that field values are complete and valid
   if (nameVal == "" || passwordVal == "") {
     errMsg = "Complete all values or click Cancel."
   } else if (nameVal.length < 4 || passwordVal.length < 6) {
    errMsg = "Your name must be at least four characters, and your password
  at
least six";
   }
```

```
   if (!errMsg) {
     // do some document manipulation here. Exercise left to the reader
    }
   return errMsg;
}
```

objectTag()

Description

Les fonctions `objectTag()` et `insertObject()` s'excluent mutuellement : si ces deux fonctions sont définies dans un document, la fonction `objectTag()` a la priorité. Pour plus d'informations, consultez la section *insertObject()*, page 172.

Cette fonction insère une chaîne de code dans le document de l'utilisateur. Dans Dreamweaver MX, le renvoi d'une chaîne vide ou d'une valeur `null` (aussi appelée « `return`; ») signale à Dreamweaver qu'aucune action ne doit être effectuée.

REMARQUE	Dans ce cas, les modifications sont censées avoir été effectuées manuellement avant l'instruction `return` ; aussi, ne rien faire dans ce cas n'équivaut pas à cliquer sur Annuler.

Dans Dreamweaver 4, si le mode Code est actif et que la sélection porte sur une plage (par opposition au point d'insertion), cette dernière est remplacée par la chaîne renvoyée par la fonction `objectTag()`. La valeur est `true`, même si la fonction `objectTag()` renvoie une chaîne vide ou ne renvoie rien. La fonction `objectTag()` renvoie une chaîne vide ou une valeur `null` car l'utilisateur a déjà modifié manuellement le document. Dans le cas contraire, des guillemets doubles ("") suppriment souvent les modifications en remplaçant la sélection.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend la chaîne à insérer dans le document de l'utilisateur.

Exemple

L'exemple suivant de la fonction `objectTag()` insère une combinaison `OBJECT/EMBED` pour un contrôle ActiveX et un plug-in spécifiques :

```
function objectTag() {
return '\n' +
'<OBJECT CLASSID="clsid:166F1OOB-3A9R-11FB-8075444553540000" \n'¬
```

```
+ 'CODEBASE="http://www.mysite.com/product/cabs/¬
myproduct.cab#version=1,0,0,0" \n' + 'NAME="MyProductName"> \n' ¬
+ '<PARAM NAME="SRC" VALUE=""> \n' + '<EMBED SRC="" HEIGHT="" ¬
WIDTH="" NAME="MyProductName"> \n' + '</OBJECT>'
}
```

windowDimensions()

Description

Cette fonction définit les dimensions précises de la boîte de dialogue Options. Si cette fonction n'est pas définie, les dimensions de la fenêtre sont calculées automatiquement.

REMARQUE	Ne définissez cette fonction que si vous souhaitez utiliser une boîte de dialogue Options ayant des dimensions supérieures à 640 x 480 pixels.

Arguments

platform

- La valeur de l'argument *platform* est soit `"macintosh"`, soit `"windows"`, selon la plate-forme utilisée par l'utilisateur.

Valeurs renvoyées

Dreamweaver attend une chaîne au format `"widthInPixels,heightInPixels"`.

Les dimensions renvoyées sont inférieures à la taille totale de la boîte de dialogue parce qu'elles n'incluent pas la zone des boutons OK et Annuler. Si les dimensions renvoyées ne permettent pas de faire apparaître toutes les options, des barres de défilement s'affichent.

Exemple

L'exemple suivant de la fonction `windowDimensions()` règle les dimensions de la boîte de dialogue Paramètres sur 648 x 520 pixels sous Windows et sur 660 x 580 sur Macintosh :

```
function windowDimensions(platform){
  var retval = ""
  if (platform = = "windows"){
  retval = "648, 520";
  }else{
  retval = "660, 580";
  }
  return retval;
}
```

CHAPITRE 7

Commandes

7

Les commandes de Macromedia Dreamweaver 8 permettent d'exécuter presque n'importe quel type de modification dans le document actif de l'utilisateur, dans d'autres documents ouverts ou dans tout document HTML situé sur un disque local. Les commandes peuvent insérer, supprimer ou réorganiser les balises et les attributs HTML, les commentaires et le texte.

Les commandes sont des fichiers HTML. La section `BODY` d'un fichier de commandes peut contenir un formulaire HTML acceptant des options pour la commande (par exemple, le tri des éléments d'un tableau et la colonne de référence du tri). La balise `HEAD` d'un fichier de commandes contient des fonctions JavaScript qui traitent les entrées de formulaire de la section `BODY` et contrôlent les modifications apportées au document de l'utilisateur.

Le tableau ci-dessous recense les fichiers utilisés pour créer une commande.

Chemin	Fichier	Description
Configuration/Commands/	*nom_commande*.htm	Désigne l'interface utilisateur.
Configuration/Commands/	*nom_commande*.js	Contient les fonctions à exécuter.

Fonctionnement des commandes

Lorsque l'utilisateur clique sur un menu contenant une commande, les événements suivants se produisent :

1. Dreamweaver appelle la fonction `canAcceptCommand()` pour déterminer si l'option de menu doit être désactivée. Si la fonction `canAcceptCommand()` renvoie une valeur `false`, la commande est estompée dans le menu et la procédure s'arrête. Si la fonction `canAcceptCommand()` renvoie une valeur `true`, la procédure peut se poursuivre.
2. L'utilisateur sélectionne une commande dans le menu.

3. Dreamweaver appelle la fonction `receiveArguments()`, si elle est définie, dans le fichier de commandes de menu sélectionné afin de permettre à la commande de traiter tous les arguments transmis depuis l'option de menu ou la fonction `dreamweaver.runCommand()`. Pour plus d'informations sur la fonction `dreamweaver.runCommand()`, voir le *Guide des API de Dreamweaver*.
4. Dreamweaver appelle la fonction `commandButtons()`, si elle est définie, pour identifier les boutons qui figurent dans la partie droite de la boîte de dialogue Options et le code qui doit être exécuté lorsque l'utilisateur clique sur ces boutons.
5. Dreamweaver recherche la balise `FORM` dans le fichier de commandes. S'il existe un formulaire, Dreamweaver appelle la fonction `windowDimensions()`, qui redimensionne la boîte de dialogue Options contenant les éléments `BODY` du fichier. Si la fonction `windowDimensions()` n'est pas définie, Dreamweaver redimensionne automatiquement la boîte de dialogue.
6. Si la balise `BODY` du fichier de commandes contient le gestionnaire `onLoad`, Dreamweaver l'exécute (que la boîte de dialogue soit affichée ou non). Si aucune boîte de dialogue ne s'affiche, les étapes restantes ne sont pas exécutées.
7. L'utilisateur sélectionne les options de la commande. Dreamweaver exécute les gestionnaires d'événements associés aux champs du formulaire au fur et à mesure que l'utilisateur les rencontre.
8. L'utilisateur clique sur l'un des boutons définis par la fonction `commandButtons()`.
9. Dreamweaver exécute le code associé. La boîte de dialogue reste affichée jusqu'à ce que l'un des scripts de la commande appelle la fonction `window.close()`.

Ajout de commandes au menu Commandes

Dreamweaver ajoute automatiquement tous les fichiers se trouvant dans le dossier Configuration/Commandes au bas du menu Commandes. Pour empêcher qu'une commande ne s'affiche dans le menu Commandes, insérez le commentaire suivant sur la première ligne du fichier :

```
<!-- MENU-LOCATION=NONE -->
```

Lorsque cette ligne est présente, Dreamweaver ne crée pas d'élément de menu pour le fichier et vous devez appeler `dw.runCommand()` pour exécuter la commande.

Exemple de commande simple

Cette extension simple ajoute un élément dans le menu Commandes et vous permet de convertir le texte sélectionné dans votre document en majuscules ou en minuscules. Lorsque vous cliquez sur l'élément de menu, une interface à trois boutons est activée et vous permet de soumettre votre choix.

Vous créez cette extension en exécutant les étapes suivantes :

- Création de l'interface utilisateur
- Ecriture du code JavaScript
- Test de l'extension

Dans cet exemple, deux fichiers sont créés dans le dossier Commands : Change Case.htm, qui renferme l'interface utilisateur, et Change Case.js, qui renferme le code JavaScript. Le cas échéant, vous pouvez créer uniquement le fichier Change Case.htm et insérer le code JavaScript dans la section `HEAD`.

Création de l'interface utilisateur

L'interface utilisateur est un formulaire qui contient deux boutons radio, permettant à l'utilisateur de sélectionner majuscules ou minuscules.

Pour créer l'interface utilisateur :

1. Créez un document vierge.
2. Ajoutez le code suivant dans le fichier pour créer le formulaire :

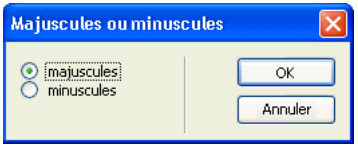
```
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine 5.0//
  dialog">
<HTML>
<HEAD>
<!-- Copyright 2001-2002 Macromedia, Inc. Tous droits réservés. -->
<Title>Make Uppercase or Lowercase</Title>
<SCRIPT SRC="Change Selection Case.js"></SCRIPT>

</HEAD>
<BODY>
 <form name="uorl">
  <table border="0">
    <!--DWLayoutTable-->
    <tr>
      <td valign="top" nowrap> <p>
          <label>
          <input type="radio" name="RadioGroup1" value="uppercase"
  checked>
          Uppercase</label>
```

```
            <br>
            <label>
            <input type="radio" name="RadioGroup1" value="lowercase">
            Lowercase</label>
          </p></td>
      </tr>
    </table>
    </div>
</form>
</BODY>
</HTML>
```

3. Enregistrez le fichier sous le nom Change Case.htm dans le dossier Configuration/Commands.

Le contenu de la balise `Title`, `Majuscules et minuscules`, s'affiche dans la barre en haut de la boîte de dialogue. Dans le formulaire, un tableau composé de deux cellules contrôle la mise en forme des éléments. Les cellules contiennent deux boutons radio : Majuscules et Minuscules. Le bouton Majuscules comporte l'attribut `checked`, ce qui en fait la sélection par défaut et confirme que l'utilisateur doit sélectionner un des deux boutons ou annuler la commande.

Le formulaire doit s'apparenter à l'image suivante.



La fonction commandButtons() insère les boutons OK et Annuler qui permettent à l'utilisateur de valider le choix ou d'annuler l'opération.

Ecriture du code JavaScript

L'exemple suivant est composé de deux fonctions API d'extension, `canAcceptCommand()` et `commandButtons()`, appelées par Dreamweaver, et d'une fonction définie par l'utilisateur, `changeCase()`, appelée depuis la fonction `commandButtons()`.

Dans cet exemple, vous allez rédiger du code JavaScript pour exécuter les tâches suivantes :

- Déterminer si la commande doit être activée ou estompée.
- Association de fonctions aux boutons OK et Annuler
- Permettre à l'utilisateur de choisir entre majuscules et minuscules

Déterminer si la commande doit être activée ou estompée.

La première tâche lors de la création d'une commande est de déterminer les cas où l'élément doit être actif ou non. Lorsque l'utilisateur clique sur le menu Commands, Dreamweaver appelle la fonction `canAcceptCommand()` pour chaque élément de menu afin de déterminer si celui-ci doit être activé. Si `canAcceptCommand()` renvoie la valeur `true`, Dreamweaver affiche le texte d'élément de menu comme actif ou activé. Si `canAcceptCommand()` renvoie la valeur `false`, Dreamweaver estompe l'élément de menu. Dans l'exemple suivant, l'élément de menu est actif lorsque l'utilisateur a sélectionné du texte dans le document.

Pour déterminer si la commande doit être activée ou estompée :

1. Créez un document vierge.
2. Ajoutez le code suivant :

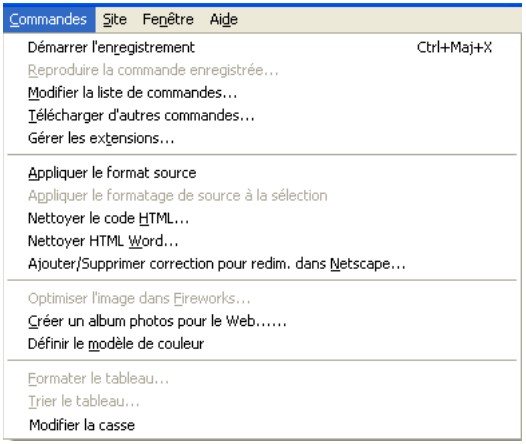
```
function canAcceptCommand(){
  var theDOM = dw.getDocumentDOM(); // Get the DOM of the current
  document
  var theSel = theDOM.getSelection(); // Get start and end of selection
  var theSelNode = theDOM.getSelectedNode(); // Get the selected node
  var theChildren = theSelNode.childNodes; // Get children of selected
  node
  return (theSel[0] != theSel[1] && (theSelNode.nodeType ==
  Node.TEXT_NODE¬
    || theSelNode.hasChildNodes() && (theChildren[0].nodeType == ¬
    Node.TEXT_NODE)));
}
```

3. Enregistrez le fichier sous le nom Change Case.js dans le dossier Configuration/Commands.

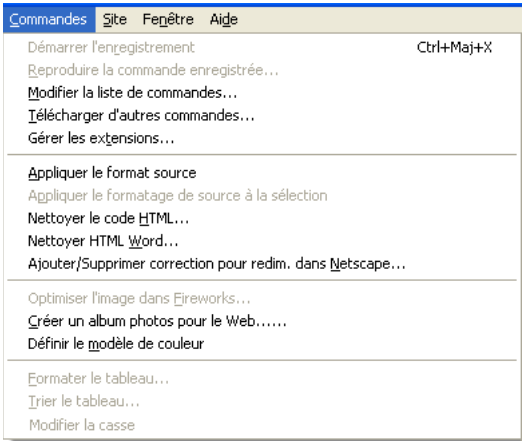
Les premières lignes de la fonction `canAcceptCommand()` extraient le texte sélectionné en récupérant le DOM du document de l'utilisateur. La fonction `getSelection()` est alors appelée sur l'objet de document. Ensuite, la fonction récupère le nœud qui contient le texte sélectionné, suivi par ses enfants, comme indiqué dans le code suivant. Enfin, la dernière ligne vérifie si la sélection (ou son premier enfant) est constituée de texte et envoie en retour la valeur `true` ou `false`.

La première partie de l'instruction `return` (`theSel[0] != theSel[1]`) vérifie si l'utilisateur a effectué une sélection dans le document. La variable `theSel` est un tableau à deux entrées maintenant les décalages de début et de fin de la sélection dans le document. Si les deux valeurs ne sont pas égales, du contenu a été sélectionné. Si les valeurs des deux entrées sont égales, il y a juste un point d'insertion. Rien n'a été sélectionné.

La partie suivante de l'instruction `return` (`&& (theSelNode.nodeType == Node.TEXT_NODE`) vérifie que le nœud sélectionné est constitué de texte. Si la réponse est positive, la fonction `canAcceptCommand()` renvoie la valeur `true`. Si le type de nœud n'est pas constitué de texte, le test se poursuit afin de vérifier si le nœud a des enfants (`|| theSelNode.hasChildNodes()`) et si le type du premier nœud enfant est constitué de texte (`&& (theChildren[0].nodeType == Node.TEXT_NODE)`). Si les deux conditions sont réunies (true), `canAcceptCommand()` renvoie la valeur `true` et Dreamweaver active l'élément de menu en bas du menu Commandes, comme indiqué ci-dessous :



Dans le cas contraire, `canAcceptCommand()` renvoie la valeur `false` et Dreamweaver estompe l'élément, comme indiqué ci-dessous :



Association de fonctions aux boutons OK et Annuler

Lorsque l'utilisateur clique sur le bouton OK ou Annuler, l'extension doit exécuter l'action appropriée. Pour ce faire, vous spécifiez la fonction JavaScript associée à un clic sur l'un des deux boutons.

Pour associer des fonctions aux boutons OK et Annuler :

1. Ouvrez le fichier Change Case.js dans le dossier Configuration/Commands.
2. En fin de fichier, insérez le code suivant :

```
function commandButtons() {
  return new Array("OK", "changeCase()", "Cancel", "window.close()");
}
```

3. Enregistrez le fichier.

La fonction `commandButtons()` entraîne l'affichage des boutons OK et Annuler par Dreamweaver et indique la marche à suivre pour Dreamweaver lorsque l'utilisateur clique sur ces boutons. La fonction `commandButtons()` indique à Dreamweaver qu'il faut appeler `changeCase()` lorsque l'utilisateur clique sur OK et appeler `window.close()` lorsque l'utilisateur clique sur Annuler.

Permettre à l'utilisateur de choisir entre majuscules et minuscules

Lorsque l'utilisateur clique sur un élément de menu, l'extension nécessite un mécanisme permettant à l'utilisateur de choisir entre majuscules et minuscules. L'interface utilisateur fournit ce mécanisme en définissant deux boutons radio permettant d'offrir ce choix à l'utilisateur.

Pour permettre à l'utilisateur de choisir entre majuscules et minuscules :

1. Ouvrez le fichier Change Case.js.
2. En fin de fichier, insérez le code suivant :

```
function changeCase() {
  var uorl;

  // Check whether user requested uppercase or lowercase.
  if (document.forms[0].elements[0].checked)
    uorl = 'u';
  else
    uorl = 'l';

  // Get the DOM.
  var theDOM = dw.getDocumentDOM();
```

```
    // Get the outerHTML of the HTML tag (the
    // entire contents of the document).
    var theDocEl = theDOM.documentElement;
    var theWholeDoc = theDocEl.outerHTML;

    // Get the node that contains the selection.
    var theSelNode = theDOM.getSelectedNode();

    // Get the children of the selected node.
    var theChildren = theSelNode.childNodes;
    var i = 0;
    if (theSelNode.hasChildNodes()){
      while (i < theChildren.length){
        if (theChildren[i].nodeType == Node.TEXT_NODE){
          var selText = theChildren[i].data;
          var theSel = theDOM.nodeToOffsets(theChildren[0]);
          break;
        }
        ++i;
      }
    }
    else {
    // Get the offsets of the selection
    var theSel = theDOM.getSelection();
    // Extract the selection
    var selText = theWholeDoc.substring(theSel[0],theSel[1]);
  }
  if (uorl == 'u'){
      theDocEl.outerHTML = theWholeDoc.substring(0,theSel[0]) +
      selText.toUpperCase() + theWholeDoc.substring(theSel[1]);
    }
    else {
      theDocEl.outerHTML = theWholeDoc.substring(0,theSel[0]) +
      selText.toLowerCase() + theWholeDoc.substring(theSel[1]);
    }
  // Set the selection back to where it was when you started
    theDOM.setSelection(theSel[0],theSel[1]);
    window.close(); // close extension UI
  }
```

3. Enregistrez le fichier sous le nom Change Case.js dans le dossier Configuration/Commands.

La fonction `changeCase()` est une fonction définie par l'utilisateur qui est appelée par la fonction `commandButtons()` lorsque l'utilisateur clique sur OK. Cette fonction modifie la casse du texte sélectionné (passage en majuscules ou minuscules). L'interface utilisateur reposant sur des boutons radio, une sélection sera obligatoirement effectuée ; il n'est pas nécessaire de tester le cas où aucun choix n'est effectué par l'utilisateur.

La fonction `changeCase()` teste en premier lieu la propriété `document.forms[0].elements[0].checked`. La propriété `document.forms[0].elements[0]` se rapporte au premier élément du premier formulaire de l'objet de document actuel, à savoir l'interface utilisateur de l'extension. La propriété `checked` prend la valeur `true` si l'élément est activé. Dans le cas contraire, il prend la valeur `false`. Dans l'interface, `elements[0]` se rapporte au premier bouton radio, à savoir le bouton Majuscules. Un des boutons radio étant forcément sélectionné lorsque l'utilisateur clique sur OK, le code suppose que, si le choix n'est pas Majuscules, c'est Minuscules qui a été choisi. La fonction règle la variable `uorl` sur "u" ou "l" pour stocker la réponse de l'utilisateur.

Le code restant dans la fonction récupère le texte sélectionné, le convertit dans la casse sélectionnée et le copie de nouveau à sa place dans le document.

Pour extraire le texte sélectionné pour le document de l'utilisateur, la fonction récupère le DOM. Elle récupère ensuite l'élément racine du document, à savoir la balise `html`. Le document entier est enfin extrait dans la variable `theWholeDoc`.

Ensuite, `changeCase()` appelle la fonction `getSelectedNode()` pour modifier le nœud contenant le texte sélectionné. Il récupère également les nœuds enfants (`theSelNode.childNodes`) au cas où la sélection serait une balise contenant du texte, comme `<b>text</b>`.

S'il y a des nœuds enfants, (`hasChildNodes()` renvoie la valeur `true`), la commande navigue de nœud en nœud à la recherche d'un nœud constitué de texte. Si un tel nœud est trouvé, le texte (`theChildren[i].data`) est stocké dans `selText` et les décalages du nœud textuel sont stockés dans `theSel`.

S'il n'y a aucun nœud enfant, la commande appelle la fonction `getSelection()` et stocke les décalages de début et de fin de sélection dans `theSel`. Il extrait ensuite la chaîne entre ces deux décalages et la conserve dans `selText`.

La fonction vérifie alors la variable `uorl` pour déterminer si l'utilisateur a sélectionné les majuscules. Si tel est le cas, la fonction écrit à nouveau le code HTML dans le document en sections : dans un premier temps, du début du document au début de la sélection ; ensuite, le texte sélectionné, converti en majuscules (`selText.toUppercase()`) et enfin, de la fin du texte sélectionné à la fin du document.

Si l'utilisateur sélectionne minuscules, la fonction effectue la même opération mais appelle `selText.toLowerCase()` pour convertir le texte sélectionné en minuscules.

Finalement, `changeCase()` réinitialise la sélection et appelle `window.close()` pour fermer l'interface utilisateur.

Test de l'extension

Une fois les fichiers placés dans le dossier Commands, vous pouvez tester l'extension.

Pour tester l'extension :

1. Redémarrez Dreamweaver ou rechargez les extensions. Pour plus d'informations sur le rechargement des extensions, voir *Rechargez les extensions.*, page 111.

   L'entrée Modifier la casse doit maintenant être affichée dans le menu Commandes.
2. Tapez du texte dans un document.
3. Sélectionnez le texte.

REMARQUE	La commande Modifier la casse est estompée tant que vous n'avez pas sélectionné de texte dans le document.

4. Sélectionnez Modifier la casse dans le menu Commandes.

   Le texte change de casse.

API des commandes

Les fonctions personnalisées de l'API des commandes de menu ne sont pas obligatoires.

canAcceptCommand()

Description

Cette fonction indique si la commande est adaptée à la sélection en cours.

REMARQUE	Ne définissez la fonction `canAcceptCommand()` que s'il existe au moins un cas où elle renvoie la valeur `false`. Si la fonction n'est pas définie, la commande est considérée comme appropriée. Ce processus entraîne des gains de temps et de performances.

Arguments

Néant

Valeurs renvoyées

Dreamweaver s'attend à recevoir la valeur `true` si la commande est appropriée. Si la valeur est `false`, la commande apparaît estompée dans le menu.

Exemple

Dans l'exemple suivant, la fonction `canAcceptCommand()` rend la commande disponible uniquement lorsque la sélection correspond à un tableau :

```
function canAcceptCommand(){
  var retval=false;
  var selObj=dw.getDocumentDOM.getSelectedNode();
  return (selObj.nodeType == Node.ELEMENT_NODE && ¬
  selObj.tagName=="TABLE");{
    retval=true;
}
return retval;
}
```

commandButtons()

Description

Cette fonction définit les boutons devant figurer dans la partie droite de la boîte de dialogue Options et leur comportement lorsque l'utilisateur clique dessus. Si cette fonction n'est pas définie, aucun bouton n'apparaît et la section `BODY` du fichier de commandes s'étend de façon à remplir la totalité de la boîte de dialogue.

Arguments

Néant

Valeurs renvoyées

Dreamweaver renvoie un tableau contenant un nombre pair d'éléments. Le premier élément est une chaîne contenant le libellé du premier bouton. Le deuxième élément est une chaîne de code JavaScript définissant le comportement du premier bouton lorsque l'utilisateur clique dessus. Les autres éléments définissent les boutons supplémentaires de la même manière.

Exemple

L'instance suivante de `commandButtons()` définit trois boutons : OK, Annuler et Aide :

```
function commandButtons(){
  return new Array("OK" , "doCommand()" , "Cancel" , ¬
  "window.close()" , "Help" , "showHelp()");
}
```

isDomRequired()

Description

Cette fonction détermine si la commande requiert un DOM valide pour fonctionner. Si cette fonction renvoie une valeur `true` ou si la fonction n'est pas définie, Dreamweaver suppose que la commande nécessite un DOM valide et synchronise les modes Affichage de code et Création du document avant de l'exécuter. La synchronisation entraîne la mise à jour de toutes les modifications effectuées en mode Affichage de code dans le mode Création.

Arguments

Néant

Valeurs renvoyées

Dreamweaver renvoie la valeur `true` si une commande nécessite un DOM valide pour fonctionner, sinon `false` .

receiveArguments()

Description

Cette fonction traite tous les arguments provenant d'un élément de menu ou de la fonction `dw.runCommand()`.

Arguments

`{arg1}, {arg2},...{argN}`

- Si l'attribut `arguments` est défini pour une balise `menuitem`, la valeur de l'attribut est transmise à la fonction `receiveArguments()` sous forme d'un ou plusieurs arguments. Les arguments peuvent également être transmis à une commande par la fonction `dw.runCommand()`.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

windowDimensions()

Description

Cette fonction définit les dimensions de la boîte de dialogue des paramètres. Si cette fonction n'est pas définie, les dimensions de la fenêtre sont calculées automatiquement.

REMARQUE	Ne définissez cette fonction que si vous souhaitez utiliser une boîte de dialogue Options ayant des dimensions supérieures à 640 x 480 pixels.

Arguments

platform

- La valeur de l'argument *platform* est soit `"macintosh"`, soit `"windows"`, selon la plate-forme utilisée par l'utilisateur.

Valeurs renvoyées

Dreamweaver renvoie une chaîne au format `"widthInPixels,heightInPixels"`.

Les dimensions renvoyées sont inférieures à la taille totale de la boîte de dialogue parce qu'elles n'incluent pas la zone des boutons OK et Annuler. Si les dimensions renvoyées ne permettent pas de faire apparaître toutes les options, des barres de défilement s'affichent.

Exemple

Dans l'exemple suivant, la fonction `windowDimensions()` définit les dimensions de la boîte de dialogue des paramètres à 648 x 520 pixels :

```
function windowDimensions(){
  return "648,520";
}
```

CHAPITRE 8

Menus et commandes de menu

8

Macromedia Dreamweaver 8 crée les menus à partir de la structure définie dans le fichier menus.xml du dossier Configuration/Menus de Dreamweaver. Vous pouvez réorganiser, renommer ou supprimer des commandes de menu en modifiant le fichier menus.xml. Vous pouvez également ajouter, modifier ou supprimer des raccourcis clavier associés aux commandes de menu. Cependant, il est généralement plus simple de passer à cet effet par l'Editeur de raccourcis clavier (voir l'aide de Dreamweaver). Les modifications effectuées dans les menus Dreamweaver ne sont prises en compte qu'après le redémarrage de Dreamweaver ou le rechargement des extensions.

Sur un système d'exploitation multiutilisateur, lorsque vous effectuez des modifications dans Dreamweaver qui affectent le fichier menus.xml (par exemple, si vous modifiez les raccourcis clavier via l'Editeur de raccourcis clavier), Dreamweaver crée un nouveau fichier menus.xml dans votre dossier de configuration utilisateur. Pour personnaliser le fichier menus.xml dans un système d'exploitation multiutilisateur, modifiez la copie du fichier qui se trouve dans votre dossier de configuration utilisateur ou copiez le fichier menus.xml principal dans ce dossier, si Dreamweaver ne l'a pas déjà fait. Pour plus d'informations, consultez la section *Dossiers de configuration multiutilisateur*, page 111.

Si vous ouvrez le fichier menus.xml dans un éditeur XML, il est possible que des messages d'erreur concernant les esperluettes (&) du fichier menus.xml s'affichent. Nous vous recommandons de modifier le fichier menus.xml dans un éditeur de texte ; évitez de le modifier dans Dreamweaver. Pour toutes informations de base sur XML, voir l'aide de Dreamweaver.

REMARQUE

Avant de modifier le fichier menus.xml ou tout autre fichier de configuration Dreamweaver, effectuez une copie de sauvegarde. En effet, en cas d'erreur, la seule manière de restaurer un ensemble de menus consiste à remplacer le fichier menus.xml. Si vous avez oublié d'effectuer une sauvegarde, vous pouvez utiliser celle du fichier menus.xml par défaut qui se trouve dans le dossier Configuration. Cette copie se nomme menus.bak. Pour restaurer l'ensemble de menus par défaut, remplacez le fichier menus.xml par une copie du fichier menus.bak.

A propos du fichier menus.xml

Le fichier menus.xml contient une liste structurée des barres de menus, menus, commandes de menu, séparateurs, listes de raccourcis et raccourcis clavier. Ces éléments sont décrits par des balises XML que vous pouvez modifier dans un éditeur de texte.

REMARQUE	Procédez avec prudence lorsque vous modifiez les menus. Dreamweaver ignore les menus ou commandes de menu comportant des erreurs de syntaxe XML.

Une barre de menus (indiquée par des balises `menubar` d'ouverture et de fermeture) est un menu ou un ensemble de menus distinct. Par exemple, il est possible que vous ayez une barre de menus principale, une barre de menus fenêtre Site distincte (apparaissant sous Windows mais pas sur Macintosh) et une barre de menus pour chaque menu contextuel. Chaque barre de menus comporte un ou plusieurs menus indiqués par des balises menu. Chacun de ces menus comporte une ou plusieurs commandes de menu, indiquées par des balises `menuitem` et définies par divers attributs. Un menu peut également comporter des séparateurs (indiqués par des balises separator) ainsi que des sous-menus.

Outre les raccourcis clavier associés aux commandes de menus, Dreamweaver compte un grand nombre d'autres raccourcis clavier, notamment des raccourcis secondaires et des raccourcis activés dans des contextes particuliers. Par exemple, le raccourci Ctrl+Y (Windows) ou Commande+Y (Macintosh) correspond à la commande Rétablir, mais il est également possible d'utiliser le raccourci Ctrl+Maj+Z ou Commande+Maj+Z. Ces raccourcis secondaires (ainsi que d'autres raccourcis ne pouvant pas être représentés dans les balises des commandes de menu) sont définis dans les listes de raccourcis du fichier menus.xml. Ces listes de raccourcis (indiquées par des balises `shortcutlist`) comportent un ou plusieurs raccourcis indiqués par des balises shortcut.

Les sections ci-dessous décrivent la syntaxe des balises menus.xml. Les attributs facultatifs sont signalés dans la liste des attributs par des accolades ({}). Les attributs qui ne sont pas signalés ainsi sont considérés comme obligatoires.

<menubar>

Description

Fournit des informations sur une barre de menus de la structure de menus de Dreamweaver.

Attributs

`name, {app}, id, {platform}`

- `name` Nom de la barre de menus. Bien que l'attribut `name` soit obligatoire, vous pouvez lui assigner la valeur "".
- `app` Nom de l'application dans laquelle la barre de menus est disponible. Non utilisé actuellement.
- `id` ID de menu de la barre de menus. Les ID de menu du fichier menus.xml doivent tous être uniques.
- `platform` Indique que la barre de menus ne doit apparaître que sur la plate-forme spécifiée. Valeurs gérées : `"win"` et `"mac"`.

Contenu

Cette balise doit comporter une ou plusieurs balises `menu`.

Contenant

Néant

Exemple

La barre de menus principale (fenêtre de document) est définie par la balise menubar suivante :

```
<menubar name="Main Window" id="DWMainWindow">
<!-- balises menu ici -->
</menubar>
```

<menu>

Description

Fournit des informations sur un menu ou un sous-menu de la structure de menus de Dreamweaver.

Attributs

`name, {app}, id, {platform}, {showIf}`

- `name` Nom du menu tel qu'il apparaît dans la barre de menus. Pour définir la clé d'accès (mnémonique) du menu sous Windows, insérez un trait de soulignement (_) avant la lettre d'accès. Sur Macintosh, le trait de soulignement est supprimé automatiquement.
- `app` Nom de l'application dans laquelle le menu est disponible. Non utilisé actuellement.
- `id` ID du menu. Les ID de menu du fichier doivent tous être uniques.
- `platform` Indique que le menu ne doit apparaître que sur la plate-forme spécifiée. Valeurs gérées : `"win"` et `"mac"`.

- `showIf` Indique que le menu ne doit apparaître que si l'activateur Dreamweaver spécifié a la valeur `true`. Les activateurs possibles sont : `_SERVERMODEL_ASP`, `_SERVERMODEL_ASPNET`, `_SERVERMODEL_JSP`, `_SERVERMODEL_CFML` (pour toutes les versions de Macromedia ColdFusion), `_SERVERMODEL_CFML_UD4` (réservé à UltraDev version 4 de ColdFusion), `_SERVERMODEL_PHP`, `_FILE_TEMPLATE`, `_VIEW_CODE`, `_VIEW_DESIGN`, `_VIEW_LAYOUT`, `_VIEW_EXPANDED_TABLES` et `_VIEW_STANDARD`. Vous pouvez spécifier plusieurs activateurs en les séparant par des virgules (chaque virgule signifie ET). Vous pouvez utiliser des points d'exclamation (`"!"`) pour indiquer NE PAS. Ainsi, si vous souhaitez que le menu n'apparaisse qu'en mode Code dans une page ASP, définissez les attributs de la façon suivante : `showIf="_VIEW_CODE, _SERVERMODEL_ASP"`.

Contenu

Cette balise peut contenir une ou plusieurs balises `menuitem` et `separator`. Elle peut également contenir d'autres balises `menu` (pour créer des sous-menus) ainsi que des balises HTML comment standard.

Contenant

Cette balise doit être imbriquée dans une balise `menubar`.

Exemple

```
<menu name="_File" id="DWMenu_File">
  <!-- balises menuitem, separator, menu et comment ici -->
</menu>
```

<menuitem>

Description

Définit une commande de menu Dreamweaver.

Attributs

```
name, id, {app}, {key}, {platform}, {enabled}, {arguments}, {command},
  {file}, {checked}, {dynamic}, {isdomrequired}, {showIf}
```

- `name` Nom de la commande intégrée au menu. Un trait de soulignement indique que la lettre suivante est définie comme clé d'accès (mnémonique) de la commande (sous Windows uniquement).
- `id` Permet à Dreamweaver d'identifier l'élément. Cet ID doit être unique dans l'ensemble de la structure de menus. Si vous ajoutez de nouvelles commandes de menu au fichier menus.xml, indiquez le nom de votre entreprise ou toute autre chaîne unique comme préfixe aux ID des commandes de menu.

- `app` Nom de l'application dans laquelle figure la commande de menu. Non utilisé actuellement.
- `key` Raccourci clavier de la commande, le cas échéant. Utilisez les chaînes suivantes pour spécifier les touches de modification :
  - `Cmd` désigne la touche Ctrl (Windows) ou Commande (Macintosh).
  - `Alt` et `Opt` désignent la touche Alt (Windows) ou la touche Option (Macintosh). Ces chaînes sont interchangeables.
  - `Shift` désigne la touche Maj, quelle que soit la plate-forme utilisée.
  - `Ctrl` désigne la touche Ctrl, quelle que soit la plate-forme utilisée.
  - Lorsque plusieurs touches de modification sont assignées à un raccourci, elles sont séparées par un signe plus (+). Par exemple, la chaîne `Cmd+Opt+5` dans l'attribut `key` signifie que l'utilisateur doit appuyer sur Ctrl+Alt+5 (Windows) ou Commande+Option+5 (Macintosh) afin d'exécuter la commande de menu.
  - Les touches spéciales sont indiquées par leur nom : `F1` à `F12`, `Pg. Suiv.`, `Pg. Préc.`, `Orig.`, `Fin`, `Inser`, `Suppr`, `Tab`, `Echap`, `Ret. Arr.` et `Espace`. Vous pouvez également assigner les touches de modification aux touches spéciales.
- `platform` Indique la plate-forme sur laquelle apparaît l'élément. Les valeurs gérées sont les suivantes `"win"` (Windows) ou `"mac"` (Macintosh). Si vous ne spécifiez aucun attribut `platform`, la commande de menu apparaît sur les deux plates-formes. Si vous souhaitez qu'une commande de menu ait un comportement spécifique en fonction de la plate-forme, spécifiez deux commandes de menu avec un nom identique (mais des ID distincts) : la première contient l'attribut `platform="win"`, la seconde l'attribut `platform="mac"`.
- `enabled` Fournit un code JavaScript (généralement un appel de fonction JavaScript) déterminant si la commande de menu est actuellement activée. Si la fonction renvoie la valeur `false`, la commande de menu est estompée. La valeur par défaut est `"true"`, mais il est préférable de spécifier systématiquement une valeur même si la valeur est `"true"`.
- `arguments` Fournit des arguments que Dreamweaver transmet au code du fichier JavaScript indiqué dans l'attribut `file`. Ces arguments doivent être entourés d'apostrophes ('), à l'intérieur de la chaîne de valeur d'un attribut délimitée par des guillemets doubles.
- `command` Spécifie une expression JavaScript qui est exécutée lorsque l'utilisateur sélectionne l'élément dans le menu. Pour spécifier un code JavaScript complexe, utilisez un fichier JavaScript (indiqué dans l'attribut `file`). Vous devez spécifier l'attribut `file` ou `command` pour chaque commande de menu.

- `file` Nom du fichier HTML contenant le code JavaScript qui contrôle la commande de menu. Spécifiez le chemin du fichier par rapport au dossier Configuration (par exemple, pour la commande de menu Aide > Bienvenue, spécifiez l'attribut `file="Commands/Welcome.htm"`). L'attribut `file` remplace les attributs `command`, `enabled` et `checked`. Vous devez spécifier l'attribut `file` ou `command` pour chaque commande de menu. Pour plus d'informations sur la création d'un fichier de commandes par l'intermédiaire du panneau Historique, voir l'aide de Dreamweaver. Pour plus d'informations sur la rédaction des commandes JavaScript, voir Chapitre 7, “Commandes,” on page 177.
- `checked` Expression JavaScript déterminant si une coche doit s'afficher en regard de la commande de menu dans le menu ; si la valeur de cette expression est true, la coche s'affiche.
- `dynamic` Si cet attribut est spécifié, il indique que la commande de menu doit être déterminée dynamiquement par un fichier HTML ; ce dernier contient un code JavaScript définissant le texte et l'état de la commande de menu. Si vous spécifiez l'attribut `dynamic` pour une balise, vous devez également spécifier l'attribut `file`.
- `isdomrequired` Indique s'il est nécessaire de synchroniser les modes Création et Code avant d'exécuter le code associé à cette commande de menu. Les valeurs admises sont : `"true"` (par défaut) et `"false"`. Si vous définissez la valeur `"false"`, les modifications apportées au fichier par la commande de menu n'utilisent pas le DOM Dreamweaver. Pour plus d'informations sur le DOM, voir Chapitre 5, “Modèle d'objet de document (DOM) Dreamweaver,” on page 133.
- `showIf` Indique que la commande de menu ne doit apparaître que si la valeur de l'activateur Dreamweaver spécifié est `true`. Les activateurs possibles sont : `_SERVERMODEL_ASP`, `_SERVERMODEL_ASPNET`, `_SERVERMODEL_JSP`, `_SERVERMODEL_CFML` (pour toutes les versions de ColdFusion), `_SERVERMODEL_CFML_UD4` (applicable uniquement à UltraDev version 4 de ColdFusion), `_SERVERMODEL_PHP`, `_FILE_TEMPLATE`, `_VIEW_CODE`, `_VIEW_DESIGN`, `_VIEW_LAYOUT`, `_VIEW_EXPANDED_TABLES` et `_VIEW_STANDARD`. Vous pouvez spécifier plusieurs activateurs en les séparant par des virgules (chaque virgule signifie ET). Vous pouvez utiliser des points d'exclamation (`"!"`) à titre d'exclusion. Ainsi, si vous souhaitez que la commande de menu apparaisse en mode Code, mais pas sur une page ColdFusion, définissez l'attribut de la façon suivante : `showIf="_VIEW_CODE, !_SERVERMODEL_CFML"`.

Contenu

Aucun (balise vide).

Contenant

Cette balise doit être imbriquée dans une balise `menu`.

Exemple

```
<menuitem name="_New" key="Cmd+N" enabled="true"
  command="dw.createDocument()" id="DWMenu_File_New" />
```

<separator>

Description

Indique qu'un séparateur doit apparaître à l'emplacement correspondant du menu.

Attributs

`{app}`

- `app` Nom de l'application dans laquelle le séparateur apparaît. Non utilisé actuellement.

Contenu

Aucun (balise vide).

Contenant

Cette balise doit être imbriquée dans une balise `menu`.

Exemple

```
<separator />
```

<shortcutlist>

Description

Spécifie une liste de raccourcis dans le fichier menus.xml.

Attributs

`{app}, id, {platform}`

- `app` Nom de l'application dans laquelle la liste de raccourcis est disponible. Non utilisé actuellement.
- `id` ID de la liste de raccourcis. Cet ID doit être identique à celui de la barre de menus (ou menu contextuel) de Dreamweaver à laquelle les raccourcis sont associés. Les valeurs admises sont les suivantes : `"DWMainWindow", "DWMainSite", "DWTimelineContext"` et `"DWHTMLContext"`.
- `platform` Indique que la liste de raccourcis ne doit apparaître que sur la plate-forme spécifiée. Les valeurs possibles sont : `"win"` et `"mac"`.

Contenu

Une ou plusieurs balises `shortcut` peuvent être imbriquées dans cette balise. Celle-ci peut également contenir une ou plusieurs balises comment (dont la syntaxe est identique à celle des balises comment HTML).

Contenant

Néant

Exemple

```
<shortcutlist id="DWMainWindow">
<!-- balises shortcut et comment ici -->
</shortcutlist>
```

<shortcut>

Description

Spécifie un raccourci clavier dans le fichier menus.xml.

Attributs

`key, {app}, {platform}, {file}, {arguments}, {command}, id, {name}`

- `key` Combinaison de touches permettant d'activer le raccourci clavier. Pour plus d'informations sur la syntaxe, voir <menuitem>.
- `app` Nom de l'application dans laquelle le raccourci est disponible. Non utilisé actuellement.
- `platform` Indique que le raccourci ne fonctionne que sur la plate-forme spécifiée. Les valeurs possibles sont : "`win`" et "`mac`". Si vous ne définissez pas cet attribut, le raccourci fonctionne sur les deux plates-formes.
- `file` Chemin d'accès au fichier contenant le code JavaScript exécuté par Dreamweaver lorsque le raccourci clavier est utilisé. L'attribut `file` prime sur l'attribut `command`. Vous devez spécifier l'attribut `file` ou `command` pour chaque raccourci.
- `arguments` Fournit des arguments que Dreamweaver transmet au code dans le fichier JavaScript indiqué dans l'attribut `file`. Ces arguments doivent être insérés entre des apostrophes ('), à l'intérieur de la chaîne des valeurs d'attribut délimitée par les guillemets (").
- `command` Code JavaScript exécuté par Dreamweaver lorsque le raccourci clavier est utilisé. Spécifiez l'attribut `file` or `command` pour chaque raccourci.
- `id` Identificateur unique d'un raccourci.

- `name` Nom de la commande exécutée par le raccourci clavier, basé sur le modèle des noms de commande de menu. Par exemple, l'attribut `name` du raccourci F12 correspond à `"Aperçu dans le navigateur principal "`.

Contenu

Aucun (balise vide).

Contenant

Cette balise doit être imbriquée dans une balise `shortcutlist`.

Exemple

```
<shortcut key="Cmd+Shift+Z" file="Menus/MM/Edit_Clipboard.htm"
arguments="'redo'" id="DWShortcuts_Edit_Redo" />
```

<tool>

Description

Cette balise représente un outil unique et contient tous les raccourcis associés à ce dernier sous la forme de sous-balises, stockées dans le fichier menus.xml.

Attributs

`{name}, id`

- `name` Version localisée du nom de l'outil.
- `id` Identificateur d'outil interne qui désigne l'outil auquel s'appliquent les raccourcis.

Contenu

Cette balise peut contenir une ou plusieurs balises `activate`, `override` ou `action`.

Contenant

Cette balise doit être imbriquée dans une balise `menu`.

Exemple

```
<tool name="Hand tool" id="com.macromedia.dreamweaver.tools.hand">
  <!-- balises tool ici -->
</tool>
```

<action>

Description

Contient la combinaison de touches et le code JavaScript à exécuter lorsque l'outil est actif et que l'utilisateur compose la combinaison de touches.

Attributs

`{name}, key, command, id`

- `name` Version localisée de l'action.
- `key` Combinaison de touches permettant d'exécuter l'action. Pour plus d'informations sur la syntaxe, voir <menuitem>.
- `command` Instructions JavaScript à exécuter. Le format de cet attribut est identique à celui de l'attribut `command` de <shortcut>.
- `id` Identificateur unique de l'action.

Contenu

Aucun (balise vide).

Contenant

Cette balise doit être imbriquée dans une balise `tool`.

Exemple

```
<action name="Set magnification to 50%" key="5" command="dw.activeViewScale
  = 0.50" id ="DWTools_Zoom_50" />
```

<activate>

Description

Contient la combinaison de touches permettant d'activer l'outil.

Attributs

`{name}, key, id`

- `name` Version localisée de l'action.
- `key` Combinaison de touches permettant d'activer l'outil. Pour plus d'informations sur la syntaxe, voir <menuitem>.
- `id` Identificateur unique de l'action.

Contenu

Aucun (balise vide).

Contenant

Cette balise doit être imbriquée dans une balise `tool`.

Exemple

```
<activate name="Switch to Hand tool" key="H" id="DWTools_Hand_Active1" />
```

<override>

Description

Contient la combinaison de touches permettant d'activer temporairement l'outil. S'il exécute un autre outil modal, l'utilisateur peut appuyer sur cette touche et la maintenir enfoncée pour basculer d'un outil à l'autre.

Attributs

`{name}, key, id`

- `name` Version localisée de l'action.
- `key` Combinaison de touches permettant d'activer rapidement l'outil. Pour plus d'informations sur la syntaxe, voir <menuitem>.
- `id` Identificateur unique de l'action.

Contenu

Aucun (balise vide).

Contenant

Cette balise doit être imbriquée dans une balise `tool`.

Exemple

```
<override name="Quick switch to Hand tool" key="Space"
  id="DWTools_Hand_Override" />
```

Modification des menus et commandes de menu

En modifiant le fichier menus.xml, vous pouvez déplacer les commandes de menu au sein d'un menu ou d'un menu à un autre. Vous pouvez également ajouter des séparateurs dans les menus ou les supprimer et déplacer des menus au sein d'une barre de menus ou d'une barre de menus à une autre.

En outre, vous pouvez ajouter des éléments dans les menus contextuels ou en retirer en appliquant la même procédure que pour les autres menus.

Pour plus d'informations, consultez *A propos du fichier menus.xml*, page 192.

Pour déplacer une commande de menu :

1. Quittez Dreamweaver.
2. Effectuez une copie de sauvegarde du fichier menus.xml.
3. Ouvrez le fichier menus.xml dans un éditeur de texte comme BBEdit, HomeSite ou Wordpad (ne l'ouvrez pas dans Dreamweaver).
4. Coupez une balise `menuitem` complète, depuis la chaîne `<menuitem` au début de la chaîne jusqu'au caractère `/>` final.
5. Placez le point d'insertion à l'emplacement requis pour la commande de menu. (Cet emplacement doit se situer entre une balise `menu` et la balise `/menu` correspondante.)
6. Collez la commande de menu à son nouvel emplacement.

Pour créer un sous-menu lors du déplacement d'une commande de menu :

1. Placez le point d'insertion dans un menu (entre une balise `menu` et la balise `/menu` correspondante).
2. Insérez une nouvelle balise `menu` et une balise `/menu` dans le menu.
3. Ajoutez les nouvelles commandes de menu dans le nouveau sous-menu.

Pour insérer un séparateur entre deux commandes de menu :

- Insérez une balise `separator/` entre les deux balises `menuitem`.

Pour supprimer un séparateur existant :

- Supprimez la ligne `separator/` correspondante.

Pour déplacer un menu :

1. Quittez Dreamweaver.
2. Effectuez une copie de sauvegarde du fichier menus.xml.
3. Ouvrez le fichier menus.xml dans un éditeur de texte comme BBEdit, HomeSite ou Wordpad (ne l'ouvrez pas dans Dreamweaver).
4. Coupez un menu ainsi que son contenu, depuis la balise d'ouverture `menu` jusqu'à la balise de fermeture `/menu`.
5. Placez le point d'insertion au nouvel emplacement du menu (cet emplacement doit se situer entre une balise `menubar` et la balise `/menubar` correspondante).
6. Copiez le menu à son nouvel emplacement.

Modification du nom d'un menu ou d'une commande de menu

Pour modifier le nom d'un menu ou d'une commande de menu, il suffit de modifier le fichier menus.xml.

Pour modifier le nom d'un menu ou d'une commande de menu :

1. Quittez Dreamweaver.
2. Effectuez une copie de sauvegarde du fichier menus.xml.
3. Ouvrez le fichier menus.xml dans un éditeur de texte comme HomeSite, BBEdit ou Wordpad (ne l'ouvrez pas dans Dreamweaver).
4. Si vous modifiez le nom d'une commande de menu, localisez la balise `menuitem` adéquate, puis modifiez son attribut `name`. Si vous modifiez le nom d'un menu, localisez la balise `menu` adéquate, puis modifiez son attribut`name`. Dans tous les cas, ne modifiez pas l'attribut `id`.
5. Enregistrez, puis fermez le fichier menus.xml. Relancez Dreamweaver afin que les modifications soient prises en compte.

Modification des raccourcis clavier

Si les raccourcis clavier par défaut ne vous conviennent pas, vous pouvez les modifier, les supprimer ou en créer de nouveaux. Pour ce faire, la solution la plus simple est d'utiliser l'Editeur de raccourcis clavier (pour plus d'informations, voir l'aide de Dreamweaver). Il est également possible de modifier les raccourcis clavier directement dans le fichier menus.xml, mais les risques d'erreur sont plus élevés que si vous utilisez l'Editeur de raccourcis clavier.

Pour modifier un raccourci clavier :

1. Quittez Dreamweaver.
2. Effectuez une copie de sauvegarde du fichier menus.xml.
3. Ouvrez le fichier menus.xml dans un éditeur de texte comme BBEdit, HomeSite ou Wordpad (ne l'ouvrez pas dans Dreamweaver).
4. Consultez la matrice des raccourcis clavier (disponible à partir du centre de support de Dreamweaver) et choisissez un raccourci non utilisé ou que vous souhaitez réassigner.

   Si vous réassignez un raccourci clavier, reportez les modifications sur une copie imprimée de la matrice pour vous y référer ultérieurement.
5. Lorsque vous réassignez un raccourci clavier, localisez la commande de menu à laquelle il renvoie, puis supprimez l'attribut `key="shortcut"` de cette commande de menu.

6. Localisez la commande de menu à laquelle vous souhaitez assigner ou réassigner le raccourci clavier.
7. Si un raccourci clavier est déjà assigné à cette commande de menu, localisez l'attribut key dans la ligne. Si aucun raccourci ne lui a été assigné, ajoutez l'attribut `key=""` parmi les autres attributs au sein de la balise menuitem.
8. Entre les guillemets (") de l'attribut `key`, saisissez le nouveau raccourci clavier.

   Si vous saisissez une combinaison de touches, séparez ces dernières par un signe plus (+). Pour plus d'informations sur les modificateurs, voir la description de la balise `menuitem` dans <menuitem>.

   Si le raccourci clavier est déjà assigné à une autre commande de menu et que vous n'avez pas supprimé cette assignation, le raccourci ne s'applique qu'à la première commande à laquelle il est associé dans le fichier menus.xml.

   REMARQUE

   Vous pouvez utiliser un même raccourci clavier pour une commande de menu Windows et Macintosh.

9. Saisissez le nouveau raccourci à l'emplacement adéquat de la matrice des raccourcis clavier.

Modification des menus déroulants et des menus contextuels

Dreamweaver comporte des menus déroulants ou contextuels dans un grand nombre de panneaux et de boîtes de dialogue. Certains menus contextuels sont définis dans le fichier menus.xml. Les autres sont définis dans divers fichiers XML. Il vous est possible de créer, supprimer ou modifier des éléments de ces menus. Cependant, en règle générale, il est préférable de créer une extension afin d'effectuer ces changements.

Les menus déroulants et contextuels de Dreamweaver répertoriés ci-dessous sont définis dans des fichiers XML au moyen de balises identiques à celles du fichier menus.xml :

- Sources de données (menu déroulant plus (+) du panneau Liaisons) : définies dans les fichiers DataSources.xml situés dans les sous-dossiers du dossier DataSources.
- Comportements de serveur (menu déroulant plus (+) du panneau Comportements de serveur) : définis dans les fichiers ServerBehaviors.xml situés dans les sous-dossiers du dossier ServerBehaviors.
- Formats de serveur (menu déroulant plus (+) de la boîte de dialogue Modifier la liste de formats) : définis dans les fichiers ServerFormats.xml situés dans les sous-dossiers du dossier ServerFormats.

- Les éléments du menu déroulant formats qui permettent une liaison dans le panneau Liaisons sont définis dans les fichiers Formats.xml situés dans les sous-dossiers du dossier ServerFormats. Vous pouvez ajouter des entrées à ce menu à partir de Dreamweaver via la boîte de dialogue d'ajout de format.
- Les commandes de menu de la boîte de dialogue Editeur de la bibliothèque de balises sont définies dans le fichier TagLibraries/TagImporters/TagImporters.xml.
- Les commandes de menu des paramètres de la boîte de dialogue de génération de comportement, elle-même comprise dans le Créateur de comportements de serveur, sont définies dans le fichier Shared/Controls/String Menu/Controls.xml.
- Les commandes des menus contextuels associées aux composants ColdFusion sont définies dans le fichier Components/ColdFusion/CFCs/CFCsMenus.xml.
- Les commandes des menus contextuels associées aux sources de données ColdFusion sont définies dans le fichier Components/ColdFusion/DataSources/DataSourcesMenus.xml.
- Les commandes des menus contextuels associées aux JavaBeans sont définies dans le fichier Components/Jsp/JavaBeans/JavaBeanMenus.xml.
- Les commandes des menus contextuels associées aux divers composants de serveur sont définies dans des fichiers XML situés dans les sous-dossiers du dossier Components.

Commandes de menu

Les commandes de menu rendent les menus plus souples et plus dynamiques. A l'instar des commandes ordinaires, les commandes de menu permettent d'exécuter pratiquement n'importe quel type de modification dans le document actif, dans d'autres documents ouverts ou dans tout document HTML situé sur un disque local. L'API des commandes de menu se développe à partir de l'API des commandes ordinaires pour accomplir plusieurs tâches liées à l'affichage et à l'appel de la commande à partir du système de menus.

Les commandes de menu sont des fichiers HTML référencés dans l'attribut `file` d'une balise `menuitem` du fichier menus.xml. La section `BODY` d'un fichier de commandes de menu peut contenir un formulaire HTML acceptant des options pour la commande (par exemple, le tri des éléments d'un tableau et la colonne de référence du tri). La section `HEAD` d'un fichier de commandes de menu contient des fonctions JavaScript qui traitent les entrées de formulaire de la section `BODY` et contrôlent les modifications apportées au document de l'utilisateur.

Les commandes de menu sont stockées dans le dossier Configuration/Menus du dossier de l'application Dreamweaver.

Le tableau ci-dessous recense les fichiers utilisés pour créer une commande de menu.

Chemin	Fichier	Description
Configuration/Menus/	menus.xml	Contient une liste structurée des barres de menus, menus, commandes de menu, séparateurs, listes de raccourcis et raccourcis clavier. Modifiez ce fichier pour ajouter de nouveaux menus et commandes de menu.
Configuration/Menus/	*nom_commande*.htm	Contient les fonctions requises par la commande de menu.

REMARQUE

Si vous ajoutez des commandes de menu personnalisées dans Dreamweaver, faites-le au plus haut niveau du dossier Menus ou créez un sous-dossier. Le dossier Macromedia MM est réservé aux commandes de menu fournies avec Dreamweaver.

Modification du menu Commandes

Vous pouvez ajouter certains types de commandes au menu Commandes et les renommer sans avoir à modifier le fichier menus.xml. Pour plus d'informations sur le fichier menus.xml, voir *Modification des menus et commandes de menu*, page 201.

REMARQUE

Le mot « commande » a deux significations distinctes dans Dreamweaver. Au sens strict du terme, une commande est un type d'extension particulier. Cependant, dans certains contextes, le terme “commande” correspond à “élément de menu”, c'est-à-dire tout élément apparaissant dans un menu Dreamweaver, quelle que soit sa fonction ou son fonctionnement.

Pour créer de nouvelles commandes placées automatiquement dans le menu Commandes, utilisez le panneau Historique. Vous pouvez également utiliser Extension Manager pour installer de nouvelles extensions, notamment des commandes. Pour plus d'informations, voir l'aide de Dreamweaver.

Pour réorganiser les éléments du menu Commandes ou pour déplacer des éléments d'un menu à un autre, il est nécessaire de modifier le fichier menus.xml.

Pour renommer une commande que vous avez créée :

1. Choisissez Commandes > Modifier la liste des commandes.

Une boîte de dialogue s'affiche, répertoriant toutes les commandes que vous pouvez renommer (les commandes se trouvant dans le menu Commandes par défaut n'apparaissent pas dans cette liste et ne peuvent pas être modifiées de cette manière).

2. Sélectionnez la commande que vous souhaitez renommer.
3. Indiquez son nouveau nom.
4. Cliquez sur Fermer.

Le nouveau nom apparaît dans le menu Commandes.

Pour supprimer une commande que vous avez créée :

1. Choisissez Commandes > Modifier la liste des commandes.

   Une boîte de dialogue s'affiche, répertoriant toutes les commandes que vous pouvez supprimer (les commandes se trouvant dans le menu Commandes par défaut n'apparaissent pas dans cette liste et ne peuvent pas être supprimées de cette manière).
2. Sélectionnez la commande que vous souhaitez supprimer.
3. Cliquez sur Supprimer, puis confirmez que vous voulez supprimer la commande.

   La commande est supprimée. La suppression d'une commande n'implique pas seulement la suppression de la commande du menu : le fichier contenant le code de la commande est également supprimé. Cette procédure de suppression doit donc être utilisée avec discernement. Si vous souhaitez supprimer une commande sans supprimer le fichier, localisez le fichier sous Configuration/Commands, puis placez-le dans un autre dossier.
4. Cliquez sur Fermer.

Fonctionnement des commandes de menu

Lorsque l'utilisateur clique sur un menu avec un élément contenant une commande de menu, les événements suivants se produisent :

1. Si l'une des balises `menuitem` du menu contient l'attribut `dynamic`, Dreamweaver appelle la fonction `getDynamicContent()` dans le fichier de commandes de menu associé afin de compléter le menu.
2. Dreamweaver appelle la fonction `canAcceptCommand()` dans chaque fichier de commandes de menu référencé dans le menu pour vérifier si la commande correspond à l'élément sélectionné.
   - Si la fonction `canAcceptCommand()` renvoie la valeur `false`, l'élément de menu apparaît grisé.

- Si la fonction `canAcceptCommand()` renvoie la valeur `true` ou si elle n'est pas définie, Dreamweaver appelle la fonction `isCommandChecked()` pour déterminer si l'élément de menu doit être coché. Si la fonction `isCommandChecked()` n'est pas définie, aucune coche n'apparaît.

3. Dreamweaver appelle la fonction `setMenuText()` pour déterminer le texte devant s'afficher dans le menu.

   Si la fonction `setMenuText()` n'est pas définie, Dreamweaver utilise le texte spécifié dans la balise `menuitem`.

4. L'utilisateur sélectionne un élément dans le menu.
5. Dreamweaver appelle la fonction `receiveArguments()`, si elle est définie, dans le fichier de commandes de menu sélectionné afin de permettre à la commande de traiter tous les arguments transmis depuis l'élément de menu.

   REMARQUE S'il s'agit d'un élément de menu dynamique, l'ID correspondant est le seul argument transmis.

6. Dreamweaver appelle la fonction `commandButtons()`, si elle est définie, pour identifier les boutons qui figurent dans la partie droite de la boîte de dialogue Options et le code qui doit être exécuté lorsque l'utilisateur clique sur ces boutons.
7. Dreamweaver recherche une balise `FORM` dans le fichier de commandes de menu.

   S'il existe un formulaire, Dreamweaver appelle la fonction `windowDimensions()` pour déterminer la taille de la boîte de dialogue Options qui contient les éléments `BODY` du fichier.

   Si la fonction `windowDimensions()` n'est pas définie, Dreamweaver redimensionne automatiquement la boîte de dialogue.
8. Si la balise `BODY` du fichier de commandes de menu contient le gestionnaire `onLoad`, Dreamweaver exécute le code associé à ce gestionnaire (qu'une boîte de dialogue soit affichée ou non). Si aucune boîte de dialogue ne s'affiche, les étapes restantes ne sont pas exécutées.
9. L'utilisateur sélectionne des options dans la boîte de dialogue. Dreamweaver exécute les gestionnaires d'événements associés aux champs du formulaire au fur et à mesure que l'utilisateur les rencontre.
10. L'utilisateur clique sur l'un des boutons définis par la fonction `commandButtons()`.
11. Dreamweaver exécute le code associé au bouton sur lequel l'utilisateur a cliqué.

12. La boîte de dialogue reste affichée jusqu'à ce que l'un des scripts du fichier de commandes de menu appelle la fonction `window.close()`.

Exemple de commande de menu simple

Cet exemple de commande de menu simple explique le fonctionnement des commandes de menu Annuler et Rétablir. La commande de menu Annuler permet à l'utilisateur d'annuler sa dernière modification, la commande Rétablir permet de rétablir une modification précédemment annulée au moyen de Annuler.

Pour appliquer cet exemple, suivez les étapes ci-après :

- Création de commandes de menu
- Ecriture du code JavaScript
- Enregistrement du fichier de commandes dans le dossier Menu

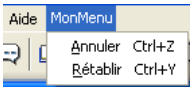
Création de commandes de menu

Ajoutez les balises de menu HTML suivantes à la fin du fichier menus.xml afin de créer un menu intitulé MonMenu contenant les éléments Annuler et Rétablir.

```
<menu name="MyMenu" id="MyMenu_Edit">
<menuitem name="MyUndo" key="Cmd+Z" file="Menus/MyMenu.htm"
  arguments="'undo'" id="MyMenu_Edit_Undo" />
<menuitem name="MyRedo" key="Cmd+Y" file="Menus/MyMenu.htm"
  arguments="'redo'" id="MyMenu_Edit_Redo" />
</menu>
```

L'attribut `key` définit les touches des raccourcis clavier que l'utilisateur peut utiliser pour appeler l'élément de menu. L'attribut `file` indique le nom du fichier de commandes exécuté par Dreamweaver lorsque ce dernier appelle l'élément de menu. La valeur de l'attribut `arguments` définit les arguments transmis par Dreamweaver lorsque ce dernier appelle la fonction `receiveArguments()`.

La figure suivante représente ces éléments de menu :



Ecriture du code JavaScript

Lorsque l'utilisateur sélectionne Annuler ou Rétablir dans le menu MonMenu, Dreamweaver appelle le fichier de commandes MonMenu.htm spécifié par l'attribut `file` de la balise `menuitem`. Créez le fichier de commandes MonMenu.htm dans le dossier Configuration/Menus de Dreamweaver et ajoutez les trois fonctions API des commandes de menu (`canAcceptCommand()`, `receiveArguments()` et `setMenuText()`) afin de mettre en œuvre la logique associée aux éléments de menu Annuler et Rétablir. Les sections ci-dessous décrivent ces fonctions.

canAcceptCommand()

Dreamweaver appelle la fonction `canAcceptCommand()` pour chaque élément du menu MonMenu afin de déterminer s'il doit être activé ou désactivé. Dans le fichier MonMenu.htm, la fonction `canAcceptCommand()` vérifie la valeur de l'argument `arguments[0]` afin de déterminer si Dreamweaver traite un élément de menu Rétablir ou Annuler. Si l'argument `"undo"` est défini, la fonction `canAcceptCommand()` appelle la fonction d'activateur `dw.canUndo()` et renvoie la valeur renvoyée (`true` ou `false`). De même, si l'argument `"redo"` est défini, la fonction `canAcceptCommand()` appelle la fonction d'activateur `dw.canRedo()` et renvoie sa valeur à Dreamweaver. Si la fonction `canAcceptCommand()` renvoie la valeur `false`, Dreamweaver grise l'élément de menu pour lequel la fonction a été appelée. L'exemple suivant décrit le code de la fonction `canAcceptCommand()`:

```
function canAcceptCommand()
{
  var selarray;
  if (arguments.length != 1) return false;
  var bResult = false;

  var whatToDo = arguments[0];
  if (whatToDo == "undo")
  {
    bResult = dw.canUndo();
  }
  else if (whatToDo == "redo")
  {
    bResult = dw.canRedo();
  }
  return bResult;
}
```

receiveArguments()

Dreamweaver appelle la fonction `receiveArguments()` pour traiter les arguments définis dans la balise `menuitem`. Pour les éléments de menu Annuler et Rétablir, la fonction `receiveArguments()` appelle soit la fonction `dw.undo()`, soit la fonction `dw.redo()`, selon la valeur de l'argument `arguments[0]` (`"undo"` ou `"redo"` respectivement). La fonction `dw.undo()` annule la dernière action effectuée par l'utilisateur dans la fenêtre de document, la boîte de dialogue ou le panneau actif. La fonction `dw.redo()` rétablit la dernière opération annulée.

Le code de la fonction `receiveArguments()` ressemble à ceci :

```
function receiveArguments()
{
  if (arguments.length != 1) return;

  var whatToDo = arguments[0];
  if (whatToDo == "undo")
  {
    dw.undo();
  }
  else if (whatToDo == "redo")
  {
    dw.redo();
  }
}
```

Dans cette commande, la fonction `receiveArguments()` traite les arguments et exécute la commande. Des commandes de menu plus complexes peuvent appeler des fonctions différentes pour exécuter la commande. Par exemple, le code suivant vérifie si le premier argument est `"foo"` ; si c'est le cas, il appelle la fonction `doOperationX()` et lui transmet le deuxième argument. Si le premier argument est `"bar"`, le code appelle la fonction `doOperationY()` et lui transmet le deuxième argument. La fonction `doOperationX()` ou `doOperationY()` est chargée de l'exécution de la commande.

```
function receiveArguments(){
  if (arguments.length != 2) return;

  var whatToDo = arguments[0];

  if (whatToDo == "foo"){
    doOperationX(arguments[1]);
  }else if (whatToDo == "bar"){
    doOperationX(arguments[1]);
  }
}
```

setMenuText()

Dreamweaver appelle la fonction `setMenuText()` pour déterminer le texte de l'élément de menu. Si la fonction `setMenuText()` n'est pas définie, Dreamweaver utilise le texte spécifié dans l'attribut `name` de la balise `menuitem`.

La fonction `setMenuText()` vérifie la valeur de l'argument transmis par Dreamweaver (`arguments[0]`). Si la valeur de cet argument est `"undo"`, Dreamweaver appelle la fonction `dw.getUndoText()` ; si sa valeur est `"redo"`, Dreamweaver appelle la fonction `dw.getRedoText()`. La fonction `dw.getUndoText()` renvoie le texte spécifiant l'opération que Dreamweaver doit annuler. Par exemple, si l'utilisateur utilise plusieurs fois la commande Rétablir, la fonction `dw.getUndoText()` peut renvoyer le texte de menu « Annuler Modifier source ». De même, la fonction `dw.getRedoText()` renvoie le texte spécifiant l'opération que Dreamweaver doit rétablir. Si l'utilisateur utilise plusieurs fois la commande Annuler, la fonction `dw.getRedoText()` peut renvoyer le texte de menu « Répéter Modifier source ».

Le code de la fonction `setMenuText()` ressemble à ceci :

```
function setMenuText()
{
  if (arguments.length != 1) return "";

  var whatToDo = arguments[0];
  if (whatToDo == "undo")
    return dw.getUndoText();
  else if (whatToDo == "redo")
    return dw.getRedoText();
  else return "";
}
```

Enregistrement du fichier de commandes dans le dossier Menu

Pour mettre en œuvre les éléments de menu Annuler et Rétablir, vous devez enregistrer le fichier MonMenu.htm dans le dossier Configuration/Menus de Dreamweaver ou dans un sous-dossier créé à cet effet. L'emplacement du fichier doit correspondre à celui que vous avez spécifié dans la balise `menuitem`. Pour que Dreamweaver puisse l'exploiter, vous devez redémarrer Dreamweaver ou recharger les extensions. Pour plus d'informations sur le rechargement des extensions, voir *Rechargez les extensions.*, page 111.

Pour exécuter les commandes de menu, sélectionnez l'élément de menu lorsqu'il est activé. Dreamweaver appelle alors les fonctions dans le fichier de commandes, comme indiqué dans *Fonctionnement des commandes de menu*, page 207.

Exemple de menu dynamique

Cet exemple décrit la mise en œuvre du sous-menu Aperçu dans le navigateur de Dreamweaver qui permet d'afficher la liste des navigateurs disponibles. Cet exemple décrit également l'ouverture du fichier en cours (ou des fichiers sélectionnés dans le panneau Site) dans le navigateur spécifié par l'utilisateur. Pour mettre en œuvre ce menu dynamique, suivez les étapes ci-dessous :

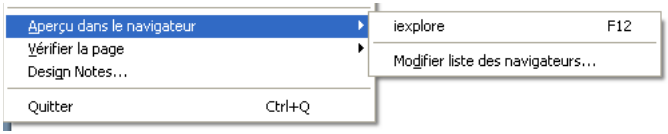
- Création des éléments de menu dynamiques
- Ecriture du code JavaScript

Création des éléments de menu dynamiques

Les balises de menu du fichier menus.xml présentées ci-dessous définissent le sous-menu Aperçu dans le navigateur du menu Fichier :

```
<menu name="_Preview in Browser" id="DWMenu_File_PIB">
  <menuitem dynamic name="No Browsers Selected"
    file="Menus/MM/PIB_Dynamic.htm" arguments="'No Browsers'"
    id="DWMenu_File_PIB_Default" />
  <separator />
  <menuitem name="_Edit Browser List..." enabled="true"
    command="dw.editBrowserList()" id="DWMenu_File_PIB_EditBrowserList" />
  </menu>
```

La première balise `menuitem` définit l'élément de menu par défaut Pas de navigateur sélectionné. Cet élément apparaît dans le sous-menu si vous n'avez spécifié aucun navigateur pour l'élément Aperçu dans le navigateur dans les Préférences. Si vous avez spécifié Microsoft Internet Explorer, le sous-menu est similaire à l'illustration ci-dessous :



L'attribut `name` du premier élément de menu indique le fichier de commandes PIB_Dynamic.htm. Ce fichier contient les lignes suivantes :

```
<SCRIPT LANGUAGE="javascript" SRC="PIB_Dynamic.js"></SCRIPT>
```

La balise `script` inclut le code JavaScript du fichier PIB_Dynamic.js, lequel contient le code JavaScript associé au sous-menu Aperçu dans le navigateur. Il est possible d'enregistrer ce code directement dans le fichier PIB_Dynamic.htm, mais il est plus utile de l'enregistrer dans un fichier séparé. Ceci permet en effet d'inclure le même code dans un grand nombre de commandes.

Ecriture du code JavaScript

Etant donné que la première balise `menuitem` contient l'attribut `dynamic`, Dreamweaver appelle la fonction `getDynamicContent()` du fichier PIB_Dynamic.js décrite dans l'exemple suivant :

```
function getDynamicContent(itemID)
{
  var browsers = null;
  var PIB = null;
  var i;
  var j=0;
  browsers = new Array();
  PIB = dw.getBrowserList();

  for (i=0; i<PIB.length; i=i+2)
  {
    browsers[j] = new String(PIB[i]);

    if (dw.getPrimaryBrowser() == PIB[i+1])
       browsers[j] += "\tF12";
    else if (dw.getSecondaryBrowser() == PIB[i+1])
      browsers[j] += "\tCmd+F12";

    browsers[j] += ";id='"+escQuotes(PIB[i])+"'";

    if (itemID == "DWPopup_PIB_Default")
      browsers[j] = MENU_strPreviewIn + browsers[j];

    j = j+1;
  }
  return browsers;
}
```

La fonction `getDynamicContent()` appelle la fonction `dw.getBrowserList()` afin d'obtenir un tableau des noms de navigateurs indiqués dans la section Aperçu dans le navigateur des Préférences de Dreamweaver. Ce tableau contient les noms de tous les navigateurs et les chemins d'accès aux fichiers exécutables. Ensuite, pour chaque élément du tableau `(i=0; i<PIB.length; i=i+2)`, la fonction `getDynamicContents()` place le nom du navigateur (`PIB[i]`) dans un second tableau appelé `browsers` (`browsers[j] = new String(PIB[i]);`). Si un navigateur a été désigné comme navigateur principal ou secondaire, la fonction ajoute les noms des raccourcis clavier qui permettent de l'appeler. La chaîne `";id="` est ensuite ajoutée, suivie du nom du navigateur entre apostrophes (exemple : `;id='iexplore'`). Si l'argument `itemID` est "`DWPopup_PIB_Default`", la fonction fait précéder l'élément du tableau de la chaîne `Preview in.` Une fois qu'une entrée a été créée pour chaque navigateur répertorié dans Préférences, la fonction `getDynamicContent()` renvoie les `navigateurs` du tableau vers Dreamweaver. Si vous n'avez sélectionné aucun navigateur dans Préférences, la fonction renvoie la valeur `null` et Dreamweaver affiche Pas de navigateur sélectionné dans le menu.

canAcceptCommand()

Dreamweaver appelle ensuite la fonction `canAcceptCommand()` pour chaque balise `menuitem` faisant référence à un fichier de commandes avec l'attribut `file`. Si la fonction `canAcceptCommand()` renvoie la valeur `false`, l'élément de menu s'affiche en grisé. Si la fonction `canAcceptCommand()` renvoie la valeur `true`, Dreamweaver active l'élément de menu. Si la fonction renvoie `true` ou si elle n'est pas définie, Dreamweaver appelle la fonction `isCommandChecked()` pour déterminer si l'élément de menu doit être coché. Si la fonction `isCommandChecked()` n'est pas définie, aucune coche n'apparaît.

```
function canAcceptCommand()
{
  var PIB = dw.getBrowserList();

  if (arguments[0] == 'primary' || arguments[0] == 'secondary')
    return havePreviewTarget();

  return havePreviewTarget() && (PIB.length > 0);
}
```

La fonction `canAcceptCommand()` du fichier PIB_Dynamic.js récupère à nouveau la liste des navigateurs créée dans Préférences. Elle vérifie ensuite si le premier argument (`arguments[0]`) est principal ou secondaire. Si tel est le cas, elle renvoie la valeur renvoyée par la fonction `havePreviewTarget()`. Dans le cas contraire, elle effectue des tests afin de vérifier l'appel de la fonction `havePreviewTarget()` *et* de voir si des navigateurs ont été spécifiés ou non (`PIB.length > 0`). Si *les deux* tests sont vrais (true), la fonction renvoie la valeur `true`. Si au moins un test est faux (false), la fonction renvoie la valeur `false`.

havePreviewTarget()

La fonction `havePreviewTarget()` est définie par l'utilisateur et renvoie la valeur `true` si Dreamweaver dispose d'une cible valide à afficher dans le navigateur. Ce peut être un document ou un groupe de fichiers sélectionné dans le panneau Site. La fonction `havePreviewTarget()` ressemble à l'exemple ci-dessous :

```
function havePreviewTarget()
{
  var bHavePreviewTarget = false;

  if (dw.getFocus(true) == 'site')
  {
    if (site.getFocus() == 'remote')
    {
      bHavePreviewTarget = site.getRemoteSelection().length > 0 &&
                    site.canBrowseDocument();
    }
    else if (site.getFocus() != 'none')
    {
      var selFiles = site.getSelection();

      if (selFiles.length > 0)
      {
        var i;

        bHavePreviewTarget = true;

        for (i = 0; i < selFiles.length; i++)
        {
          var selFile = selFiles[i];

          // Pour les connexions à un serveur, les fichiers
          // seront déjà des URL distantes.

          if (selFile.indexOf("://") == (-1))
          {
            var urlPrefix = "file:///";
            var strTemp = selFile.substr(urlPrefix.length);

            if (selFile.indexOf(urlPrefix) == -1)
              bHavePreviewTarget = false;
            else if (strTemp.indexOf("/") == -1)
              bHavePreviewTarget = false;
            else if (!DWfile.exists(selFile))
              bHavePreviewTarget = false;
            else if (DWfile.getAttributes(selFile).indexOf("D") != -1)
              bHavePreviewTarget = false;
          }
          else
```

```
            {
              bHavePreviewTarget = true;
            }
          }
        }
      }
    }
    else if (dw.getFocus() == 'document' ||
      dw.getFocus() == 'textView' || dw.getFocus("true") == 'html' )
    {
      var dom = dw.getDocumentDOM('document');
      if (dom != null)
      {
        var parseMode = dom.getParseMode();
        if (parseMode == 'html' || parseMode == 'xml')
          bHavePreviewTarget = true;
      }
    }

    return bHavePreviewTarget;
}
```

La fonction `havePreviewTarget()` règle la valeur `bHavePreviewTarget` sur `false` comme valeur renvoyée par défaut. Cette fonction effectue deux tests de base en appelant la fonction `dw.getFocus()` pour déterminer quelle partie de l'application est active. Lors du premier test, elle vérifie que le panneau Site est actif (`if (dw.getFocus(true) == 'site'`)). Si ce n'est pas le cas, elle vérifie, lors du second test, si un document (`dw.getFocus() == 'document'`), le mode Texte (`dw.getFocus() == 'textView'`) ou l'inspecteur de code (`dw.getFocus("true") == 'html'`) est actif. Si aucun test n'est vrai (true), la fonction renvoie la valeur `false`.

Si le panneau Site est actif, la fonction vérifie si l'affichage est défini sur distant. Si c'est le cas, la fonction définit `bHavePreviewTarget` sur `true`, s'il existe des fichiers distants (`site.getRemoteSelection().length > 0`) et que vous pouvez les ouvrir dans un navigateur (`site.canBrowseDocument()`). Si l'affichage n'est pas distant, et s'il n'est pas défini sur Aucun, la fonction obtient une liste des fichiers sélectionnés (`var selFiles = site.getSelection();`) sous la forme d'URL `file:///`.

La fonction effectue un test sur chaque élément de la liste afin de vérifier si la chaîne de caractères
"`://`" y figure. Si ce n'est pas le cas, le code effectue une série de tests sur l'élément de la liste. Si l'élément ne se présente pas sous la forme d'une URL `file:///` (`if (selFile.indexOf(urlPrefix) == -1)`), la valeur renvoyée est `false`. Si le reste de la chaîne apparaissant après le préfixe `file:///` ne contient pas de barre oblique (/) (`if (strTemp.indexOf("/") == -1)`), la valeur renvoyée est définie sur `false`. Si le fichier n'existe pas (`else if (!DWfile.exists(selFile))`), la valeur renvoyée est définie sur `false`. Enfin, la fonction vérifie si le fichier spécifié est un dossier (`else if (DWfile.getAttributes(selFile).indexOf("D") != -1)`). Si `selfile` est un dossier, la fonction renvoie la valeur `false`. Autrement, si la cible est un fichier, la fonction définit `bHavePreviewTarget` sur la valeur `true`.

Si un document, le mode Texte ou l'inspecteur de code est actif (`else if (dw.getFocus() == 'document' || dw.getFocus() == 'textView' || dw.getFocus("true") == 'html' )`), la fonction obtient le DOM et vérifie s'il s'agit d'un document HTML ou XML. Si tel est le cas, la fonction définit `bHavePreviewTarget` sur `true`. Enfin, la fonction renvoie la valeur stockée dans `bHavePreviewTarget`.

receiveArguments()

Dreamweaver appelle la fonction `receiveArguments()` pour que la commande traite tous les arguments provenant de l'élément de menu. En ce qui concerne le menu Aperçu dans le navigateur, la fonction `receiveArguments()` appelle le navigateur sélectionné par l'utilisateur. La fonction `receiveArguments()` ressemble à ceci :

```
function receiveArguments()
{
  var whichBrowser = arguments[0];
  var theBrowser = null;
  var i=0;
  var browserList = null;
  var result = false;

  if (havePreviewTarget())
  {
    // Code pour vérifier si l'appel provient d'un raccourci clavier
    if (whichBrowser == 'primary' || whichBrowser == 'secondary')
    {
      // pour obtenir le chemin du navigateur sélectionné
      if (whichBrowser == 'primary')
      {
        theBrowser = dw.getPrimaryBrowser();
      }
      else if (whichBrowser == 'secondary')
```

```
    {
      theBrowser = dw.getSecondaryBrowser();
    }

    // Faire correspondre le chemin au nom du navigateur correspondant
    // qui apparaît dans le menu
    browserList = dw.getBrowserList();
    while(i < browserList.length)
    {
      if (browserList[i+1] == theBrowser)
        theBrowser = browserList[i];
      i+=2;
    }
  }
  else
    theBrowser = whichBrowser;
  // Lancer le navigateur uniquement si un navigateur valide est
sélectionné.
  if (theBrowser != "file:///" && typeof(theBrowser) != "undefined" &&
theBrowser.length > 0)
  {
    if (dw.getFocus(true) == 'site')
    {
      // Obtenir uniquement le premier élément de la sélection car
      // browseDocument() ne peut prendre en charge un tableau.
      //dw.browseDocument(site.getSelection()[0],theBrowser);
      site.browseDocument(theBrowser);
    }
    else
      dw.browseDocument(dw.getDocumentPath('document'),theBrowser);
  }
  else
  {
    // Dans le cas contraire, un utilisateur a appuyé sur F12 ou
Ctrl+F12, demander s'il souhaite
    // indiquer maintenant un navigateur primaire ou secondaire.
    if (whichBrowser == 'primary')
    {
      result = window.confirm(MSG_NoPrimaryBrowserDefined);
    }
    else if (whichBrowser == 'secondary')
    {
      result = window.confirm(MSG_NoSecondaryBrowserDefined);
    }

    // Si l'utilisateur a cliqué sur OK, les préférences s'affichent avec
le panneau du navigateur
    if (result)
      dw.showPreferencesDialog('browsers');
  }
```

```
    }
}
```

La fonction commence par régler la variable `whichBrowser` sur la valeur transmise par Dreamweaver, `arguments[0]`. Outre la définition d'autres valeurs par défaut, la fonction règle également le `renvoi` sur la valeur par défaut, `false`.

Une fois les variables initialisées, la fonction `receiveArguments()` appelle la fonction définie par l'utilisateur `havePreviewTarget()` et teste le résultat. Si le résultat du test est vrai (true), la fonction vérifie si l'utilisateur a sélectionné le navigateur principal ou secondaire. Dans ce cas, la fonction définit la variable `theBrowser` sur le chemin du fichier exécutable qui lance le navigateur (`dw.getPrimaryBrowser()` ou `dw.getSecondaryBrowser()`). La fonction effectue ensuite une boucle qui examine la liste des navigateurs renvoyée par `dw.getBrowsersList()`. Si le chemin vers un navigateur de la liste correspond au chemin vers le navigateur principal ou secondaire, la fonction définit la variable `theBrowser` sur la valeur de correspondance dans `browserList`. La valeur contient le nom du navigateur et le chemin vers le fichier exécutable qui lance le navigateur. Si `havePreviewTarget()` renvoie la valeur `false`, la fonction définit la variable `theBrowser` sur la valeur de la variable `whichBrowser`.

La fonction `receiveArguments()` teste ensuite la variable `theBrowser` pour vérifier qu'elle ne débute pas par un chemin, qu'elle n'est pas "non définie" (`"undefined"`) et que sa taille est supérieure à 0. Si toutes ces conditions sont remplies et que le panneau Site est actif, la fonction `receiveArguments()` appelle la fonction `site.browseDocument()` pour qu'elle ouvre dans le navigateur sélectionné les fichiers sélectionnés dans le panneau Site. Si le panneau Site n'est pas actif, la fonction `receiveArguments()` appelle la fonction `dw.browseDocument()` et lui transmet le chemin du document actif et la valeur de la variable `theBrowser`, qui spécifie le nom du navigateur à utiliser pour afficher le document.

Si l'utilisateur a appuyé sur des touches de raccourci (F12 ou Ctrl+F12) et qu'aucun navigateur principal ou secondaire n'a été spécifié, une boîte de dialogue s'affiche pour l'avertir. Si l'utilisateur clique sur OK, la fonction appelle la fonction `dw.showPreferencesDialog()` avec l'argument `browsers` pour que l'utilisateur puisse spécifier un navigateur à ce stade.

API des commandes de menu

Les fonctions personnalisées de l'API des commandes de menu ne sont pas obligatoires.

canAcceptCommand()

Description

Détermine si l'élément de menu doit être actif ou grisé.

Arguments

{arg1}, {arg2},...{argN}}

- S'il s'agit d'une option de menu dynamique, l'ID unique correspondant spécifié dans la fonction `getDynamicContents()` est le seul argument transmis. Si l'attribut `arguments` est défini pour une balise `menuitem`, la valeur de l'attribut est transmise à la fonction `canAcceptCommand()` (et aux fonctions `isCommandChecked()`, `receiveArguments()` et `setMenuText()`) sous forme d'un ou plusieurs arguments. L'attribut `arguments` permet de distinguer deux éléments de menu appelant la même commande de menu.

REMARQUE	L'attribut `arguments` n'est pas pris en compte pour les options de menu dynamiques.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'élément doit être activé ; `false` dans le cas contraire.

commandButtons()

Description

Définit les boutons figurant dans la partie droite de la boîte de dialogue Options et leur comportement lorsque l'utilisateur clique dessus. Si cette fonction n'est pas définie, aucun bouton n'apparaît et la section `BODY` du fichier de commandes de menu s'étend de façon à remplir la totalité de la boîte de dialogue.

Arguments

Aucun.

Valeurs renvoyées

Dreamweaver attend un tableau contenant un nombre pair d'éléments. Le premier élément est une chaîne contenant le libellé du premier bouton. Le deuxième élément est une chaîne de code JavaScript définissant le comportement du premier bouton lorsque l'utilisateur clique dessus. Les autres éléments définissent les boutons supplémentaires de la même manière.

Exemple

Dans l'exemple suivant, la fonction `commandButtons()` définit les boutons OK, Annuler et Aide.

```
function commandButtons(){
  return new Array("OK" , "doCommand()" , "Cancel" , ¬
  "window.close()" , "Help" , "showHelp()");
}
```

getDynamicContent()

Description

Extrait le contenu de la partie dynamique du menu.

Arguments

`menuID`

- L'argument *`menuID`* est la valeur de l'attribut `id` dans la balise `menuitem` associée à l'élément de menu.

Valeurs renvoyées

Dreamweaver attend un tableau de chaînes, où chaque chaîne contient le nom d'un élément de menu et son ID unique, séparés par un point-virgule. Si cette fonction renvoie la valeur `null`, aucune modification n'est apportée au menu.

Exemple

Dans l'exemple suivant, la fonction `getDynamicContent()` renvoie un tableau de quatre éléments de menu (Mon élément de menu 1, Mon élément de menu 2, Mon élément de menu 3, Mon élément de menu 4) :

```
function getDynamicContent(){
  var stringArray= new Array();
  var i=0;
  var numItems = 4;

  for (i=0; i<numItems;i++)
    stringArray[i] = new String("My Menu Item " + i + ";¬
    id='My-MenuItem" + i + "'");

  return stringArray;
}
```

isCommandChecked()

Description

Détermine si une coche doit apparaître à côté de l'élément de menu.

Arguments

{arg1}, {arg2},...{argN}

- S'il s'agit d'une option de menu dynamique, l'ID unique correspondant spécifié dans la fonction `getDynamicContents()` est le seul argument transmis. Si l'attribut `arguments` est défini pour une balise `menuitem`, la valeur de l'attribut est transmise à la fonction `isCommandChecked()` (et aux fonctions `canAcceptCommand()`, `receiveArguments()` et `setMenuText()`) sous forme d'un ou de plusieurs arguments. L'attribut `arguments` permet de distinguer deux éléments de menu appelant la même commande de menu.

REMARQUE	L'attribut `arguments` n'est pas pris en compte pour les options de menu dynamiques.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si une coche doit apparaître à côté de l'option de menu ; `false` dans le cas contraire.

Exemple

```
function isCommandChecked()
{
  var bChecked = false;
  var cssStyle = arguments[0];

  if (dw.getDocumentDOM() == null)
    return false;

  if (cssStyle == "(None)")
  {
    return dw.cssStylePalette.getSelectedStyle() == '';
  }
  else
  {
    return dw.cssStylePalette.getSelectedStyle() == cssStyle;
  }
    return bChecked;
}
```

receiveArguments()

Description

Traite tous les arguments transmis à partir d'un élément du menu ou de la fonction `dw.runCommand()`. S'il s'agit d'une option de menu dynamique, elle traite l'ID de l'option de menu dynamique.

Arguments

`{arg1}, {arg2},...{argN}`

- S'il s'agit d'une option de menu dynamique, l'ID unique correspondant spécifié dans la fonction `getDynamicContents()` est le seul argument transmis. Si l'attribut `arguments` est défini pour une balise `menuitem`, la valeur de l'attribut est transmise à la fonction `receiveArguments()` (et aux fonctions `canAcceptCommand()`, `isCommandChecked()` et `setMenuText()`) sous forme d'un ou de plusieurs arguments. L'attribut `arguments` permet de distinguer deux éléments de menu appelant la même commande de menu.

REMARQUE	L'attribut `arguments` n'est pas pris en compte pour les options de menu dynamiques.

Valeurs renvoyées

Dreamweaver n'attend rien.

Exemple

```
function receiveArguments()
{
  var styleName = arguments[0];
  if (styleName == "(None)")
    dw.getDocumentDOM('document').applyCSSStyle('','');
  else
    dw.getDocumentDOM('document').applyCSSStyle('',styleName);
}
```

setMenuText()

Description

Spécifie le texte devant s'afficher dans le menu.

REMARQUE	N'utilisez pas cette fonction si vous utilisez getDynamicContent().

Arguments

{arg1}, {arg2},...{argN}

- Si l'attribut `arguments` est défini pour une balise `menuitem`, la valeur de cet attribut est transmise à la fonction `setMenuText()` (et aux fonctions `canAcceptCommand()`, `isCommandChecked()` et `receiveArguments()`) sous la forme d'un ou de plusieurs arguments. L'attribut `arguments` permet de distinguer deux éléments de menu appelant la même commande de menu.

Valeurs renvoyées

Dreamweaver attend la chaîne devant apparaître dans le menu.

Exemple

window`function setMenuText()`

```
{
  if (arguments.length != 1) return "";

  var whatToDo = arguments[0];
  if (whatToDo == "undo")
    return dw.getUndoText();
  else if (whatToDo == "redo")
    return dw.getRedoText();
  else return "";
}
```

Dimensions()

Description

Définit les dimensions de la boîte de dialogue des paramètres. Si cette fonction n'est pas définie, les dimensions de la fenêtre sont calculées automatiquement.

REMARQUE	Ne définissez cette fonction que si vous souhaitez utiliser une boîte de dialogue supérieure à 640 x 480 pixels.

Arguments

platform

- La valeur de l'argument *platform* est soit `"macintosh"`, soit `"windows"`, selon la plate-forme utilisée par l'utilisateur.

Valeurs renvoyées

Dreamweaver attend une chaîne au format `"widthInPixels,heightInPixels"`.

Les dimensions renvoyées sont inférieures à la taille totale de la boîte de dialogue parce qu'elles n'incluent pas la zone des boutons OK et Annuler. Si les dimensions renvoyées ne permettent pas de faire apparaître toutes les options, des barres de défilement s'affichent.

Exemple

Dans l'exemple suivant, la fonction `windowDimensions()` définit les dimensions de la boîte de dialogue des paramètres à 648 x 520 pixels :

```
function windowDimensions(){
  return "648,520";
}
```

CHAPITRE 9

Barres d'outils

9

Pour créer une nouvelle barre d'outils dans Macromedia Dreamweaver 8, il suffit de créer un fichier de définition de barre d'outils et de placer ce fichier dans le dossier Configuration/Toolbars. Dans ce fichier de barre d'outils, vous définissez, à l'aide de quelques balises XML personnalisées, des éléments tels que des boutons-poussoir, des boutons radio, des zones de texte et des menus déroulants. Ensuite, vous attribuez des attributs et des commandes à ces éléments pour définir leur aspect et leur comportement, et pouvez ajouter d'autres fichiers de barre d'outils et des éléments de référence définis dans d'autres barres d'outils.

Le tableau ci-dessous recense les fichiers utilisés pour créer une barre d'outils :

Chemin	Fichier	Description
Configuration/Toolbars/	toolbars.xml	Modifiez ce fichier pour changer le contenu de la barre d'outils.
Configuration/Toolbars/	*nouvelle_barre_outils*.xml	Créez ce fichier pour générer une barre d'outils.
Configuration/Toolbars/	*fichier_image*.gif	Image d'icône associée à l'élément de barre d'outils.
Configuration/Commands/	MyCommand.htm	Fichier de commandes associé à l'élément de barre d'outils.

Fonctionnement des barres d'outils

Les barres d'outils sont définies par les fichiers XML et d'image localisés dans le dossier Toolbars du dossier de configuration principal de Dreamweaver. Les barres d'outils par défaut de Dreamweaver sont stockées dans le fichier toolbars.xml du dossier Configuration/Toolbars. Au démarrage, Dreamweaver charge tous les fichiers de barre d'outils dans le dossier Toolbars. Pour ajouter une nouvelle barre d'outils, vous pouvez simplement copier le fichier correspondant dans le dossier Toolbars, plutôt que de modifier le fichier toolbars.xml original.

Les fichiers XML de barre d'outils définissent une ou plusieurs barres d'outils, ainsi que les éléments qui les composent. Une barre d'outils est une liste d'éléments tels que des boutons, des zones de texte, des menus déroulants, etc. Un élément de barre d'outils est un contrôle auquel l'utilisateur peut accéder.

Certains types de contrôles de barre d'outils, tels que les boutons-poussoirs et les menus déroulants, sont associés à une icône. Les icônes sont stockées dans le sous-dossier images du dossier Toolbars. Elles sont généralement au format de fichier GIF ou JPEG, mais tous les formats compatibles avec Dreamweaver sont acceptés. Les images pour les barres d'outils créées dans Macromedia sont stockées dans le dossier Toolbars/images/MM.

Tout comme pour les menus, vous pouvez spécifier la fonction de chaque élément de la barre d'outils en définissant ses attributs ou par l'intermédiaire d'un fichier de commandes. Les fichiers de commandes des barres d'outils créées dans Macromedia sont stockés dans le dossier Toolbars/MM.

CONSEIL

L'API de barre d'outils est compatible avec l'API de commandes de menu. Les contrôles de barre d'outils peuvent ainsi réutiliser les fichiers de commandes des menus.

Contrairement aux menus, les éléments de barre d'outils peuvent être définis indépendamment des barres d'outils qui les utilisent. Vous pouvez ainsi utiliser des éléments de barre d'outils dans plusieurs barres d'outils en appliquant la balise `itemref`.

Lorsque Dreamweaver charge une barre d'outils pour la première fois, sa visibilité et sa position sont déterminées par le fichier de définition de la barre d'outils. Par la suite, sa visibilité et sa position sont enregistrées et restaurées à partir du registre (Windows) ou du fichier de préférences Dreamweaver (Macintosh).

Comportement des barres d'outils

Sous Windows, les barres d'outils Dreamweaver se comportent, en général, comme les barres d'outils standard du système. Les barres d'outils Dreamweaver présentent toutefois les caractéristiques suivantes :

- Vous pouvez les ancrer, les détacher et les repositionner par rapport aux autres barres d'outils en utilisant la fonction de glisser-déposer.
- Vous pouvez les ancrer horizontalement sur le bord supérieur ou inférieur la fenêtre.

  Dans l'espace de travail de Dreamweaver, qui intègre toutes les fenêtres de documents de Dreamweaver dans un unique cadre parent, vous pouvez spécifier si les barres d'outils doivent s'ancrer à l'espace de travail ou à la fenêtre de document.

Les barres d'outils ancrées à l'espace de travail de Dreamweaver n'apparaissent qu'une seule fois. Elles fonctionnent toujours sur le document au premier plan. Dans l'espace de travail de Dreamweaver, vous pouvez ancrer les barres d'outils au-dessus, en dessous, à gauche ou à droite de la barre Insérer. Les barres d'outils ancrées à l'espace de travail de Dreamweaver ne sont pas automatiquement désactivées lorsque la fenêtre de document est fermée. Les éléments de la barre d'outils déterminent s'ils doivent rester actifs lorsque aucun document n'est ouvert.

Lorsque les barres d'outils sont ancrées sur la fenêtre de document, une instance s'affichera pour chaque fenêtre. La barre d'outils attachée à une fenêtre de document est désactivée lorsque la fenêtre dans laquelle elle est ancrée n'est pas la fenêtre active. Les gestionnaires de mise à jour sont exécutés dès que leur fenêtre devient la fenêtre active.

Vous ne pouvez pas faire glisser les barres d'outils entre la fenêtre de document et l'espace de travail de Dreamweaver.

- La taille des barres d'outils ne change pas. La taille d'une barre d'état ne se réduit pas lorsque le conteneur est réduit ou lorsque d'autres barres d'outils sont ajoutées.
- Vous pouvez afficher ou masquer les barres d'outils à partir du menu Affichage > Barres d'outils.
- Les barres d'outils ne peuvent pas se chevaucher.
- Lorsque vous faites glisser une barre d'outils, seul son contour s'affiche.

Sur Macintosh, les barres d'outils sont toujours attachées à la fenêtre de document. Vous pouvez les afficher ou les masquer à partir du menu, mais vous ne pouvez pas les faire glisser, les réorganiser ni les détacher.

Fonctionnement des commandes de barres d'outils

Lorsque Dreamweaver dessine une barre d'outils, les événements suivants se produisent :

1. Pour chaque élément de contrôle de la barre d'outils, Dreamweaver détermine si l'attribut `file` existe.
2. Si l'attribut `file` existe, Dreamweaver appelle la fonction `canAcceptCommand()` pour définir si le contrôle doit être activé dans le contexte actuel du document.

   Par exemple, dans la zone de texte Titre du document de la barre d'outils de Dreamweaver, la fonction `canAcceptCommand()` vérifie l'existence d'un DOM et si le document actif est un fichier HTML. Si ces deux conditions sont vérifiées, la fonction renvoie la valeur `true` et Dreamweaver active la zone de texte dans la barre d'outils.
3. Si l'attribut `file` existe, Dreamweaver ignore les attributs suivants, s'ils sont spécifiés : `checked`, `command`, `DOMRequired`, `enabled`, `script`, `showif`, `update` et `value`.

4. Si l'attribut `file` n'existe pas, Dreamweaver traite les attributs définis pour l'élément de contrôle de la barre d'outils : `checked`, `command`, `DomRequired`, etc.

   Pour plus d'informations sur les attributs de balises d'éléments spécifiques, voir *Attributs de balises d'éléments*, page 244.
5. Dreamweaver appelle la fonction `getCurrentValue()` à chaque cycle de mise à jour, comme spécifié par l'attribut `update`, pour déterminer la valeur de contrôle à afficher.
6. L'utilisateur sélectionne un élément dans la barre d'outils.
7. Dreamweaverappelle la fonction `receiveArguments()` pour traiter les arguments spécifiés par l'attribut `arguments` de l'élément de la barre d'outils.

Pour plus d'informations sur le rôle des fonctions spécifiques dans l'API de commande de barre d'outils, voir *API de commande de la barre d'outils*, page 251.

Fichier de commandes de barre d'outils simple

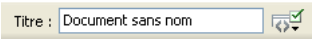
Cet exemple simple implémente un élément de zone de texte Titre comme indiqué sur la barre d'outils Document de Dreamweaver. L'élément de zone de texte permet à l'utilisateur d'entrer un nom pour le document Dreamweaver en cours. Procédez comme suit pour implémenter cet exemple de barre d'outils :

- Création de la zone de texte
- Ecriture du code JavaScript

Création de la zone de texte

Pour ajouter une barre d'outils dans Dreamweaver, vous placez le fichier XML contenant la définition de barre d'outils dans le dossier Configuration de Dreamweaver.

L'illustration suivante présente la zone de texte Titre :



L'élément de barre d'outils `editcontrol` suivant définit une zone de texte nommée Titre :

```
<EDITCONTROL ID="DW_SetTitle"
  label="Title: "
  tooltip="Document Title"
  width="150"
  file="Toolbars/MM/EditTitle.htm"/>
```

L'attribut `tooltip` entraîne l'affichage dans Dreamweaver de la mention Titre du document dans une info-bulle lorsque l'utilisateur positionne le curseur sur la zone de texte. L'attribut `width` définit la taille du champ en pixels. L'attribut `file` précise que le fichier EditTitle.htm contient les fonctions JavaScript qui agissent sur la zone de texte. Pour voir la définition complète de la barre d'outils Document de Dreamweaver, reportez-vous à la barre d'outils principale (`id="DW_Toolbar_Main"`) dans le fichier toolbars.xml.

Ecriture du code JavaScript

Lorsque l'utilisateur agit sur la zone de texte, Dreamweaver invoque le fichier de commandes EditTitle.htm dans le dossier Toolbars/MM. Ce fichier contient trois fonctions JavaScript qui agissent sur la zone de texte Titre. Ces fonctions sont : `canAcceptCommand()`, `receiveArguments()` et `getCurrentValue()`.

canAcceptCommand() : active l'élément de la barre d'outils

La fonction `canAcceptCommand()` se compose d'une ligne de code qui s'assure qu'un DOM (modèle d'objet de document) est disponible et que le document est analysé comme HTML. La fonction renvoie le résultat de ces tests. Si les conditions renvoient la valeur `true`, Dreamweaver active l'élément de zone de texte sur la barre d'outils. Si la fonction renvoie la valeur `false`, Dreamweaver désactive l'élément.

La fonction se présente comme suit :

```
function canAcceptCommand()
{
  return (dw.getDocumentDOM() != null && dw.getDocumentDOM().getParseMode()
  == 'html');
}
```

receiveArguments() : recherche le titre

Dreamweaver appelle la fonction `receiveArguments()`, présentée dans l'exemple suivant, lorsque l'utilisateur entre une valeur dans la zone de texte Titre du document et appuie sur la touche Entrée, ou sélectionne une autre commande.

La fonction se présente comme suit :

```
function receiveArguments(newTitle)
{
  var dom = dw.getDocumentDOM();
  if (dom)
    dom.setTitle(newTitle);
}
```

Dreamweaver transmet `newTitle`, à savoir la valeur entrée par l'utilisateur, à la fonction `receiveArguments()`. La fonction `receiveArguments()` vérifie immédiatement s'il existe un DOM. Si c'est le cas, la fonction `receiveArguments()` définit le titre du nouveau document en transmettant `newTitle` à la fonction `dom.setTitle()`.

getCurrentValue() : recherche le titre

Lorsqu'un cycle de mise à jour survient, selon la fréquence par défaut déterminée dans le gestionnaire d'événements `onEdit`, Dreamweaver appelle la fonction `getCurrentValue()` pour déterminer la valeur à afficher pour le contrôle. La fréquence de mise à jour par défaut du gestionnaire d'événements `onEdit` détermine la fréquence de mise à jour car le contrôle de zone de texte de Titre ne dispose pas d'attribut update.

Pour la zone de texte Titre du document, la fonction `getCurrentValue()` suivante appelle la fonction `dom.getTitle()` de l'interface de programmation d'application (API) JavaScript pour obtenir et renvoyer le titre en cours.

La fonction se présente comme suit :

```
function getCurrentValue()
{
  var title = "";
  var dom = dw.getDocumentDOM();
  if (dom)
    title = dom.getTitle();
  return title;
}
```

La fonction `getTitle()` renvoie la valeur Document sans nom, affichée dans la zone de texte, en attendant que l'utilisateur attribue un titre au document. Après la saisie d'un titre par l'utilisateur, la fonction `getTitle()` renvoie cette valeur et Dreamweaver l'affiche comme nouveau titre du document.

Pour voir le fichier HTML complet qui contient les fonctions JavaScript pour la zone de texte Titre, reportez-vous au fichier EditTitle.htm dans le dossier Toolbars/MM.

Le dossier MM est réservé aux fichiers Macromedia Créez un nouveau dossier dans le dossier Toolbars afin d'y stocker votre code JavaScript.

Fichier de définition de la barre d'outils

Une barre d'outils est une simple liste de boutons radio, boutons-poussoirs, zones de texte et autres éléments de barre d'outils, éventuellement divisée par des balises `separator`. Chaque élément de barre d'outils peut être une référence à un élément à l'aide de la balise `itemref`, une séparation, à l'aide de la balise `separator` ou une définition complète d'élément de barre d'outils pour une case à cocher ou une zone de texte, comme décrit, par exemple, dans *Balises d'éléments de barre d'outils*, page 238.

Tous les fichiers de définition de barre d'outils commencent par les déclarations suivantes :

```
<?xml version="1.0" encoding="optional_encoding"?>
<!DOCTYPE toolbarset SYSTEM "-//Macromedia//DWExtension toolbar 5.0">
```

Si l'encodage est ignoré, Dreamweaver utilise l'encodage par défaut du système d'exploitation.

Après les déclarations, le fichier consiste en une simple balise `toolbarset` qui contient un nombre indéfini des balises suivantes : balises `toolbar`, `itemref`, `separator`, `include` et *`itemtype`*, dans lesquelles *`itemtype`* est `button`, `checkbutton`, `radiobutton`, `menubutton`, `dropdown`, `combobox`, `editcontrol` ou `colorpicker`. L'exemple suivant, un extrait abrégé du fichier toolbars.xml, représente la hiérarchie des balises dans le fichier de barre d'outils. Dans cet exemple, les attributs d'éléments de barre d'outils qui sont décrits dans les sections suivantes, sont remplacés par des .) (...).

```
<?xml version="1.0"?>
<!DOCTYPE toolbarset SYSTEM "-//Macromedia//DWExtension toolbar 5.0">
<toolbarset>

<!-- main toolbar -->
  <toolbar id="DW_Toolbar_Main" label="Document">
    <radiobutton id="DW_CodeView" . . ./>
    <radiobutton id="DW_SplitView" . . ./>
    <radiobutton id="DW_DesignView" . . ./>
    <separator/>
    <checkbutton id="DW_LiveDebug" . . ./>
    <checkbutton id="DW_LiveDataView" . . ./>
    <separator/>
    <editcontrol id="DW_SetTitle" . . ./>
    <menubutton id="DW_FileTransfer" . . ./>
    <menubutton id="DW_Preview" . . ./>
    <separator/>
    <button id="DW_DocRefresh" . . ./>
    <button id="DW_Reference" . . ./>
    <menubutton id="DW_CodeNav" . . ./>
    <menubutton id="DW_ViewOptions" . . ./>
  </toolbar>
</toolbarset>
```

La section suivante décrit en détail chacune des balises de barre d'outils.

<toolbar>

Description

Définit une barre d'outils. Dreamweaver affiche les éléments et les séparateurs de gauche à droite dans l'ordre spécifié et les met en forme automatiquement. Le fichier de barre d'outils ne spécifie pas l'espacement entre les éléments, mais vous pouvez définir la largeur de certains types d'éléments.

Attributs

`id, label, {container}, {initiallyVisible}, {initialPosition}, {relativeTo}`

- `id="id_unique"` Obligatoire. Une chaîne d'identificateur doit être unique dans un fichier ainsi que dans tous les fichiers inclus dans ce fichier. Les fonctions API JavaScript qui manipulent une barre d'outils se réfèrent à cette dernière par son ID. Pour plus d'informations sur ces fonctions, voir le *Guide des API de Dreamweaver*. Si deux barres d'outils dans un même fichier possèdent le même ID, Dreamweaver affiche un message d'erreur.
- `label="string"` Obligatoire. L'attribut label spécifie l'étiquette, une chaîne de caractères que Dreamweaver affiche pour l'utilisateur. L'étiquette s'affiche dans le menu Affichage > Barre d'outils et dans la barre de titre de la barre d'outils lorsque celle-ci est flottante.
- `container="mainframe"` ou `"document"` Adopte par défaut la valeur `"mainframe"`. Indique la position d'ancrage de la barre d'outils dans l'espace de travail de Dreamweaver sous Windows. Si le contenant indique `"mainframe"`, la barre d'outils apparaît à l'extérieur de l'espace de travail et fonctionne sur le document au premier plan. Si le contenant indique `"document"`, la barre d'outils apparaît dans chaque fenêtre de document. Sur Macintosh, toutes les barres d'outils s'affichent dans chaque fenêtre de document.
- `initiallyVisible="true"` ou `"false"`. Cette balise indique si la barre d'outils doit être visible la première fois que Dreamweaver la charge depuis le dossier Toolbars. Après le premier chargement, l'utilisateur contrôle sa visibilité. Dreamweaver enregistre l'état en cours dans le registre système (Windows) ou dans le fichier de préférences de Dreamweaver (Macintosh) lorsque l'utilisateur quitte Dreamweaver. Dreamweaver rétablit le paramètre à partir du registre ou du fichier de préférences au redémarrage. Vous pouvez manipuler la visibilité de la barre d'outils en utilisant les fonctions `dom.getToolbarVisibility()` et `dom.setToolbarVisibility()`, comme décrit dans le *Guide des API de Dreamweaver*. Si vous ne définissez pas une valeur pour l'attribut `initiallyVisible`, celui-ci adopte la valeur `true`.

- `initialPosition="top"`, `"below"` ou `"floating"`. Indique la position initiale de la barre d'outils définie par Dreamweaver, par rapport aux autres barres d'outils lors du premier chargement. Les valeurs possibles de l'argument `initialPosition` sont décrites dans la liste suivante :
  - `top` Il s'agit de la position par défaut, de façon à ce que la barre d'outils s'affiche en haut de la fenêtre du document. Si plusieurs barres d'outils spécifient `top` pour un type de fenêtre donné, elles apparaissent dans l'ordre de chargement dans Dreamweaver. Cet ordre risque d'être aléatoire si les barres d'outils sont localisées dans des fichiers séparés.
  - `below` La barre d'outils apparaît au début de la ligne directement en dessous de la barre d'outils spécifiée dans l'attribut `relativeTo`. Dreamweaver affiche un message d'erreur s'il ne trouve pas la barre d'outils `relativeTo`. Si plusieurs barres d'outils spécifient `below` par rapport à la même barre d'outils, elles apparaissent dans l'ordre de chargement dans Dreamweaver. Cet ordre risque d'être aléatoire si les barres d'outils sont localisées dans des fichiers séparés.
  - `floating` La barre d'outils n'est pas ancrée à la fenêtre ; elle flotte au-dessus du document. Dreamweaver place automatiquement la barre d'outils de sorte qu'elle soit décalée par rapport aux autres barres d'outils flottantes. Sur Macintosh, `floating` est traité de la même façon que `top`.

  Comme avec l'attribut `initiallyVisible`, l'attribut `initialPosition` s'applique uniquement lors du premier chargement de la barre d'outils par Dreamweaver. Ensuite, la position de la barre d'outils est enregistrée dans le registre ou dans le fichier de préférences de Dreamweaver. Vous pouvez rétablir la position de la barre d'outils en utilisant la fonction `dom.setToolbarPosition()`. Pour plus d'informations sur la fonction `dom.setToolbarPosition()`, voir le *Guide des API de Dreamweaver*.

  Si vous ne spécifiez pas une valeur pour l'attribut `initialPosition`, Dreamweaver positionne la barre d'outils selon l'ordre de chargement.
- `relativeTo="`*`id_barre_outils`*`"` Cet attribut est requis si l'attribut `initialPosition` indique `below`. Dans les autres cas, cet argument est ignoré. Indique l'ID de la barre d'outils en dessous de laquelle cette barre d'outils doit être placée.

Contenu

La balise `toolbar` comprend les balises `include`, `itemref` et `separator` ainsi que des définitions d'éléments individuels tels que `button`, `combobox`, `dropdown`, etc. Pour obtenir une description des définitions d'éléments que vous pouvez spécifier, voir *Balises d'éléments de barre d'outils*, page 238.

Contenant

La balise `toolbarset`.

Exemple

```
<toolbar id="MyDWedit_toolbar" label="Edit">
```

<include/>

Description

Charge les éléments de barre d'outils depuis le fichier spécifié avant de continuer à charger le fichier en cours. Les éléments de barre d'outils définis dans le fichier inclus peuvent être référencés dans le fichier en cours. Si un fichier tente plusieurs fois d'inclure un autre fichier, Dreamweaver affiche un message d'erreur et ignore la fonction include. Les balises `toolbar` dans le fichier inclus sont ignorées, mais les éléments dans ces barres d'outils restent disponibles comme références dans le fichier en cours.

Attributs

- Le chemin de `fichier`, relatif au dossier Toolbars, du fichier XML de la barre d'outils à inclure.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<include file="mine/editbar.xml"/>
```

<itemtype/>

Description

Définit un élément de barre d'outils. Les éléments de barre d'outils peuvent être des boutons, des boutons radio, des boutons-poussoirs, des zones de liste modifiables, des menus déroulants, etc. Pour obtenir la liste des types d'éléments de barre d'outils que vous pouvez définir, voir *Balises d'éléments de barre d'outils*, page 238.

Attributs

Les attributs varient en fonction de l'élément que vous définissez. Pour obtenir la liste des attributs que vous pouvez spécifier pour les éléments de barre d'outils, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<button id="strikeout_button" .../>
```

<itemref/>

Description

Fait référence à (et inclut dans la barre d'outils en cours) un élément de barre d'outils défini dans une barre d'outils précédente ou en dehors de toutes les barres d'outils.

Attributs

`id, {showIf}`

- `id="`*`reference_id`*`"` Obligatoire. Il doit s'agir de l'ID d'un élément défini au préalable ou inclus dans le fichier. Dreamweaver n'autorise pas les références en aval. Si une balise d'élément de barre d'outils fait référence à un ID non défini, Dreamweaver affiche un message d'erreur et ignore la référence.
- `showIf="`*`script`*`"` Indique que cet élément apparaît uniquement sur la barre d'outils si le script spécifié renvoie la valeur `true`. Par exemple, vous pouvez utiliser `showIf` pour afficher certains boutons dans une application donnée uniquement, ou lorsqu'une page est écrite en langage côté serveur, tel que ColdFusion, ASP ou JSP. Si vous ne spécifiez pas l'attribut `showIf`, l'élément apparaît systématiquement. Dreamweaver vérifie cette propriété à chaque fois que l'activateur de l'élément s'exécute ; c'est-à-dire, en fonction de la valeur de l'attribut `update`. Vous devez utiliser cet attribut avec prudence. L'attribut `showIf` peut être utilisé dans la définition de l'élément ou dans une référence à l'élément depuis une barre d'outils. Si la définition et la référence spécifient toutes deux l'attribut `showIf`, Dreamweaver affiche l'élément uniquement si les deux conditions sont vérifiées. L'attribut `showIf` est l'équivalent de la fonction `showIf()` dans un fichier de commandes.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<itemref id="strikeout_button">
```

<separator/>

Description

Insère un séparateur à l'emplacement en cours dans la barre d'outils.

Attributs

`{showIf}`

- L'attribut `showif` indique que le séparateur ne doit apparaître sur la barre d'outils que si le script donné renvoie la valeur `true`. Par exemple, vous pouvez utiliser l'attribut `showIf` pour afficher le séparateur uniquement dans une application donnée ou uniquement lorsque la page contient un certain type de document. Si l'attribut `showIf` n'est pas spécifié, le séparateur s'affiche systématiquement.

Contenu

Néant

Contenant

La balise `toolbar`.

Exemple

```
<separator/>
```

Balises d'éléments de barre d'outils

Chaque type d'éléments de barre d'outils possède sa propre balise et son propre ensemble d'attributs obligatoires et facultatifs. Vous pouvez définir des éléments `toolbar` à l'intérieur ou à l'extérieur des barres d'outils. En général, il est préférable de les définir à l'extérieur et de les référencer à partir des barres d'outils à l'aide de la balise `itemref`.

Vous pouvez définir les types d'éléments suivants dans une barre d'outils.

<button>

Description

Ce bouton-poussoir exécute une commande spécifique lorsque vous cliquez dessus. Il ressemble au bouton Référence de la barre d'outils de Dreamweaver et se comporte comme tel.

Attributs

```
id, image, tooltip, command, {showIf}, {disabledImage}, {overImage},
  {label}, {file}, {domRequired}, {enabled}, {update}, {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<BUTTON ID="DW_DocRefresh"
  image="Toolbars/images/MM/refresh.gif"
  disabledImage="Toolbars/images/MM/refresh_dis.gif"
  tooltip="Refresh Design View (F5)"
  enabled="((dw.getDocumentDOM() != null) && (dw.getDocumentDOM().getView()
  != 'browse') && (!dw.getDocumentDOM().isDesignViewUpdated()))"
  command="dw.getDocumentDOM().synchronizeDocument()"
  update="onViewChange,onCodeViewSyncChange"/>
```

<checkbutton>

Description

Un bouton-poussoir est un bouton qui peut être activé ou désactivé et qui exécute une commande spécifique lorsqu'il est activé. Lorsqu'il est activé, il apparaît enfoncé et en surbrillance. Lorsqu'il est désactivé, il apparaît plat. Dreamweaver implémente les états suivants pour les boutons-poussoirs : au passage de la souris ; enfoncé ; au passage de la souris si enfoncé ; désactivé si enfoncé. Le gestionnaire spécifié par l'attribut `checked` ou par la fonction `isCommandChecked()` doit garantir que le fait de cliquer sur ce bouton modifie son état.

Attributs

```
id, {showIf}, image, {disabledImage}, {overImage}, tooltip, {label},
  {file}, {domRequired}, {enabled}, checked, {update}, command, {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<CHECKBUTTON ID="DW_LiveDebug"
  image="Toolbars/images/MM/debugview.gif"
  disabledImage="Toolbars/images/MM/globe_dis.gif"
  tooltip="Live Debug"
  enabled="dw.canLiveDebug()"
  checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() ==
  'browse'"
  command="dw.toggleLiveDebug()"
  showIf="dw.canLiveDebug()"
  update="onViewChange"/>
```

‹radiobutton›

Description

Le bouton radio se comporte comme le bouton-poussoir, mais apparaît en relief lorsqu'il est désactivé. Dreamweaver implémente les états suivants pour les boutons radio : au passage de la souris ; enfoncé ; au passage de la souris si enfoncé ; désactivé si enfoncé. Dreamweaver ne force pas l'exclusion mutuelle entre les boutons radio. Le gestionnaire spécifié par l'attribut `checked` ou par la fonction `isCommandChecked()` doit garantir que les états activé et désactivé des boutons radio sont cohérents.

Les boutons radio fonctionnent comme les boutons des modes Code, Création et affichage à deux volets de la barre d'outils de document de Dreamweaver.

Attributs

```
id, image, tooltip, checked, command, {showIf}, {disabledImage},
  {overImage}, {label}, {file}, {domRequired}, {enabled}, {update},
  {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<RADIOBUTTON ID="DW_CodeView"
  image="Toolbars/images/MM/codeView.gif"
  disabledImage="Toolbars/images/MM/codeView_dis.gif"
  tooltip="Show Code View"
  domRequired="false"
  enabled="dw.getDocumentDOM() != null"
  checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() ==
  'code'"
  command="dw.getDocumentDOM().setView('code')"
  update="onViewChange"/>
```

<menubutton>

Description

Un bouton de menu est un bouton qui invoque le menu contextuel spécifié par l'attribut `menuid`. Dreamweaver implémente les états survol de souris et pression pour les boutons de menu. Dreamweaver ne dessine pas la flèche du menu (la flèche pointant vers le bas qui indique que les éléments de menu sont attachés au bouton) ; vous devez l'inclure dans votre icône. Les boutons Gestion de fichiers et Navigation dans le code sur la barre d'outils de document de Dreamweaver sont des exemples de boutons de menu.

Attributs

```
id, image, tooltip, menuID, domRequired, enabled, {showIf},
  {disabledImage}, {overImage}, {label}, {file}, {update}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<MENUBUTTON ID="DW_CodeNav"
  image="Toolbars/images/MM/codenav.gif"
  disabledImage="Toolbars/images/MM/codenav_dis.gif"
  tooltip="Code Navigation"
  enabled="dw.getFocus() == 'textView' || dw.getFocus() == 'html'"
```

```
  menuID="DWCodeNavPopup"
  update="onViewChange"/>
```

<dropdown>

Description

Un menu déroulant est un menu non modifiable qui exécute une commande spécifique attachée à une fonction JavaScript lorsque vous sélectionnez une entrée et que le menu se met à jour. Ce menu ressemble à l'option Format de l'inspecteur de propriétés de texte et fonctionne de la même manière, mais s'affiche en taille standard et non en taille réduite comme dans l'inspecteur de propriétés.

Attributs

```
id, tooltip, file, enabled, checked, value, command, {showIf}, {label},
  {width}, {domRequired}, {update}, {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<dropdown id="Font_Example"
  width="115"
  tooltip="Font"
  domRequired="false"
  file="Toolbars/mine/fontExample.htm"
  update="onSelChange"/>
```

<combobox>

Description

Une zone de liste est un menu déroulant modifiable qui exécute une commande lorsque vous sélectionnez une entrée ou que vous apportez une modification dans la zone de texte, puis changez la sélection. Le menu ressemble à l'option Police de l'inspecteur de propriétés de texte et fonctionne de la même manière, mais s'affiche en taille standard et non en taille réduite comme dans l'inspecteur de propriétés.

Attributs

```
id, file, tooltip, enabled, value, command, {showIl}, {label}, {width},
  {domRequired}, {update}, {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<COMBOBOX ID="Address_URL"
  width="300"
  tooltip="Address"
  label="Address: "
  file="Toolbars/MM/AddressURL.htm"
  update="onBrowserPageBusyChange"/>
```

‹editcontrol›

Description

Une zone de contrôle d'édition est une zone de texte modifiable qui exécute une commande lorsque vous apportez une modification dans la zone de texte et changez la sélection.

Attributs

```
id, tooltip, file, value, command, {showIf}, {label}, {width},
  {domRequired}, {enabled}, {update}, {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<EDITCONTROL ID="DW_SetTitle"
  label="Title: "
  tooltip="Document Title"
  width="150"
  file="Toolbars/MM/EditTitle.htm"/>
```

<colorpicker>

Description

Un sélecteur de couleur est un panneau de couleurs, sans zone de texte associée, qui exécute une commande lorsque l'utilisateur sélectionne une nouvelle couleur. Ce panneau ressemble au sélecteur de couleur de l'inspecteur de propriétés de Dreamweaver et se comporte de la même manière. Vous pouvez choisir une autre icône pour remplacer l'icône par défaut.

Attributs

```
id, tooltip, value, command, {showIf}, {image}, {disabledImage},
  {overImage}, {label}, {colorRect}, {file}, {domRequired}, {enabled},
  {update}, {arguments}
```

Pour obtenir une description de chaque attribut, voir *Attributs de balises d'éléments*, page 244.

Contenu

Néant

Contenant

La balise `toolbar` ou `toolbarset`.

Exemple

```
<colorpicker id="Color_Example"
  image="Toolbars/images/colorpickerIcon.gif"
  disabledImage="Toolbars/images/colorpickerIconD.gif"
  colorRect="0 12 16 16"
  tooltip="Text Color"
  domRequired="false"
  file="Toolbars/mine/colorExample.htm"
  update="onSelChange"/>
```

Attributs de balises d'éléments

Les attributs de balises d'éléments de barre d'outils ont les significations suivantes :

id="unique_id"

Obligatoire. L'attribut `id` est l'identificateur de l'élément de barre d'outils. L'attribut `id` doit être unique dans le fichier en cours, ainsi que dans tous les fichiers inclus dans le fichier en cours. La balise `itemref` utilise l'`id` de l'élément pour référencer et inclure un élément dans une barre d'outils.

Exemple

```
<button id="DW_DocRerefresh" . . . >
```

showIf="script"

Facultatif. Cet attribut précise que l'élément s'affiche sur la barre d'outils uniquement si le script renvoie la valeur `true`. Par exemple, vous pouvez utiliser l'attribut `showIf` pour afficher certains boutons uniquement lorsqu'une page est écrite dans un certain type de langage côté serveur, tel que ColdFusion, ASP ou JSP. Si vous ne spécifiez pas l'attribut `showIf`, l'élément apparaît systématiquement.

L'attribut `showIf` est vérifié à chaque fois que l'activateur de l'élément s'exécute ; c'est-à-dire, en fonction de la valeur de l'attribut `update`. Il est conseillé d'utiliser l'attribut `showIf` avec prudence.

Vous pouvez spécifier l'attribut `showIf` dans la définition de l'élément et dans une référence à l'élément sur une balise `itemref`. Si la définition et la référence spécifient l'attribut `showIf`, l'élément s'affiche uniquement si les deux conditions indiquent la valeur true. L'attribut `showIf` est l'équivalent de la fonction `showIf()` dans un fichier de commandes de barre d'outils. Si vous spécifiez l'attribut `showIf` et la fonction `showif()`, la fonction remplace l'attribut.

Exemple

```
showIf="dw.canLiveDebug()"
```

image="image_path"

Cet attribut est obligatoire pour les boutons, les boutons-poussoirs, les boutons radio, les boutons de menu et de zone de liste modifiable. L'attribut `image` est facultatif pour les sélecteurs de couleurs, et est ignoré pour les autres types d'éléments. L'attribut `image` spécifie le chemin, par rapport au dossier Configuration, du fichier de l'icône qui s'affiche sur le bouton. En général, le fichier de l'icône est au format GIF ou JPEG, mais tous les formats compatibles avec Dreamweaver sont acceptés.

Si vous spécifiez une icône pour un sélecteur de couleur, celle-ci le remplace entièrement. Si l'attribut `colorRect` est également défini, la couleur en cours s'affiche au-dessus de l'icône dans le rectangle spécifié.

Exemple

```
image="Toolbars/images/MM/codenav.gif"
```

disabledImage="image_path"

Facultatif. Dreamweaver ignore l'attribut `disabledImage` des éléments autres que les boutons, les boutons-poussoirs, les boutons radio, les boutons de menu, de sélecteurs de couleurs et de zone de liste modifiable. Cet attribut spécifie le chemin, par rapport au dossier Configuration, du fichier de l'icône que Dreamweaver affiche si le bouton est désactivé. Si vous ne spécifiez pas l'attribut `disabledImage`, Dreamweaver affiche l'image spécifiée dans l'attribut `image` lorsque le bouton est désactivé.

Exemple

```
disabledImage="Toolbars/images/MM/codenav_dis.gif"
```

overImage="image_path"

Facultatif. Dreamweaver ignore l'attribut `overImage` des éléments autres que les boutons, les boutons-poussoirs, les boutons radio, les boutons de menu, de sélecteurs de couleurs et de zone de liste modifiable. Cet attribut spécifie le chemin, par rapport au dossier Configuration, du fichier de l'icône que Dreamweaver affiche lorsque l'utilisateur survole le bouton à l'aide de la souris. Si vous ne spécifiez pas l'attribut `overImage`, le bouton ne change pas lorsque l'utilisateur le survole, mais Dreamweaver dessine un cercle autour du bouton.

Exemple

```
overImage="Toolbars/images/MM/codenav_ovr.gif"
```

tooltip="tooltip string"

Obligatoire. Cet attribut spécifie le texte d'identification ou info-bulle qui apparaît lorsque le curseur de la souris survole l'élément de la barre d'outils.

Exemple

```
tooltip="Code Navigation"
```

label="label string"

Facultatif. L'attribut spécifie l'étiquette qui s'affiche à côté de l'élément. Dreamweaver n'ajoute pas automatiquement deux points aux étiquettes. Les étiquettes des éléments autres que les boutons s'affichent toujours à gauche de l'élément. Dreamweaver place les étiquettes des boutons, boutons-poussoirs, boutons radio, de menu et de zone de liste modifiable à l'intérieur du bouton et à droite de l'icône.

Exemple

```
label="Title: "
```

width="number"

Facultatif. Cet attribut ne s'applique qu'aux éléments de zone de texte, menu contextuel et zone de liste en spécifiant la largeur de l'élément en pixels. Si vous ne spécifiez pas l'attribut `width`, Dreamweaver utilise une largeur par défaut moyenne.

Exemple

```
width="150"
```

menuID="menu_id"

Cet attribut est requis pour les boutons de menu et les zones de liste, excepté si vous spécifiez la fonction `getMenuID()` dans un fichier de commande associé. Dreamweaver ignore l'attribut `menuID` pour les autres types d'éléments. Cet attribut spécifie l'ID de la barre de menu contenant le menu contextuel qui doit apparaître lorsque l'utilisateur clique sur le bouton, le bouton de menu ou de liste. L'ID provient de l'attribut ID d'une balise `menubar` dans le fichier menus.xml.

Exemple

```
menuID="DWCodeNavPopup"
```

colorRect="left top right bottom"

L'attribut est facultatif pour les sélecteurs de couleur qui disposent d'un attribut image. L'attribut `colorRect` est ignoré pour tous les autres types d'éléments, ainsi que pour les sélecteurs de couleurs qui ne spécifient pas une image. Si vous spécifiez l'attribut `colorRect`, Dreamweaver affiche la couleur couramment sélectionnée dans le sélecteur de couleur, dans le rectangle, par rapport au côté gauche ou supérieur de l'icône. Si vous ne spécifiez pas l'attribut `colorRect`, Dreamweaver n'affiche pas la couleur couramment sélectionnée sur l'image.

Exemple

```
colorRect=”0 12 16 16”
```

file="command_file_path"

Obligatoire pour les menus déroulants et les zones de liste modifiables. L'attribut `file` est facultatif pour les autres types d'éléments. L'attribut `file` spécifie le chemin, par rapport au dossier Configuration, du fichier de commandes contenant les fonctions JavaScript qui serviront à renseigner, mettre à jour et exécuter l'élément. L'attribut `file` remplace les attributs `enabled`, `checked`, `value`, `update`, `domRequired`, `menuID`, `showIf` et `command`. En général, si vous spécifiez un fichier de commandes avec l'attribut `file`, Dreamweaver ignore tous les attributs équivalents spécifiés dans la balise. Pour plus d'informations sur les fichiers de commandes, voir *API de commande de la barre d'outils*, page 251.

Exemple

```
file="Toolbars/MM/EditTitle.htm"
```

domRequired="true" or "false"

Facultatif. Comme avec les menus, l'attribut `domRequired` spécifie si le mode Création doit être synchronisé avec le mode Code avant que Dreamweaver n'exécute la commande associée. Si vous ne définissez pas cet attribut, il adopte la valeur `true`. Cet attribut est l'équivalent de la fonction `isDOMRequired()` dans un fichier de commandes de barre d'outils.

Exemple

```
domRequired="false"
```

enabled="script"

Facultatif. Comme avec les menus, le script renvoie une valeur spécifiant si l'élément est activé. Si vous ne définissez pas cet attribut, il adopte la valeur enabled. L'attribut `enabled` est l'équivalent de la fonction `canAcceptCommand()` dans un fichier de commandes de barre d'outils.

Exemple

```
enabled="dw.getFocus() == 'textView' || dw.getFocus() == 'html'"
```

checked="script"

Cet attribut est obligatoire pour les boutons-poussoirs et radio. Dreamweaver ignore l'attribut `checked` pour les autres types d'éléments. Comme avec les menus, le script renvoie une valeur spécifiant si l'élément est activé ou pas. L'attribut `checked` est l'équivalent de la fonction `isCommandChecked()` dans un fichier de commandes de barre d'outils. Si vous ne définissez pas cet attribut, il adopte la valeur unchecked.

Exemple

```
checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() ==
  'code'"
```

value="script"

Cet attribut est obligatoire pour les menus déroulants, les zones de liste modifiables, les zones de texte et les sélecteurs de couleurs. Dreamweaver ignore l'attribut `value` pour les autres types d'éléments.

Pour déterminer la valeur à afficher pour les menus déroulants et les zones de liste modifiables, Dreamweaver appelle d'abord la fonction `isCommandchecked()` pour chaque élément dans le menu. Si la fonction `isCommandchecked()` renvoie la valeur `true` pour un élément, Dreamweaver affiche la valeur du premier. Si aucun élément ne renvoie la valeur `true` ou si la fonction `isCommandChecked()` n'est pas définie, Dreamweaver appelle la fonction `getCurrentValue()` ou exécute le script spécifié par l'attribut `value`. Si le contrôle est une zone de liste modifiable, Dreamweaver affiche la valeur renvoyée. Si le contrôle est un menu déroulant, Dreamweaver ajoute temporairement la valeur renvoyée dans la liste et l'affiche.

Dans tous les autres cas, le script renvoie la valeur en cours à afficher. Pour les menus déroulants ou les zones de liste modifiables, cette valeur doit correspondre à l'un des éléments dans la liste du menu. Pour les zones de liste modifiables et les zones de texte, la valeur peut correspondre à une chaîne quelconque renvoyée par le script. Pour les sélecteurs de couleurs, la valeur doit être une couleur valide, mais Dreamweaver ne met pas ceci en vigueur.

L'attribut value est l'équivalent de la fonction `getCurrentValue()` dans un fichier de commandes de barre d'outils.

update="update_frequency_list"

Facultatif. Cet attribut spécifie la fréquence d'exécution des gestionnaires des attributs `enabled`, `checked`, `showif` et `value` pour mettre à jour l'état visible de l'élément. L'attribut `update` est l'équivalent de la fonction `receiveArguments()` dans un fichier de commandes de barre d'outils.

Vous devez spécifier la fréquence de la mise à jour des éléments de barre d'outils car ces éléments, contrairement aux éléments de menu, sont toujours visibles. C'est pourquoi vous devez toujours sélectionner la fréquence la moins élevée possible et vous assurer que les gestionnaires des attributs `enabled`, `checked` et `value` sont aussi simples que possible.

La liste suivante présente les gestionnaires possibles pour la fonction *liste_fréquence_mise_à_jour*, dans l'ordre croissant de fréquence. Si vous ne définissez pas une valeur pour l'attribut `update`, la fréquence de mise à jour prend la valeur `onEdit`. Vous pouvez spécifier plusieurs fréquences de mise à jour, séparées par des virgules. Les gestionnaires s'exécutent sur l'un des événements spécifiés suivants :

- `onServerModelChange` s'exécute lorsque le modèle de serveur de la page en cours change.
- `onCodeViewSyncChange` s'exécute lorsque le mode Code se synchronise ou se désynchronise avec le mode Création.
- `onViewChange` s'exécute chaque fois que l'utilisateur bascule entre les modes Code et Création, ou entre les modes Code, Création et affichage à deux volets.
- `onEdit` s'exécute chaque fois que le document est modifié en mode Création. Les modifications apportées en mode Code ne déclenchent pas cet événement.
- `onSelChange` s'exécute chaque fois que la sélection est modifiée en mode Création. Les modifications apportées en mode Code ne déclenchent pas cet événement.
- `onEveryIdle` s'exécute régulièrement lorsque l'application est inactive. Ceci peut demander un certain temps car les gestionnaires `enabler/checked/showif/value` sont exécutés fréquemment. Utilisez-la uniquement pour les boutons dont l'état actif doit être modifié rapidement à des moments spécifiques.

REMARQUE	En réalité, dans tous ces cas, Dreamweaver exécute les gestionnaires après l'événement spécifié, lorsque l'application est au repos. L'exécution des gestionnaires après chaque modification ou changement de sélection n'est pas garantie ; les gestionnaires s'exécutent dès que possible après une série de modifications ou de changements de sélection. Par contre, les gestionnaires s'exécutent lorsque l'utilisateur clique sur un élément de barre d'outils.

Exemple

```
update="onViewChange"
```

command="script"

Cet attribut est obligatoire pour tous les éléments sauf pour les boutons de menu. Dreamweaver ignore l'attribut `command` pour les boutons de menu. Spécifie la fonction JavaScript à exécuter lorsque l'utilisateur effectue l'une des opérations suivantes :

- L'utilisateur clique sur un bouton.
- L'utilisateur sélectionne un élément dans un menu déroulant ou dans une zone de liste.
- L'utilisateur appuie sur la touche Tab pour passer à une autre zone, appuie sur la touche Retour ou clique hors d'une zone de texte ou de liste.
- L'utilisateur sélectionne une couleur dans un sélecteur de couleur.

L'attribut `command` est l'équivalent de la fonction `receiveArguments()` dans un fichier de commandes de barre d'outils.

Exemple

```
command="dw.toggleLiveDebug()"
```

arguments="argument_list"

Facultatif. Cet attribut spécifie la liste d'arguments, séparés par des virgules, pour transmettre la fonction `receiveArguments()` à un fichier de commandes de barre d'outils. Si vous ne définissez pas l'attribut `arguments`, Dreamweaver transmet l'ID de l'élément de la barre d'outils. En outre, les menus déroulants, les zones de liste modifiables, les zones de texte et les sélecteurs de couleurs transmettent leur valeur en cours comme premier argument, avant ceux spécifiés par l'attribut `arguments` et avant l'ID de l'élément si aucun argument n'est spécifié.

Exemple

Dans une barre d'outils qui comporte les boutons Annuler et Répéter, chaque bouton appelle le fichier de commandes de menu, Edit_Clipboard.htm, et transmet un argument spécifiant l'action, comme indiqué dans l'exemple suivant :

```
<button id="DW_Undo"
 image="Toolbars/images/MM/undo.gif"
 disabledImage="Toolbars/images/MM/undo_dis.gif"
 tooltip="Undo"
 file="Menus/MM/Edit_Clipboard.htm"
 arguments="'undo'"
 update="onEveryIdle"/>

<button id="DW_Redo"
 image="Toolbars/images/MM/redo.gif"
 disabledImage="Toolbars/images/MM/redo_dis.gif"
 tooltip="Redo"
 file="Menus/MM/Edit_Clipboard.htm"
 arguments="'redo'"
 update="onEveryIdle"/>
```

API de commande de la barre d'outils

Dans de nombreux cas où vous avez spécifié un script pour un attribut, vous pouvez également implémenter cet attribut à l'aide d'une fonction JavaScript dans un fichier de commandes. Cette action est nécessaire lorsque les fonctions doivent accepter des arguments, comme dans le gestionnaire d'une zone de texte, et obligatoire pour les menus déroulants et les zones de liste modifiables.

L'API du fichier de commandes pour les éléments de la barre d'outils est une extension de l'API du fichier de commandes de menu ; vous pouvez ainsi réutiliser les fichiers de commandes de menu en tant que fichiers de commandes de barre d'outils en leur ajoutant des fonctions supplémentaires spécifiques aux barres d'outils.

canAcceptCommand()

Disponibilité

Dreamweaver MX.

Description

Détermine si la barre d'outils est activée. La condition par défaut d'un élément étant l'état activé, vous n'avez pas besoin de définir cette fonction, sauf si la valeur `false` est renvoyée dans au moins un cas.

Arguments

Dans le cas des menus déroulants, zones de liste modifiables, zones de texte et sélecteurs de couleurs, le premier argument est la valeur en cours dans le contrôle. La fonction `getDynamicContent()` peut, éventuellement, associer des ID individuels aux éléments d'un menu déroulant. Si l'élément sélectionné dans le menu déroulant dispose d'un ID, Dreamweaver transmet cet ID à la fonction `canAcceptCommand()` au lieu de transmettre la valeur. En ce qui concerne les zones de liste modifiables, si le contenu actuel de la zone de texte ne correspond pas à une entrée dans le menu déroulant, Dreamweaver transmet le contenu de la zone de texte. Dreamweaver compare le contenu de la liste avec celui du menu déroulant, sans tenir compte des majuscules et des minuscules, afin de vérifier la correspondance des éléments.

Si vous spécifiez l'attribut `arguments` pour cet élément de la barre d'outils dans le fichier toolbars.xml, les arguments sont transmis ensuite. Si vous n'avez pas défini l'attribut `arguments`, Dreamweaver transmet l'ID de l'élément.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'élément est activé, `false` dans les autres cas.

Exemple

```
function canAcceptCommand()
{
  return (dw.getDocumentDOM() != null);
}
```

getCurrentValue()

Disponibilité

Dreamweaver MX.

Description

Renvoie la valeur en cours pour afficher l'élément. Dreamweaver appelle la fonction `getCurrentValue()` pour les menus déroulants, les zones de liste modifiables, les zones de texte et les sélecteurs de couleurs. Pour les menus déroulants, la valeur en cours doit représenter l'un des éléments dans le menu. Si ce n'est pas le cas, Dreamweaver sélectionne le premier élément de la liste. Pour les zones de liste modifiables et les zones de texte, cette valeur peut correspondre à une chaîne quelconque renvoyée par la fonction. Pour les sélecteurs de couleurs, la valeur doit être une couleur valide, mais Dreamweaver ne force pas cette condition. Cette fonction est l'équivalent de l'attribut `value`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend l'affichage d'une chaîne contenant la valeur en cours. Pour le sélecteur de couleur, la chaîne contient la forme RVB de la couleur sélectionnée (par exemple #FFFFFF pour le blanc).

Exemple

```
function getCurrentValue()
{
  var title = "";
  var dom = dw.getDocumentDOM();
  if (dom)
    title = dom.getTitle();
  return title;
}
```

getDynamicContent()

Disponibilité

Dreamweaver MX.

Description

Cette fonction est obligatoire pour les menus déroulants et les zones de liste modifiables. Comme pour les menus, cette fonction renvoie un tableau de chaînes pour renseigner le menu déroulant. Chacune de ces chaînes peut éventuellement se terminer par `";id=id"`. Si un ID est spécifié, Dreamweaver le transmet à la fonction `receiveArguments()` au lieu de faire apparaître la chaîne dans le menu.

Le nom `getDynamicContent()` prête à confusion car cette fonction doit être utilisée même si la liste des entrées dans le menu est fixe. Par exemple, le fichier Text_Size.htm du dossier Configuration/Menus/MM n'est pas un menu dynamique ; il est conçu pour être appelé depuis des éléments statiques d'un ensemble de menus. Toutefois, l'ajout d'une fonction `getDynamicContent()` renvoyant une liste de tailles de police possibles permet d'utiliser le même fichier de commandes dans un menu déroulant de barre d'outils. Les éléments de barre d'outils ignorent les caractères de soulignement dans les chaînes des tableaux renvoyés, pour vous permettre de réutiliser les fichiers de commandes de menu. Dans le fichier de commandes de menu, Dreamweaver ignore la fonction `getDynamicContent()` car l'élément de menu n'est pas marqué comme dynamique.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau de chaînes pour renseigner le menu.

Exemple

```
function getDynamicContent()
{
  var items = new Array;
  var filename = dw.getConfigurationPath() + "/Toolbars/MM/
  AddressList.xml";
  var location = MMNotes.localURLToFilePath(filename);
  if (DWfile.exists(location))
  {
    var addressData = DWfile.read(location);
    var addressDOM = dw.getDocumentDOM(dw.getConfigurationPath() +
      '/Shared/MM/Cache/empty.htm');
    addressDOM.documentElement.outerHTML = addressData;
    var addressNodes = addressDOM.getElementsByTagName("url");
    if (addressNodes.length)
    {
      for (var i=0; i < addressNodes.length ; i++ )
      {
        items[i] = addressNodes[i].address + ";id='" +
          addressNodes[i].address + "'";
      }
```

```
    }
  }
  return items;
```

getMenuID()

Disponibilité

Dreamweaver MX.

Description

Valide uniquement pour les boutons de menu. Dreamweaver appelle la fonction `getMenuID()` pour obtenir l'ID du menu devant apparaître lorsque l'utilisateur clique sur le bouton.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne contenant un ID de menu, défini dans le fichier menus.xml.

Exemple

```
function getMenuID()
{
  var dom = dw.getDocumentDOM();
  var menuID = '';
  if (dom)
  {
    var view = dom.getView();
    var focus = dw.getFocus();
    if (view == 'design')
    {
      menuID = 'DWDesignOnlyOptionsPopup';
    }
    else if (view == 'split')
    {
      if (focus == 'textView')
      {
        menuID = 'DWSplitCodeOptionsPopup';
      }
      else
      {
        menuID = 'DWSplitDesignOptionsPopup';
      }
    }
    else if (view == 'code')
```

```
    {
      menuID = 'DWCodeOnlyOptionsPopup';
    }
    else
    {
      menuID = 'DWBrowseOptionsPopup';
    }
  }
  return menuID;
}
```

getUpdateFrequency()

Disponibilité

Dreamweaver MX.

Description

Spécifie la fréquence d'exécution des gestionnaires des attributs `enabled`, `checked`, `showif` et `value` pour mettre à jour l'état visible de l'élément.

Vous devez spécifier la fréquence de la mise à jour des éléments de barre d'outils car ces éléments, contrairement aux éléments de menu, sont toujours visibles. C'est pourquoi vous devez toujours choisir la fréquence la moins élevée possible et vous assurer que les gestionnaires des attributs `enabled`, `checked` et `value` sont aussi simples que possible.

Cette fonction est l'équivalent de l'attribut `update` dans un élément de barre d'outils.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne contenant la liste des gestionnaires de mise à jour, séparés par des virgules. Pour obtenir la liste complète des gestionnaires de mise à jour possibles, voir *update="update_frequency_list"*, page 249.

Exemple

```
function getUpdateFrequency()
{
  return onSelChange";
}
```

isCommandChecked()

Disponibilité

Dreamweaver MX.

Description

Renvoie une valeur indiquant si l'élément est sélectionné. Dans le cas des boutons, l'attribut checked indique si le bouton apparaît activé ou enfoncé. La fonction `isCommandChecked()` est l'équivalent de l'attribut `checked` dans une balise d'élément de barre d'outils.

Arguments

Dans le cas des menus déroulants, zones de liste modifiables, zones de texte et sélecteurs de couleurs, le premier argument est la valeur en cours dans le contrôle. La fonction `getDynamicContent()` peut, éventuellement, associer des ID individuels aux éléments d'un menu déroulant. Si l'élément sélectionné dans le menu dispose d'un ID, Dreamweaver transmet cet ID à la fonction `isCommandChecked()` au lieu de transmettre la valeur. En ce qui concerne les zones de liste modifiables, si le contenu actuel de la zone de texte ne correspond pas à une entrée dans le menu déroulant, Dreamweaver transmet le contenu de la zone de texte. Dreamweaver compare le contenu de la zone de liste avec celui du menu déroulant, sans tenir compte des majuscules et des minuscules, pour vérifier la correspondance des éléments.

Si vous avez spécifié l'attribut `arguments` pour cet élément, les arguments sont transmis ensuite. Si vous ne définissez pas l'attribut `arguments`, Dreamweaver transmet l'ID de l'élément.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'élément est activé, `false` dans les autres cas.

Exemple

L'exemple suivant détermine quel élément, le cas échéant, doit être activé dans un menu déroulant contenant des formats de paragraphe et des styles CSS :

```
function isCommandChecked()
{
  var bChecked = false;
  var style = arguments[0];
  var textFormat = dw.getDocumentDOM().getTextFormat();

  if (dw.getDocumentDOM() == null)
    bChecked = false;

  if (style == "(None)")
```

```
    bChecked = (dw.cssStylePalette.getSelectedStyle() == '' || textFormat ==
"" || textFormat == "P" || textFormat == "PRE");
  else if (style == "Heading 1")
    bChecked =  (textFormat == "h1");
  else if (style == "Heading 2")
    bChecked =  (textFormat == "h2");
  else if (style == "Heading 3")
    bChecked =  (textFormat == "h3");
  else if (style == "Heading 4")
    bChecked =  (textFormat == "h4");
  else if (style == "Heading 5")
    bChecked =  (textFormat == "h5");
  else if (style == "Heading 6")
    bChecked =  (textFormat == "h6");
  else
    bChecked = (dw.cssStylePalette.getSelectedStyle() == style);

  return bChecked;
}
```

isDOMRequired()

Disponibilité

Dreamweaver MX.

Description

Spécifie si la commande de barre d'outils nécessite un DOM valide pour fonctionner. Si cette fonction renvoie la valeur `true` ou si la fonction n'est pas définie, Dreamweaver suppose que la commande nécessite un DOM valide et synchronise les modes Code et Création du document avant de l'exécuter. Cette fonction est l'équivalent de l'attribut `domRequired` dans une balise d'élément de barre d'outils.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le DOM est obligatoire, `false` s'il ne l'est pas.

Exemple

```
function isDOMRequired()
{
  return false;
}
```

receiveArguments()

Disponibilité

Dreamweaver MX.

Description

Traite tous les arguments transmis depuis un élément de barre d'outils. La fonction `receiveArguments()` est l'équivalent de l'attribut `command` dans une balise d'élément de barre d'outils.

Arguments

Dans le cas des menus déroulants, zones de liste modifiables, zones de texte et sélecteurs de couleurs, le premier argument est la valeur en cours dans le contrôle. La fonction `getDynamicContent()` peut, éventuellement, associer des ID individuels aux éléments d'un menu déroulant. Si l'élément sélectionné dans le menu contextuel déroulant dispose d'un ID, Dreamweaver transmet cet ID à la fonction `receiveArguments()` au lieu de transmettre la valeur. En ce qui concerne les zones de liste modifiables, si le contenu actuel de la zone de texte ne correspond pas à une entrée dans le menu déroulant, Dreamweaver transmet le contenu de la zone de texte. Dreamweaver compare le contenu de la zone de liste avec celui du menu déroulant, sans tenir compte des majuscules et des minuscules, pour vérifier la correspondance des éléments.

Si vous avez spécifié l'attribut `arguments` pour cet élément, les arguments sont transmis ensuite. Si vous n'avez pas défini l'attribut `arguments`, Dreamweaver transmet l'ID de l'élément.

Valeurs renvoyées

Dreamweaver n'attend rien.

Exemple

```
function receiveArguments(newTitle)
{
  var dom = dw.getDocumentDOM();
  if (dom)
    dom.setTitle(newTitle);
}
```

showIf()

Disponibilité

Dreamweaver MX.

Description

Indique qu'un élément apparaît sur la barre d'outils uniquement si la fonction renvoie la valeur `true`. Par exemple, vous pouvez utiliser la fonction `showIf()` pour afficher certains boutons uniquement lorsque la page utilise un modèle de serveur donné. Si vous ne définissez pas la fonction `showif()`, l'élément s'affiche toujours. La fonction `showIf()` est l'équivalent de l'attribut `showIf` dans une balise d'élément de barre d'outils.

La fonction `showIf()` est appelée à chaque fois que l'activateur de l'élément s'exécute ; c'est-à-dire, en fonction de la valeur renvoyée par la fonction `getUpdateFrequency()`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'élément s'affiche, `false` si ce n'est pas le cas.

Exemple

```
function showif()
{
     var retval = false;
     var dom = dw.getDocumentDOM();

     if(dom)
     {
       var view = dom.getView();
       if(view == 'design')
       {
         retval = true;
       }
     }
     return retval;
}
```

CHAPITRE 10

Rapports

10

Macromedia Dreamweaver 8 gère deux types de rapports : les rapports de site et les rapports autonomes.

Rapports de site

Utilisez l'API de rapports pour créer des rapports de site personnalisés ou modifier le jeu de rapports prédéfinis fournis avec Dreamweaver 8. Les rapports de site sont accessibles uniquement à partir de la boîte de dialogue Rapports.

Les rapports de site sont localisés dans le dossier Dreamweaver Configuration/Reports. Ce dossier contient plusieurs sous-dossiers représentant les différentes catégories de rapport, chaque rapport ne pouvant appartenir qu'à une seule catégorie. Les noms de catégorie sont limités à 31 caractères. Chaque sous-dossier peut contenir un fichier nommé _foldername.txt. Si ce fichier existe, Dreamweaver utilise son contenu comme nom de la catégorie. S'il n'existe pas, Dreamweaver utilise le nom du dossier.

Lorsqu'un utilisateur choisit plusieurs rapports de site dans la boîte de dialogue Rapports, Dreamweaver place tous les résultats dans la même fenêtre de résultats dans l'onglet Rapports du site du panneau Résultats. Dreamweaver remplace ces résultats la prochaine fois qu'un utilisateur exécute un rapport de site.

Le tableau ci-dessous recense les fichiers utilisés pour créer un rapport de site :

Chemin	Fichier	Description
Configuration/Reports/*{type/}*	*nom_rapport*.js	Contient les fonctions requises pour générer le contenu du rapport.

Chemin	Fichier	Description
Configuration/Reports/*{type/}*	*nom_rapport*.htm	Appelle les fichiers JavaScript appropriés. Définit l'interface utilisateur (IU) de la boîte de dialogue Paramètres associée au rapport, le cas échéant.
Configuration/Reports/	Reports.js	Fournit les fonctions communes utilisées pour la génération de rapports.

Fonctionnement des rapports de site

1. Pour accéder aux rapports, choisissez Site > Rapports. Dans la boîte de dialogue qui s'affiche, sélectionnez les rapports à exécuter sur des cibles spécifiques.
2. Sélectionnez les fichiers pour lesquels effectuer les rapports à l'aide du menu déroulant Rapport sur. Ce menu contient les commandes suivantes : Document actif, Site local en cours entier, Fichiers sélectionnés dans le site et Dossier. Lorsque vous sélectionnez la commande Dossier, un bouton Parcourir et un champ de texte s'affichent pour vous permettre de choisir un dossier.
3. Vous pouvez personnaliser des rapports possédant des paramètres en cliquant sur le bouton Paramètres, puis en saisissant des valeurs pour ces paramètres. Pour pouvoir définir vous-même des paramètres, le rapport doit contenir une boîte de dialogue Paramètres. La définition de ces paramètres est facultative ; il n'est pas nécessaire de définir les paramètres de chaque rapport. Si un rapport ne possède pas de boîte de dialogue Paramètres, le bouton Paramètres est estompé lorsque vous sélectionnez le rapport dans la liste.
4. Une fois le rapport sélectionné et les paramètres définis, vous cliquez sur le bouton Exécuter.

REMARQUE Si un rapport contient le gestionnaire `preventFileActivity`, Dreamweaver vous interdit d'exécuter toute tâche associée aux fichiers pendant l'exécution du rapport.

Dreamweaver supprime alors tous les éléments de l'onglet Rapports du site dans le panneau Résultats. Dreamweaver appelle la fonction `beginReporting()` dans chaque rapport avant le début du processus de rapport. Si un rapport renvoie la valeur `false` à partir de cette fonction, celle-ci est retirée du processus de rapport.

5. Chaque fichier est transmis à chaque rapport sélectionné dans la boîte de dialogue à l'aide de la fonction `processFile()`. Si le rapport doit inclure des informations sur ce fichier dans la liste des résultats, la fonction `dw.resultsPalette.siteReports.addResultItem()` doit être appelée. Le processus se poursuit jusqu'à ce que tous les fichiers sélectionnés par l'utilisateur soient traités ou jusqu'à ce que l'utilisateur clique sur le bouton Arrêter situé au bas de la fenêtre. Dreamweaver affiche le nom de chaque fichier en cours de traitement et le nombre de fichiers restant à traiter.

   Dreamweaver appelle la fonction `endReporting()` dans chaque rapport à la fin du traitement des fichiers et du processus de rapport.

Exemple simple de rapport de site

L'exemple d'extension simple recense toutes les images référencées dans un fichier spécifique, un site entier, des fichiers sélectionnés ou un dossier et affiche le rapport dans l'onglet Rapports du site de la fenêtre de résultats.

Vous créez cette extension en exécutant les étapes suivantes :

- Création de la définition du rapport
- Ecriture du code JavaScript

Dans cet exemple, deux fichiers sont créés dans le dossier HTML Reports : listeImages.htm, qui contient la définition du rapport, et listeImages.js, qui contient le code JavaScript spécifique au rapport. Vous référencez en outre le fichier Reports.js, intégré à Dreamweaver.

Création de la définition du rapport

La définition du rapport spécifie le nom du rapport tel qu'il s'affiche dans la boîte de dialogue Rapports, appelle tout fichier JavaScript requis et définit l'interface utilisateur de la boîte de dialogue Paramètres, le cas échéant.

Pour créer la définition du rapport :

1. Créez le fichier Configuration/Reports/HTML Reports/listeImages.htm.
2. Ajoutez le code ci-dessous pour indiquer le nom du rapport qui s'affichera dans la boîte de dialogue Rapports, dans le titre de la page HTML.

```
<html>
<head>
<title>List Images</title>
```

3. En fin de fichier, ajoutez la balise `script` et spécifiez le fichier Reports.js dans l'attribut `src`.

```
<script src="../Reports.js"></script>
```

4. En fin de fichier, ajoutez une autre balise `script` et spécifiez le fichier listeImages.js, que vous allez créer à l'étape suivante, dans l'attribut `src`.

```
<html>
<head>
<title>List Images</title>
<script src="../Reports.js"></script>
<script src="List Images.js"></script>
```

5. Fermez la balise `head`, insérez des balises `body` d'ouverture et de fermeture, puis fermez la balise `html`.

```
</head>
<body>
</body>
</html>
```

6. Enregistrez le fichier sous le nom listeImages dans le dossier Configuration/Reports/HTML Reports.

Ecriture du code JavaScript

Dreamweaver propose le fichier Reports.js, à partir duquel vous pouvez appeler toute fonction. Vous devez cependant créer également le fichier JavaScript contenant toute fonction spécifique à votre rapport de site personnalisé.

Pour créer le fichier JavaScript :

1. Créez le fichier Configuration/Reports/HTML Reports/listeImages.js et insérez le contenu ci-dessous :

```
// Function: configureSettings
// Description: Standard report API, used to initialize and load
//  the default values. Does not initialize the UI.
//
function configureSettings() {
  return false;
}

// Function: processFile
// Description: Report command API called during file processing.
//
function processFile (fileURL) {
  if (!isHTMLType(fileURL)) //If the file isn't an HTML file
    return;                 //skip it.
  var curDOM = dw.getDocumentDOM(fileURL); // Variable for DOM
```

```
    var tagList = curDOM.getElementsByTagName('img'); // Variable for img
    tags
    var imgfilename; // Variable for file name specified in img tag
    for (var i=0; i < tagList.length; i++) { // For img tag list
      imgfilename = tagList[i].getAttribute('src'); // Get image filename
      if (imgfilename != null) { // If a filename is specified
       // Print the appropriate icon, HTML filename,
       // image filename, and line number
       reportItem(REP_ITEM_CUSTOM, fileURL, imgfilename,
    curDOM.nodeToSourceViewOffsets(tagList[i])); }
    }
}
```

2. Enregistrez le fichier sous le nom listeImages.js dans le dossier Configuration/Reports/HTML Reports.

Rapports autonomes

Vous pouvez utiliser l'API de fenêtre Résultats pour créer un rapport autonome. Les rapports autonomes sont des commandes normales qui utilisent directement l'API de la fenêtre des résultats plutôt que l'API de rapports. Les rapports autonomes sont accessibles de la même manière que les autres commandes, à partir des menus ou d'une autre commande.

Les rapports autonomes résident dans le dossier Configuration/Commands de Dreamweaver. Une commande personnalisée associée à un rapport autonome figure dans le menu Commandes.

Dreamweaver crée une nouvelle fenêtre de résultats chaque fois que l'utilisateur exécute un nouveau rapport autonome.

Chemin	Fichier	Description
Configuration/Commands	*nom_commande*.htm	Définit l'interface utilisateur de la boîte de dialogue qui s'affiche lorsque vous sélectionnez la commande et contient le code JavaScript ou une référence au fichier JavaScript qui exécute les actions requises pour générer le rapport.
Configuration/Commands	*nom_commande*.js	Génère une fenêtre de résultats et y insère le rapport.

Fonctionnement des rapports autonomes

1. La commande personnalisée, que vous créez pour générer le rapport, ouvre une nouvelle fenêtre de résultats en appelant la fonction `dw.createResultsWindow()` et en stockant l'objet de résultats renvoyés dans une variable de fenêtre. Les fonctions restantes du processus doivent être appelées en tant que méthodes de cet objet.
2. La commande personnalisée initialise le titre et le format de la fenêtre Résultats en appelant les fonctions `setTitle()` et `SetColumnWidths()` en tant que méthodes de l'objet de la fenêtre Résultats.
3. La commande peut commencer immédiatement à ajouter des éléments dans la fenêtre Résultats en appelant la fonction `addItem()` ou à itérer une liste de fichiers en appelant les fonctions `setFileList()` et `startProcessing()` en tant que méthodes de l'objet de la fenêtre Résultats.
4. Lorsque la commande appelle `resWin.startProcessing()`, Dreamweaver appelle la fonction `processFile()` pour chaque URL de fichier dans la liste. Définissez la fonction `processFile()` dans la commande autonome. Elle reçoit l'URL de fichier comme seul argument. Utilisez la fonction `setCallbackCommands()` de l'objet de la fenêtre Résultats si vous voulez que Dreamweaver appelle la fonction `processFile()` dans une autre commande.
5. Pour appeler la fonction `addItem()`, la fonction `processFile()` doit avoir accès à la fenêtre de résultats créée par la commande autonome. La fonction `processFile()` peut également appeler la fonction `stopProcessing()` de l'objet de la fenêtre Résultats pour arrêter le traitement des fichiers de la liste.

Exemple de rapport autonome simple

L'extension simple de rapport autonome recense les images référencées dans un fichier spécifique et affiche le rapport dans la fenêtre de résultats.

Vous créez cette extension en exécutant les étapes suivantes :

1. Création de l'interface utilisateur de la boîte de dialogue
2. Ecriture du code JavaScript

Dans cet exemple, deux fichiers sont créés dans le dossier Configuration/Commands : listeImages.htm, qui définit l'interface utilisateur de la boîte de dialogue qui s'affiche lorsque vous sélectionnez la commande personnalisée, et listeImages.js, qui contient le code JavaScript spécifique au rapport.

Création de l'interface utilisateur de la boîte de dialogue

La balise `BODY` du fichier HTML spécifie le contenu de la boîte de dialogue qui s'affiche lorsque l'utilisateur sélectionne la commande personnalisée et appelle tout fichier JavaScript requis.

Pour créer le fichier HTML :

1. Créez le fichier Configuration/Commands/listeImages.htm.
2. Insérez le code ci-dessous dans le fichier listeImages.htm :

```
<html>
<head>
<title>Standalone report example</title>
<script src=”Listimages.js”>
</script>
</head>
<body>
<div name="test">
<form name="myForm">
<table>
<tr>
<td>Click OK to display the standalone report.</td>
</tr>
</table>
</form>
</div>
</body>
```

3. Enregistrez le fichier sous le nom listeImages.htm dans le dossier Configuration/Commands.

Ecriture du code JavaScript

Créez ensuite le fichier JavaScript contenant les fonctions spécifiques à votre rapport autonome.

Pour créer le fichier JavaScript :

1. Créez le fichier listeImages.js dans le dossier Configuration/Commands et insérez-y le code suivant :

```
function stdaloneresultwin()
{
  var curDOM = dw.getDocumentDOM("document");
  var tagList = curDOM.getElementsByTagName('img');
  var imgfilename;
  var iOffset = 0;
  var iLineNumber = 0;
```

```
  var resWin = dw.createResultsWindow("Images in File", ¬
    ["Line", "Image"]);

  for (var i=0; i < tagList.length; i++)
  {
    // Get the name of the source file.
      imgfilename = tagList[i].getAttribute('src');
    // Get the character offset from the start of the file
    // to the start of the img tag.
    iOffset = curDOM.nodeToOffsets(curDOM.images[i]);
    // Based on the offset, figure out what line in the file
    // the img tag is on.
    iLineNumber = curDOM.getLineFromOffset(iOffset[0]);
    // As long as the src attribute specifies a file name,
      if (imgfilename != null)
    { // display the line number, and image path.
       resWin.addItem(resWin, "0", "Images in Current File", null, ¬
         null, null, [iLineNumber, imgfilename]);
      }
  }
  return;
}

// add buttons to dialog
function commandButtons()
{
  return new Array("OK", "stdaloneresultwin()", "Cancel",
"window.close()");
}
```

2. Enregistrez le fichier sous le nom listeImages.js dans le dossier Configuration/Commands.

API de rapports

La fonction `processFile()` est la seule fonction obligatoire pour l'API de rapports. Toutes les autres fonctions sont facultatives.

processFile()

Disponibilité

Dreamweaver 4.

Description

Cette fonction est appelée lorsqu'un fichier doit être traité. La commande Rapport doit traiter le fichier sans le modifier et utiliser la fonction `dw.ResultsPalette.SiteReports()`, `addResultItem()` ou `resWin.addItem()` pour renvoyer des informations sur ce fichier. Dreamweaver publie automatiquement le DOM de chaque fichier lorsque le traitement est fini.

Arguments

`strFilePath`

- L'argument *`strFilePath`* est le nom et le chemin complet du fichier à traiter.

Valeurs renvoyées

Dreamweaver n'attend rien.

beginReporting()

Disponibilité

Dreamweaver 4.

Description

Cette fonction est appelée au début du processus de rapport, avant l'exécution du rapport. Si la commande Rapport renvoie la valeur `false` à partir de la fonction, la commande Rapport est exclue du processus de rapport.

Arguments

`target`

- L'argument *`target`* est une chaîne indiquant la cible de la session de rapport. Il peut prendre l'une des valeurs suivantes : `"CurrentDoc"`, `"CurrentSite"`, `"CurrentSiteSelection"` (pour les fichiers sélectionnés dans un site), ou `"Folder:+ *le chemin vers le dossier sélectionné par l'utilisateur*"` (par exemple, `"Folder:c:temp"`).

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le rapport s'exécute avec succès, `false` si *`target`* est exclu du processus de rapport.

endReporting()

Disponibilité

Dreamweaver 4.

Description

Cette fonction est appelée à la fin du processus de rapport.

Arguments

Néant

Valeurs renvoyées

Dreamweaver n'attend rien.

commandButtons()

Disponibilité

Dreamweaver 4.

Description

Définit les boutons devant figurer dans la partie droite de la boîte de dialogue Options et leur comportement lorsque l'utilisateur clique dessus. Si cette fonction n'est pas définie, aucun bouton n'apparaît et la section `BODY` du fichier de rapport s'étend de façon à remplir la totalité de la boîte de dialogue.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau contenant un nombre pair d'éléments. Le premier élément est une chaîne contenant le libellé du premier bouton. Le deuxième élément est une chaîne de code JavaScript définissant le comportement du premier bouton lorsque l'utilisateur clique dessus. Les autres éléments définissent les boutons supplémentaires de la même manière.

Exemple

Dans l'exemple suivant, la fonction `commandButtons()` définit trois boutons : OK, Annuler et Aide.

```
function commandButtons(){
  return new Array("OK" , "doCommand()" , "Cancel" , ¬
```

```
  "window.close()" , "Help" , "showHelp()");
}
```

configureSettings()

Disponibilité

Dreamweaver 4.

Description

Détermine si le bouton Paramètres du rapport doit être activé dans la boîte de dialogue Rapports lorsqu'un rapport est sélectionné.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le bouton Paramètres de rapport doit être activé ; `false` dans le cas contraire.

windowDimensions()

Disponibilité

Dreamweaver 4.

Description

Définit les dimensions de la boîte de dialogue des paramètres. Si cette fonction n'est pas définie, les dimensions de la fenêtre sont calculées automatiquement.

REMARQUE	Ne définissez cette fonction que si vous souhaitez utiliser une boîte de dialogue Options ayant des dimensions supérieures à 640 x 480 pixels.

Arguments

`platform`

- La valeur de l'argument *`platform`* est soit `"macintosh"`, soit `"windows"`, selon la plate-forme utilisée par l'utilisateur.

Valeurs renvoyées

Dreamweaver attend une chaîne au format `"widthInPixels,heightInPixels"`.

Les dimensions renvoyées sont inférieures à la taille totale de la boîte de dialogue parce qu'elles n'incluent pas la zone des boutons Valider et Annuler. Si les dimensions renvoyées ne permettent pas de faire apparaître toutes les options, des barres de défilement s'affichent.

Exemple

Dans l'exemple suivant, la fonction `windowDimensions()` définit les dimensions de la boîte de dialogue Paramètres à 648 x 520 pixels :

```
function windowDimensions(){
  return "648,520";
}
```

CHAPITRE 11

Bibliothèques et éditeurs de balises

11

Les utilisateurs de Macromedia Dreamweaver 8 peuvent utiliser des éditeurs de balises pour insérer de nouvelles balises, modifier des balises existantes et accéder aux informations de référence sur les balises. Dreamweaver propose des éditeurs pour les langages suivants : HTML, ASP.Net, CFML, JRun et JSP. Vous pouvez personnaliser ces éditeurs et en créer de nouveaux, ou ajouter de nouvelles balises aux bibliothèques de balises.

Le sélecteur de balises utilise les informations enregistrées dans les bibliothèques de balises pour permettre aux utilisateurs de Dreamweaver de passer en revue les balises disponibles, de les sélectionner et de les insérer dans le document en cours.

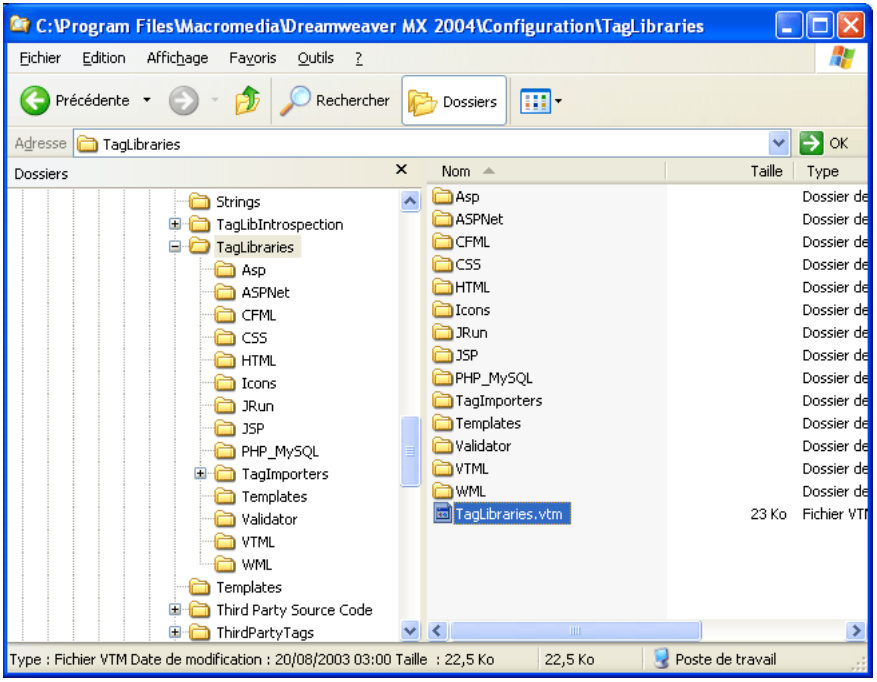
Dreamweaver enregistre les informations sur chaque balise, y compris leurs attributs, dans un groupe de sous-dossiers localisés dans le dossier Configuration/TagLibraries. Les fonctions de l'éditeur de balises et du sélecteur de balises utilisent les informations enregistrées dans ces dossiers lors de la manipulation et de la modification des balises. Avant de vous lancer dans la création d'éditeurs de balises personnalisés, vous devez comprendre la structure de la bibliothèque de balises.

Le tableau ci-dessous recense les fichiers utilisés pour créer une bibliothèque de balises :

Chemin	Fichier	Description
Configuration/TagLibraries/	TagLibraries.vtm	Recense chaque balise installée.
Configuration/TagLibraries/*language*/	*tag*.vtm	Inclut des informations sur les balises, tels les attributs correspondants, et indique si la balise est associée à une balise de fermeture, ainsi que les règles de mise en forme.
Configuration/TagLibraries/language/	*Tagimagefile*.gif	Fichier facultatif à afficher dans l'inspecteur de propriétés.

Format de fichier bibliothèque de balises

La bibliothèque de balises est composée d'un fichier racine, du fichier TagLibraries.vtm, contenant la liste des balises installées, et d'un fichier VTML pour chaque balise dans la bibliothèque. Le fichier TagLibraries.vtm fonctionne comme une table des matières et contient des pointeurs vers le fichier VTML de chaque balise. La figure suivante présente l'organisation des fichiers VTML dans Dreamweaver, par langage de balisage :



Les utilisateurs de Macromedia HomeSite peuvent reconnaître la structure des fichiers VTML, mais Dreamweaver n'utilise pas ces fichiers de la même manière que HomeSite. La différence la plus importante est que Dreamweaver contient son propre programme de rendu HTML pour l'affichage des interfaces utilisateur (UI) des extensions, et n'utilise donc pas les fichiers Dreamweaver VTML dans le processus de rendu de l'interface graphique utilisateur.

L'exemple suivant présente la structure du fichier TagLibraries.vtm :

```
<taglibraries>
<taglibrary name="Name of tag library" doctypes="HTML,ASP-JS,ASP-VB"
  tagchooser="relative path to TagChooser.xml file" id="DWTagLibrary_html">
    <tagref name="tag name" file="relative path to tag .vtm file"/>
</taglibrary>
```

```
<taglibrary name="CFML Tags" doctypes="ColdFusion" servermodel="Cold
  Fusion" tagchooser="cfml/TagChooser.xml" id="DWTagLibrary_cfml">
    <tagref name="cfabort" file="cfml/cfabort.vtm"/>
</taglibrary>

<taglibrary name="ASP.NET Tags" doctypes="ASP.NET_CSharp,ASP.NET_VB"¬
  servermodel="ASPNet" prefix="<asp:" tagchooser="ASPNet/TagChooser.xml"¬
  id="DWTagLibrary_aspnet">
    <tagref name="dataset" file="aspnet/dataset.vtm" prefix="<mm:dataset"/>
</taglibrary>
</taglibraries>
```

La balise `taglibrary` regroupe une ou plusieurs balises dans une bibliothèque de balises. Lorsque vous importez des balises ou créez un nouveau jeu de balises, vous pouvez les regrouper dans des bibliothèques de balises. Souvent, un regroupement `taglibrary` correspond à un ensemble de balises qui sont définies dans un fichier TLD de pages JSP (JavaServer Pages), un fichier DTD (Document Type Definition) de format XML, un espace de nom ASP.Net ou dans tout autre regroupement logique.

Le tableau suivant présente les attributs `taglibrary` :

Attribut	Description	Obligatoire/ facultatif
`taglibary.name`	Utilisé pour référencer la bibliothèque de balises dans l'interface utilisateur.	Obligatoire
`taglibrary.doctypes`	Indique les types de documents pour lesquels cette bibliothèque est active. Lorsque la bibliothèque est active, les balises de bibliothèque s'affichent dans le menu contextuel Indicateurs de code. Les bibliothèques de balises ne peuvent pas toutes être actives en même temps car des conflits de nom risquent de survenir (par exemple, les fichiers HTML et WML ne sont pas compatibles).	Obligatoire
`taglibrary.prefix`	Lorsque cet attribut est spécifié, les balises dans la bibliothèque de balises prennent la forme `taglibrary.prefix + tagref.name`. Par exemple, si `taglibrary.prefix` est `"<jrun:"` et `tagref.name` est `"if"`, la balise prend la forme `"<jrun:if"`. Cet attribut peut être ignoré pour une balise définie.	Facultatif

Attribut	Description	Obligatoire/ facultatif
`taglibrary.servermodel`	Si les balises dans la bibliothèque de balises s'exécutent sur un serveur d'application, l'attribut `servermodel` identifie le modèle serveur de la balise. S'il s'agit de balises côté client (et non de balises côté serveur), l'attribut `servermodel` est ignoré. L'attribut `servermodel` est utilisé également pour vérifier les navigateurs cibles.	Facultatif
`taglibrary.id`	Cet attribut peut être une chaîne quelconque différente des attributs `taglibrary.ID` des autres bibliothèques de balises du fichier. Extension Manager utilise l'attribut ID afin de laisser aux fichiers MXP la possibilité d'insérer de nouveaux `taglibrary` et fichiers `tags` dans le fichier TagLibraries.vtm.	Facultatif
`taglibrary.tagchooser`	Un chemin relatif au fichier TagChooser.xml associé à cette bibliothèque de balises.	Facultatif

Le tableau suivant présente les attributs `tagref` :

Attribut	Description	Obligatoire/ facultatif
`tagref.name`	Utilisé pour référencer la balise dans l'interface utilisateur.	Obligatoire
`tagref.prefix`	Indique comment doit s'afficher la balise en mode Source. Lorsqu'il est utilisé, l'attribut `tagref.prefix` détermine le préfixe de la balise en cours. Lorsqu'il est défini, il remplace la valeur spécifiée pour l'attribut `taglibrary.prefix`.	Facultatif
`tagref.file`	Référence le fichier VTML de la balise.	Facultatif

L'attribut `tagref.prefix` pouvant remplacer la valeur de l'attribut `taglibrary.prefix`, la relation entre ces deux attributs peut prêter à confusion. Le tableau suivant montre la relation entre les attributs `taglibrary.prefix` et `tagref.prefix` :

L'attribut `taglibrary.prefix` **est-il défini ?**	**L'attribut** `tagref.prefix` **est-il défini ?**	**Préfixe de balise obtenu**
Non	Non	`'<' + tagref.name`
Oui	Non	`taglibrary.prefix + tagref.name`
Non	Oui	`tagref.prefix`
Oui	Oui	`tagref.prefix`

Pour définir les balises, Dreamweaver utilise une version modifiée du format de fichier VTML de Macromedia. L'exemple suivant présente tous les éléments que Dreamweaver doit utiliser pour définir une balise individuelle :

```
<tag name="input" bind="value" casesensitive="no" endtag="no">
  <tagformat indentcontents="yes" formatcontents="yes" nlbeforetag ¬
  nlbeforecontents=0 nlaftercontents=0 nlaftertag=1 />
  <tagdialog file = "input.HTM"/>
  <attributes>
    <attrib name="name"/>
    <attrib name="wrap" type="Enumerated">
       <attriboption value="off"/>
       <attriboption value="soft"/>
       <attriboption value="hard"/>
    </attrib>
    <attrib name="onFocus" casesensitive="yes"/>
    <event name="onFocus"/>
  </attributes>
</tag>
```

Le tableau suivant présente les attributs de définition des balises :

Attribut	Description	Obligatoire/ facultatif
`tag.bind`	Utilisé par le panneau Liaisons de données. Lorsque vous sélectionnez une balise de ce type, l'attribut `bind` indique l'attribut par défaut pour la liaison des données.	Facultatif
`tag.casesensitive`	Indique si le nom de la balise doit respecter la casse. Si le nom de la balise doit respecter la casse, il est inséré dans le document de l'utilisateur exactement comme spécifié dans la bibliothèque de balises. S'il n'est pas nécessaire de respecter la casse, le nom de la balise est inséré en utilisant la casse par défaut spécifiée dans l'onglet Format de code de la boîte de dialogue Préférences. Si vous ne spécifiez pas l'attribut `casesensitive`, la balise ne tient pas compte des majuscules et des minuscules.	Facultatif
`tag.endtag`	Spécifie si la balise dispose d'une balise d'ouverture et d'une balise de fermeture. Par exemple, la balise `input` ne comporte pas de balise de fermeture. Il n'existe pas de balise `/input` correspondante. Si la balise de fermeture est facultative, l'attribut `ENDTAG` doit être défini sur `Yes`. Spécifiez `xml` pour appliquer la syntaxe XML à une valise vide. Par exemple, `<tag name="foo" endtag="xml" tagtype="empty">` insère `<foo/>`.	Facultatif
`tagformat`	Indique les règles de mise en forme de la balise. Dans les versions antérieures à Dreamweaver MX, les règles de mise en forme étaient enregistrées dans le fichier SourceFormat.txt.	Facultatif
`tagformat.indentcontents`	Indique si le contenu de cette balise doit être mis en retrait.	Facultatif
`tagformat.formatcontents`	Indique si le contenu de cette balise doit être analysé. Cet attribut est défini sur `No` pour les balises telles que `SCRIPT` et `STYLE`, dans lesquelles le contenu n'est pas du code HTML.	Facultatif

Attribut	Description	Obligatoire/ facultatif
`tagformat.nlbeforetag`	Le nombre de caractères de nouvelle ligne devant être insérés avant cette balise.	Facultatif
`tagformat.nlbeforecontents`	Le nombre de caractères de nouvelle ligne devant être insérés avant le contenu de cette balise.	Facultatif
`tagformat.nlaftercontents`	Le nombre de caractères de nouvelle ligne devant être insérés après le contenu de cette balise.	Facultatif
`tagformat.nlaftertag`	Le nombre de caractères de nouvelle ligne devant être insérés après cette balise.	Facultatif
`attrib.name`	Le nom de l'attribut, tel qu'il apparaît dans le code source.	Obligatoire
`attrib.type`	S'il n'est pas défini, l'attribut `attrib.type` est `"text"`. Cet attribut peut prendre les valeurs suivantes : `TEXT`–contenu texte libre `ENUMERATED`–liste de valeurs énumérées `COLOR`–valeur de couleur (nom ou hex) `FONT`–nom de police ou famille de polices `STYLE`–attribut de styles CSS `CSSSTYLE`–nom de classe CSS `CSSID`–ID de classe CSS `FILEPATH` –chemin de fichier complet `DIRECTORY`–chemin de dossier `FILENAME`–nom de fichier uniquement `RELATIVEPATH` –représentation relative du chemin `FLAG` –attribut ON/OFF qui ne contient pas de valeur	Facultatif
`attrib.casesensitive`	Indique si le nom de l'attribut doit respecter la casse. Si l'attribut `CASESENSITIVE` est absent, le nom de l'attribut ne tient pas compte des majuscules et des minuscules.	Facultatif

REMARQUE Dans les versions antérieures à Dreamweaver MX, les informations sur les balises sont enregistrées dans le fichier Configuration/TagAttributeList.txt.

Sélecteur de balises

Le sélecteur de balises vous permet d'afficher les balises dans des groupes fonctionnels et d'accéder rapidement aux balises fréquemment utilisées. Pour ajouter une balise ou un groupe de balises dans le sélecteur de balises, vous devez d'abord les ajouter dans la bibliothèque de balises. Vous pouvez effectuer cette opération à l'aide de la boîte de dialogue Editeur de la bibliothèque de balises ou en installant une extension Dreamweaver, empaquetée dans un fichier MXP.

Fichiers Tagchooser.xml

Le fichier Tagchooser.xml contient les métadonnées utilisées pour organiser les regroupements de balises qui apparaissent dans le sélecteur de balises. Chaque balise livrée avec Dreamweaver est enregistrée dans un regroupement fonctionnel, et est disponible dans le sélecteur de balises. Vous pouvez regrouper les balises existantes et créer des groupes de nouvelles balises en modifiant le fichier TagChooser.xml. Vous pouvez également personnaliser la structure des balises en créant des sous-catégories pour permettre aux utilisateurs d'accéder facilement aux balises les plus importantes pour leur travail.

Le fichier TagLibraries.vtm prend en charge l'utilisation de l'attribut `TAGLIBRARY.TAGCHOOSER`, qui pointe vers le fichier Tagchooser.xml. Si vous modifiez les fichiers Tagchooser.xml existants ou si vous en créez de nouveaux, l'attribut `TAGLIBRARY.TAGCHOOSER` doit pointer vers l'emplacement exact du sélecteur de balises pour pouvoir fonctionner correctement.

S'il n'existe pas d'attribut `TAGLIBRARY.TAGCHOOSER`, le sélecteur de balises affiche l'arborescence définie dans le TagLibraries.vtm.

Les fichiers TagChooser.xml résident dans le dossier Configuration/TagLibraries/*TagLibraryName*. L'exemple suivant présente la structure des fichiers TagChooser.xml :

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<tclibrary name="Friendly name for library node" desc='Description for
  incorporated reference' reference="Language[,Topic[,Subtopic]]">
  <category name="Friendly name for category node" desc='Description for
  incorporated reference' reference="Language[,Topic[,Subtopic]]"
  id="Unique id">
   <category name="Friendly name for subcategory node" ICON="Relative path"
  desc='Description for incorporated reference'
  reference="Language,Topic[,Subtopic]" id="Unique id">
      <element name="Friendly name for list item" value='Value to pass to
  visual dialog editors' desc='Description for incorporated reference'
  reference="Language[,Topic[,Subtopic]]" id="Unique id"/>
      ... more elements to display in the list view ...
    </category>
```

```
    ... more subcategories ...
  </category>
  ... more categories ...
</tclibrary>
```

Le tableau suivant présente les balises pouvant être utilisées dans les fichiers TagChooser.xml :

Balise	Description	Obligatoire/ facultatif
`tclibrary`	Balise extérieure qui encapsule cette structure de sélecteur de balises de la bibliothèque de balises.	Obligatoire
`tclibrary.name`	Cette valeur s'affiche dans le nœud Arborescence.	Obligatoire
`tclibrary.desc`	Cette valeur est une chaîne HTML qui s'affiche dans la section Infos sur les balises de la boîte de dialogue Sélecteur de balises. S'il n'existe pas d'attribut `DESC`, les informations Infos sur les balises proviennent du panneau Référence. Interchangeable avec `tclibrary.reference`.	Facultatif (`desc` et `reference` sont mutuellement exclusives)
`tclibrary.reference`	Cette valeur décrit le langage, la rubrique et la sous-rubrique à afficher dans la section Infos sur les balises de la boîte de dialogue Sélecteur de balises. Interchangeable avec `tclibrary.desc`.	Facultatif (`desc` et `reference` sont mutuellement exclusives)

La balise `CATEGORY` représente tous les autres nœuds dans l'arborescence sous le nœud de la balise `TCLIBRARY`, comme illustré dans le tableau suivant :

Balise	Description	Obligatoire/ facultatif
`category.name`	Cette valeur s'affiche dans le nœud Arborescence.	Obligatoire
`category.desc`	Cette valeur est une chaîne HTML qui s'affiche dans la section Infos sur les balises de la boîte de dialogue Sélecteur de balises. Si `desc` ou `reference attr` ne sont pas spécifiés, rien ne s'affiche dans la section Infos sur les balises.	Facultatif (`desc` et `reference` sont mutuellement exclusives)
`category.reference`	Cette valeur décrit le langage, la rubrique et la sous-rubrique à afficher dans la section Infos sur les balises.	Facultatif (`desc` et `reference` sont mutuellement exclusives)

Balise	Description	Obligatoire/ facultatif
`category.icon`	Cette valeur est un chemin relatif vers un fichier GIF d'icône.	Facultatif
`category.id`	Toute chaîne différente des attributs `category.id` des autres catégories dans ce fichier.	Obligatoire

Le tableau suivant répertorie les attributs de la balise `ELEMENT`, qui représente la balise à insérer :

Attribut	Description	Obligatoire/ facultatif
`element.name`	Cette valeur s'affiche comme un élément de liste.	Obligatoire
`element.value`	Une valeur placée directement dans le code ou un paramètre qui est transmis vers les boîtes de dialogue visuelles.	Obligatoire
`element.desc`	Cette valeur est une chaîne HTML et s'affiche dans le panneau Référence incorporé. S'il n'est pas spécifié, l'attribut `REFERENCE` affiche le contenu de la référence dans le panneau Référence incorporé.	Facultatif (`desc` et `reference` sont mutuellement exclusives)
`element.reference`	Jusqu'à trois chaînes, séparées par des virgules, décrivant respectivement le langage, la rubrique et la sous-rubrique. Ces informations s'affichent dans le panneau Référence. La première chaîne est obligatoire. La deuxième chaîne est obligatoire pour la balise `ELEMENT` seulement et facultative pour les balises `CATEGORY` et `TCLIBRARY`. La troisième chaîne est facultative.	Facultatif (`desc` et `reference` sont mutuellement exclusives)
`element.id`	Toute chaîne différente des attributs `element.id` des autres éléments dans ce fichier.	Facultatif

Exemple simple de création d'un éditeur de balise

Les exemples dans cette section utilisent `cfweather`, une balise ColdFusion conçue pour extraire la température courante à partir d'une base de données météorologiques, et illustrent les étapes nécessaires à la création d'un nouvel éditeur de balises.

Les attributs de la balise `cfweather` sont décrits dans le tableau suivant :

Attribut	Description
`zip`	Un code postal de cinq chiffres
`tempaturescale`	L'échelle de température (Celsius ou Fahrenheit)

Pour créer cet éditeur de balise, exécutez les étapes suivantes :

- Enregistrement de la balise dans la bibliothèque de balises
- Création d'un fichier de définition de balise (VTML)
- Création d'une interface utilisateur d'éditeur de balises
- Ajout d'une balise au sélecteur de balises

Enregistrement de la balise dans la bibliothèque de balises

Afin que Dreamweaver puisse reconnaître la nouvelle balise, celle-ci doit être identifiée dans le fichier TagLibraries.vtm, localisé dans le dossier Configuration/TagLibraries. Néanmoins, sur une plate-forme multi-utilisateurs (par exemple, Windows XP, Windows 2000, Windows NT ou Mac OS X), un autre fichier TagLibraries.vtm réside dans le dossier Configuration de l'utilisateur. C'est ce fichier qui doit être mis à jour car il s'agit de l'instance que Dreamweaver recherche et analyse.

L'emplacement du dossier Configuration varie selon la plate-forme utilisateur.

Sous Windows 2000 et Windows XP :

```
<drive>:\Documents and Settings\<username>\ ¬
  Application Data\Macromedia\Dreamweaver 8\Configuration
```

REMARQUE

Sous Windows XP, ce dossier peut se trouver dans un dossier masqué.

Pour Mac OS X :

```
<drive>:Users:<username>:Library:Application Support: ¬
  Macromedia:Dreamweaver 8:Configuration
```

Si Dreamweaver ne parvient pas à trouver le fichier TagLibraries.vtm dans le dossier Configuration de l'utilisateur, il effectue une recherche dans le dossier Configuration de Dreamweaver.

REMARQUE	Sur les plates-formes multiutilisateur, si vous modifiez la copie du fichier TagLibraries.vtm résidant dans le dossier Configuration de Dreamweaver, au lieu de modifier celle qui se trouve dans le dossier de configuration de l'utilisateur, Dreamweaver ne prend pas en charge les modifications. En effet, Dreamweaver analyse la copie du fichier TagLibraries.vtm située dans le dossier de configuration de l'utilisateur, et non pas celle située dans le dossier Configuration de Dreamweaver.

`cfweather` est une balise ColdFusion. Vous pouvez donc enregistrer la balise `cfweather` dans le groupe de bibliothèque de balises correspondant qui existe déjà.

Pour enregistrer la balise :

1. Ouvrez le fichier TagLibraries.vtm dans l'éditeur de texte.
2. Faites défiler les bibliothèques de balises existantes jusqu'à localiser le groupe `taglibrary` de balises CFML.
3. Ajoutez un élément de référence de balise, comme indiqué dans l'exemple suivant :

   ```
   <tagref name="cfweather" file="cfml/cfweather.vtm"/>
   ```

4. Enregistrez le fichier.

   La balise est maintenant enregistrée dans la bibliothèque de balises et contient un pointeur vers le fichier de définition de balise cfweather.vtm.

Création d'un fichier de définition de balise (VTML)

Lorsqu'un utilisateur sélectionne une balise enregistrée à l'aide du sélecteur de balises ou d'un éditeur de balises, Dreamweaver recherche un fichier VTML correspondant pour la définition de la balise.

Pour créer un fichier de définition de balise :

1. Dans un éditeur de texte, créez un fichier contenant les éléments suivants :

```
<TAG NAME="cfweather" endtag="no">
        <TAGFORMAT NLBEFORETAG="1" NLAFTERTAG="1"/>
        <TAGDIALOG FILE="cfweather.htm"/>

        <ATTRIBUTES>
          <ATTRIB NAME="zip" TYPE="TEXT"/>
          <ATTRIB NAME="tempaturescale" TYPE="ENUMERATED">
            <ATTRIBOPTION VALUE="Celsius"/>
            <ATTRIBOPTION VALUE="Fahrenheit"/>
          </ATTRIB>
```

```
        </ATTRIBUTES>
</TAG>
```

2. Enregistrez le fichier cfweather.vtm dans le dossier Configuration/Taglibraries/CFML.

   Le fichier de définition de balise permet à Dreamweaver d'exécuter les opérations de code contextuel, de finalisation de code et de mise en forme pour la balise `cfweather`.

Création d'une interface utilisateur d'éditeur de balises

Pour créer l'interface utilisateur de l'éditeur de balises cfweather :

1. Enregistrez le fichier cfweather.htm dans le dossier Configuration/Taglibraries/CFML :

```
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine 5.0//
  dialog">
<html>
<head>
<title>CFWEATHER</title>
<script src="../../Shared/Common/Scripts/dwscripts.js"></script>
<script src="../../Shared/Common/Scripts/ListControlClass.js"></script>
<script src="../../Shared/Common/Scripts/tagDialogsCmn.js"></script>
<script>

/************************ GLOBAL VARS ************************/
var TEMPATURESCALELIST;   // tempaurelist control (initialized in
  initializeUI())
var theUIObjects;         // array of UI objects used by common API
  functions

/*************************************************************/

// inspectTag() API function defined (required by all tag editors)
function inspectTag(tagNodeObj)
{
  // call into a common library version of inspectTagCommon defined
  // in tagDialogCmns.js (note that it's been included)
  // For more information about this function, look at the comments
  // for inspectTagCommon in tagDialogCmn.js
  tagDialog.inspectTagCommon(tagNodeObj, theUIObjects);
}

function applyTag(tagNodeObj)
{
  // call into a common library version of applyTagCommon defined
  // in tagDialogCmns.js (note that it's been included)
  // For more information about this function, look at the comments
  // for applyTagCommon in tagDialogCmn.js
  tagDialog.applyTagCommon(tagNodeObj, theUIObjects);
```

```
}

function initializeUI()
{
  // define two arrays for the values and display captions for the list
  control
  var theTempatureScaleCap = new Array("celsius","fahrenheit");
  var theTempatureScaleVal = new Array("celsius","fahrenheit");

  // instantiate a new list control
  TEMPATURESCALELIST = new ListControl("thetempaturescale");

  // add the tempaturescalelist dropdown list control to the uiobjects
  theUIObjects  = new Array(TEMPATURESCALELIST);

  // call common populateDropDownList function defined in tagDialogCmn.js to
  // populate the tempaturescale list control
  tagDialog.populateDropDownList(TEMPATURESCALELIST, theTempatureScaleCap,
  theTempatureScaleVal, 1);
}
</script>

</head>
<body onLoad="initializeUI()">
<div name="General">
  <table border="0" cellspacing="4">
    <tr>
      <td valign="baseline" align="right" nowrap="nowrap">Zip Code: </td>
      <td nowrap="nowrap">
        <input type="text" id="attr:cfargument:zip" name="thezip"
  attname="zip" style="width:100px"  /> 
      </td>
    </tr>
    <tr>
      <td valign="baseline" align="right" nowrap="nowrap">Type: </td>
      <td nowrap="nowrap">
          <select name="thetempaturescale"
  id="attr:cfargument:tempaturescale" attname="tempaturescale"
  editable="false" style="width:200px">
          </select>
      </td>
    </tr>
  </table>
</div>
</body>
</html>
```

2. Vérifiez que l'éditeur de balises fonctionne correctement en exécutant la procédure suivante :

   - Lancez Dreamweaver.

- Entrez `cfweather` en mode Code.
- Cliquez sur la balise avec le bouton droit de la souris.
- Sélectionnez Modifier la balise `cfweather` dans le menu contextuel.

Si l'éditeur de balises démarre, il a été créé correctement.

Ajout d'une balise au sélecteur de balises

Pour ajouter la balise `cfweather` au sélecteur de balises :

1. Modifiez le fichier TagChooser.xml dans le dossier Configuration/Taglibraries/CFML en ajoutant une nouvelle catégorie appelée Balises tierces, qui met en valeur la balise `cfweather`, comme indiqué dans l'exemple suivant :

```
<category name="Third Party Tags" icon="icons/Elements.gif"
  reference='CFML'>
  <element name="cfweather" value='cfweather zip=""
  temperaturescale="fahrenheit">' />
</category>
```

REMARQUE

Sur les plates-formes multiutilisateur, le fichier Tagchooser.xml existe également dans le dossier de configuration de l'utilisateur. Pour plus d'informations sur les plates-formes multiutilisateurs, voir *Enregistrement de la balise dans la bibliothèque de balises*, page 283.

2. Vérifiez que la balise `cfweather` s'affiche correctement dans le sélecteur de balises en exécutant la procédure suivante :
   - Sélectionnez Insertion > Balise.
   - Développez le groupe Balises CFML.
   - Sélectionnez le groupe Balises tierces qui s'affiche au bas du sélecteur de balises.
   - La balise `cfweather` s'affiche dans la zone de liste sur la droite.
   - Sélectionnez cfweather, puis cliquez sur le bouton Insérer.

   L'éditeur de balises doit apparaître.

API de l'éditeur de balises

Afin de créer un nouvel éditeur de balises, vous devez fournir une implémentation pour les fonctions `inspectTag()`, `validateTag()` et `applyTag()`. Pour avoir un exemple d'implémentation, voir *Création d'une interface utilisateur d'éditeur de balises*, page 285.

inspectTag()

Disponibilité

Dreamweaver MX.

Description

La fonction est appelée au premier affichage de l'éditeur de balises. La fonction reçoit la balise modifiée par l'utilisateur comme un argument sous la forme d'un objet `dom`. La fonction extrait des valeurs d'attribut de la balise en cours d'édition et utilise ces valeurs pour initialiser les éléments de formulaire dans l'éditeur de balises.

Arguments

`balise`

- L'argument `tag` constitue le nœud DOM de la balise modifiée.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

Supposons que l'utilisateur modifie la balise suivante :

```
<crfweather zip = “94065”/>
```

Si l'éditeur contient une zone de texte de modification de l'attribut `zip`, la fonction doit initialiser l'élément de formulaire pour permettre à l'utilisateur de voir le code postal dans le champ de texte, plutôt que de voir le champ vide.

Le code suivant exécute l'initialisation :

```
function inspectTag(tag)
{
  document.forms[0].zip.value = tag.zip
}
```

validateTag()

Disponibilité

Dreamweaver MX.

Description

Lorsqu'un utilisateur clique sur un nœud dans la commande d'arborescence ou sur OK, la fonction exécute la validation des entrées sur les éléments de formulaire HTML affichés actuellement.

Arguments

Aucun.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'entrée pour les éléments de formulaire HTML est valide ; `false` si les valeurs d'entrée ne sont pas valides.

Exemple

Lors de la création d'un tableau, l'utilisateur entre un nombre entier négatif pour le nombre de lignes. La fonction `validateTag()` détecte l'entrée non valide, affiche un message d'alerte et renvoie la valeur `false` .

applyTag()

Disponibilité

Dreamweaver MX.

Description

Lorsque l'utilisateur clique sur OK, Dreamweaver appelle la fonction `validateTag()`. Si la fonction `validateTag()` renvoie la valeur `true`, Dreamweaver appelle cette fonction et transmet l'objet `dom` représentant la balise en cours de modification. La fonction interprète les valeurs des éléments de formulaires et les enregistre dans l'objet `dom`.

Arguments

`balise`

- L'argument `tag` constitue le nœud DOM de la balise modifiée.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

Reprenons l'exemple `cfweather`. Dans le code suivant, si l'utilisateur remplace le code postal 94065 par 53402, afin de mettre à jour le document de l'utilisateur de sorte qu'il utilise le nouveau code postal, vous devez également mettre à jour l'objet `dom`.

```
function applyTag(tag)
{
   tag.zip = document.forms[0].zip.value
}
```

CHAPITRE 12

Inspecteurs de propriétés

12

L'inspecteur de propriétés est probablement le panneau flottant le plus connu de l'interface Macromedia Dreamweaver 8. Il est indispensable pour définir, vérifier et modifier le nom, les dimensions, l'aspect et d'autres attributs de la sélection ; il permet également de lancer des éditeurs internes et externes relatifs à l'élément sélectionné.

Dreamweaver intègre plusieurs interfaces compatibles avec l'inspecteur de propriétés permettant de définir des propriétés pour de nombreuses balises HTML standard. Parce que ces inspecteurs intégrés font partie du code Dreamweaver élémentaire, le dossier Configuration ne contient pas de fichiers d'inspecteur de propriétés correspondants. Cependant, les fichiers d'inspecteur de propriétés personnalisés permettent d'ignorer ces interfaces intégrées ou d'en créer de nouvelles pour contrôler les balises personnalisées. Les fichiers d'inspecteur de propriétés personnalisés résident dans le dossier Configuration/Inspectors du dossier de l'application Dreamweaver.

Le tableau ci-dessous recense les fichiers utilisés pour créer un inspecteur de propriétés :

Chemin	Fichier	Description
Configuration/Inspectors/	*Nom_inspecteur_propriétés*.htm	Définit l'interface utilisateur (UI) de l'inspecteur de propriétés.
Configuration/Inspectors/	*Nom_inspecteur_propriétés*.js	Contient les fonctions requises par l'inspecteur de propriétés.
Configuration/Inspectors/	*Fichier_image_balise*.gif	Fichier facultatif à afficher dans l'inspecteur de propriétés.

Fichiers d'inspecteur de propriétés

Le fichier HTML d'inspecteur de propriétés doit contenir un commentaire (en plus du commentaire doctype), juste devant la balise HTML d'ouverture, comme indiqué dans l'exemple suivant :

```
<!-- tag:serverModel:tagNameOrKeyword,priority:1to10,selection:¬
```

```
exactOrWithin,hline,vline, serverModel-->
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine5.0//pi">
```

Ce commentaire comporte les éléments suivants :

- L'élément *serverModel* indique que Dreamweaver ne doit charger cet inspecteur de propriétés que lorsque le modèle de serveur spécifié est actif.
- *tagNameOrKeyword* est la balise à inspecter ou l'un des mots clefs suivants : `*COMMENT*` (pour les commentaires), `*LOCKED*` (pour les régions verrouillées) ou `*ASP*` (pour les balises `ASP`).
- *1to10* est la priorité du fichier d'inspecteur de propriétés : `1` indique que cet inspecteur doit être utilisé uniquement lorsque aucun autre ne peut inspecter la sélection ; `10` indique que cet inspecteur a priorité sur tous les autres pouvant inspecter la sélection.
- L'élément *exactOrWithin* indique si la sélection peut être contenue dans la balise (`within`) ou si elle doit contenir exactement la balise (`exact`).
- L'élément `hline` (facultatif) indique qu'une ligne grise horizontale doit apparaître entre les moitiés supérieure et inférieure de l'inspecteur en mode étendu.
- L'élément `vline` (facultatif) indique qu'une ligne grise verticale doit apparaître entre le champ de nom de balise et le reste des propriétés dans l'inspecteur.
- L'élément *serverModel* (facultatif) indique le modèle de serveur de l'inspecteur de propriétés. Si le modèle de serveur de l'inspecteur de propriétés n'est pas le même que le modèle de serveur pour le document, Dreamweaver n'utilise pas l'inspecteur de propriétés pour afficher les propriétés de la sélection en cours. Ainsi, si le modèle de serveur du document correspond à Macromedia ColdFusion, mais que le modèle de serveur de l'inspecteur de propriétés est ASP, Dreamweaver n'utilise pas cet inspecteur de propriétés pour les sélections dans le document.

Le commentaire suivant est approprié pour un inspecteur conçu pour inspecter la balise `HAPPY` :

```
<!-- tag:HAPPY, priority:8,selection:exact,hline,vline, ¬
serverModel:ASP -->
```

Dans certains cas, vous pouvez spécifier que votre extension utilise exclusivement le rendu d'extension Dreamweaver (et non pas le moteur de rendu précédent) en insérant la ligne suivante immédiatement avant le commentaire de balise, comme dans l'exemple suivant :

```
<!--DOCTYPE HTML SYSTEM “-//Macromedia//DWExtension layout-engine 5.0//pi”-
  ->
```

La section `BODY` du fichier d'inspecteur de propriétés contient un formulaire HTML. Toutefois, au lieu d'afficher le contenu du formulaire dans une boîte de dialogue, Dreamweaver utilise le formulaire pour définir les zones d'entrée et la mise en page de l'inspecteur de propriétés.

La section HEAD d'un fichier d'inspecteur de propriétés contient des fonctions JavaScript ou une référence au(x) fichier(s) JavaScript.

Fonctionnement des fichiers d'inspecteur de propriétés

Au démarrage, Dreamweaver lit la première ligne de chaque fichier HTM et HTML dans le dossier Configuration/Inspectors, recherchant la chaîne de commentaire qui définit le type, la priorité et le type de sélection d'un inspecteur de propriétés. Les fichiers qui ne possèdent pas ce commentaire en première ligne sont ignorés.

Lorsque l'utilisateur effectue une sélection dans Dreamweaver ou déplace le point d'insertion vers un emplacement différent, les événements suivants se produisent :

1. Dreamweaver recherche tout inspecteur possédant un type de sélection `within`.
2. S'il n'existe aucun inspecteur `within`, Dreamweaver recherche l'arborescence du document à partir de la balise actuellement sélectionnée pour vérifier s'il existe des inspecteurs pour les autres balises entourant la sélection. S'il n'existe aucun inspecteur `within`, Dreamweaver recherche tous les inspecteurs possédant le type de sélection `exact`.
3. Pour la première balise possédant un ou plusieurs inspecteurs, Dreamweaver appelle chaque fonction `canInspectSelection()` de l'inspecteur. Si la fonction renvoie la valeur `false`, Dreamweaver ne considère plus l'inspecteur comme un candidat pour inspecter cette sélection.
4. S'il reste plusieurs inspecteurs potentiels après avoir appelé la fonction `canInspectSelection()`, Dreamweaver trie les inspecteurs restant en fonction de leur priorité.
5. Si plusieurs inspecteurs potentiels ont le même degré de priorité, Dreamweaver sélectionne l'inspecteur suivant l'ordre alphabétique.
6. L'inspecteur choisi apparaît dans le panneau flottant Inspecteur de propriétés. Si le fichier d'inspecteur de propriétés définit la fonction `displayHelp()`, une petite icône en forme de point d'interrogation (?) apparaît dans le coin supérieur droit de l'inspecteur.
7. Dreamweaver appelle la fonction `inspectSelection()` pour réunir des informations sur la sélection en cours et renseigner les champs de l'inspecteur.
8. Les gestionnaires d'événements associés aux champs dans l'interface de l'inspecteur de propriétés s'exécutent au fur et à mesure que l'utilisateur les rencontre (par exemple, vous pouvez avoir un événement `onBlur` qui appelle la fonction `setAttribute()` pour définir un attribut selon la valeur saisie par l'utilisateur).

Exemple simple d'inspecteur de propriétés

L'inspecteur de propriétés suivant inspecte la balise `MARQUEE`, réservée à Microsoft Internet Explorer. L'exemple vous permet de définir la valeur de l'attribut `direction` dans l'inspecteur de propriétés. Pour définir la valeur des autres attributs de la balise `MARQUEE`, servez-vous de cet exemple comme d'un modèle.

Vous créez cette extension en exécutant les étapes suivantes :

- Création de l'interface utilisateur
- Ecriture du code JavaScript
- Création de l'image
- Test de l'inspecteur de propriétés

Création de l'interface utilisateur

Vous créez un fichier HTML contenant un formulaire, qui s'affiche dans l'inspecteur de propriétés.

Pour créer l'interface utilisateur :

1. Créez un document vierge.
2. Sur la première ligne du fichier, ajoutez le commentaire suivant, qui identifie l'inspecteur de propriétés :

   ```
   <!-- tag:MARQUEE,priority:9,selection:exact,vline,hline -->
   ```

3. Pour spécifier le titre du document et le fichier JavaScript à créer, insérez le code suivant après le commentaire :

   ```
   <HTML>
   <HEAD>
   <TITLE>Marquee Inspector</TITLE>
   <SCRIPT src="marquee.js"></SCRIPT>
   </HEAD>
   <BODY>

   </BODY>
   </HTML>
   ```

4. Pour indiquer les éléments qui s'affichent dans l'inspecteur de propriétés, insérez le code suivant entre les balises `BODY` d'ouverture et de fermeture :

   ```
   <!-- Specify the image that will appear in the Property inspector -->
   <SPAN ID="image" STYLE="position:absolute; width:23px; height:17px; ¬
     z-index:16; left: 3px; top: 2px">
   ```

```
  <IMG SRC="marquee.png" WIDTH="36" HEIGHT="36" NAME="marqueeImage">
</SPAN>
<SPAN ID="label" STYLE="position:absolute; width:23px; height:17px; ¬
  z-index:16; left: 44px; top: 5px">Marquee</SPAN>

<!-- If your form fields are in different layers, you must ¬
  create a separate form inside each layer and reference it as ¬
  shown in the inspectSelection() and setInterjectionTag() ¬
  functions. -->

<SPAN ID="topLayer" STYLE="position:absolute; z-index:1; left: 125px; ¬
  top: 3px; width: 431px; height: 32px">
<FORM NAME="topLayerForm">
  <TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0">
    <TR>
      <TD VALIGN="baseline" ALIGN="right">Direction:</TD>
      <TD VALIGN="baseline" ALIGN="right">
      <SELECT NAME="marqDirection" STYLE="width:86"
        onChange="setMarqueeTag()">
        <OPTION VALUE="left">Left</OPTION>
        <OPTION VALUE="right">Right</OPTION>
      </SELECT>
    </TR>
  </TABLE>
</FORM>
</SPAN>
```

5. Enregistrez le fichier sous le nom marquee.htm dans le dossier Configuration/Inspectors.

Ecriture du code JavaScript

Vous devez ajouter des fonctions JavaScript pour vous assurer qu'il sera possible d'inspecter la sélection, pour inspecter la sélection et pour entrer les valeurs appropriées depuis l'inspecteur de propriétés.

Pour rédiger le code JavaScript :

1. Créez un document vierge.
2. Pour spécifier que l'inspecteur de propriétés s'affiche lorsque la sélection contient la balise `MARQUEE`, ajoutez la fonction suivante :

```
function canInspectSelection(){
  return true;
}
```

3. Pour actualiser la valeur de l'attribut direction qui s'affiche dans le champ de texte, ajoutez la fonction suivante en fin de fichier :

```
function inspectSelection(){
  // Get the DOM of the current document.
```

```
  var theDOM = dw.getDocumentDOM();
  // Get the selected node.
  var theObj = theDOM.getSelectedNode();

  // Get the value of the DIRECTION attribute on the MARQUEE tag.
  var theDirection = theObj.getAttribute('direction');

  // Initialize a variable for the DIRECTION attribute to -1.
  // This is used to store the menu index that corresponds to
  // the value of the attribute.
  // var typeIndex = -1;
  var directionIndex = -1;

  // If there was a DIRECTION attribute...
  if (theDirection){
    // If the value of DIRECTION is "left", set typeIndex to 0.
    if (theDirection.toLowerCase() == "left"){
      directionIndex = 0;
    // If the value of DIRECTION is "right", set typeIndex to 1.
      }else if (theDirection.toLowerCase() == "right"){
        directionIndex = 1;
       }
      }
  // If the value of the DIRECTION attribute was "left"
  // or "right", choose the corresponding
  // option from the pop-up menu in the interface.
  if (directionIndex != -1){
    document.topLayer.document.topLayerForm.marqDirection.selectedIndex
  =¬ directionIndex;
  }
}
```

4. Pour extraire la sélection actuelle et afficher dans la zone de texte de l'inspecteur de propriétés la valeur de l'attribut `direction`, ajoutez la fonction suivante en fin de fichier :

```
function setMarqueeTag(){
  // Get the DOM of the current document.
  var theDOM = dw.getDocumentDOM();
  // Get the selected node.
  var theObj = theDOM.getSelectedNode();

  // Get the index of the selected option in the pop-up menu
  // in the interface.
  var directionIndex = ¬
  document.topLayer.document.topLayerForm.marqDirection.selectedIndex;
  // Get the value of the selected option in the pop-up menu
  // in the interface.
  var theDirection = ¬
    document.topLayer.document.topLayerForm.marqDirection.¬
    options[directionIndex].value;
```

```
    // Set the value of the direction attribute to theDirection.
    theObj.setAttribute('direction',theDirection);
}
```

5. Enregistrez le fichier sous le nom marquee.js dans le dossier Configuration/Inspectors.

Création de l'image

Le cas échéant, vous pouvez créer l'image qui s'affiche dans l'inspecteur de propriétés.

Pour créer l'image :

1. Créez une image de 36 pixels de large sur 36 pixels de haut.
2. Enregistrez l'image sous le nom marquee.gif dans Configuration/Inspectors.

   En règle générale, vous pouvez enregistrer les images associées aux inspecteurs de propriétés dans tout format pris en charge par Dreamweaver.

Test de l'inspecteur de propriétés

Vous pouvez enfin tester l'inspecteur de propriétés.

Pour tester l'inspecteur de propriétés :

1. Redémarrez Dreamweaver.
2. Créez une page HTML ou ouvrez une page HTML existante.
3. Ajoutez le code suivant dans la section `BODY` de la page :

   `<MARQUEE></MARQUEE>`
4. Affichez en surbrillance le texte que vous venez d'ajouter.

   L'inspecteur de propriétés que vous avez créé pour la balise `MARQUEE` apparaît.
5. Entrez la valeur de l'attribut Direction dans l'inspecteur de propriétés.

   La balise affichée sur votre page est modifiée ; elle inclut à présent l'attribut direction et la valeur entrée dans l'inspecteur de propriétés.

API de l'inspecteur de propriétés

Deux des fonctions de l'API de l'inspecteur de propriétés (`canInspectSelection()` et `inspectSelection()`) sont obligatoires.

canInspectSelection()

Description

Détermine si l'inspecteur de propriétés est approprié pour la sélection en cours.

Arguments

Néant

REMARQUE	Utilisez `dom.getSelectedNode()` pour obtenir la sélection en cours en tant qu'objet JavaScript (pour plus d'informations sur `dom.getSelectedNode()`, voir le *Guide des API de Dreamweaver*).

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'inspecteur peut inspecter la sélection en cours, et `false` dans le cas contraire.

Exemple

L'instance suivante de la fonction `canInspectSelection()` renvoie la valeur `true` si la sélection contient l'attribut `CLASSID` et si la valeur de cet attribut est `"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"` (l'ID de classe pour Macromedia Flash Player) :

```
function canInspectSelection(){
  var theDOM = dw.getDocumentDOM();
  var theObj = theDOM.getSelectedNode();
  return (theObj.nodeType == Node.ELEMENT_NODE && ¬
    theObj.hasAttribute("classid") && ¬
    theObj.getAttribute("classid").toLowerCase()== ¬
    "clsid:D27CDB6E-AE6D-11cf-96B8-444553540000");
}
```

displayHelp()

Description

Si cette fonction est définie, une icône en forme de point d'interrogation (?) apparaît dans le coin supérieur droit de l'inspecteur de propriétés. Cette fonction est appelée lorsque l'utilisateur clique sur l'icône.

Arguments

Néant

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

L'occurrence suivante de `displayHelp()` ouvre un fichier dans une fenêtre du navigateur. Ce fichier explique les champs de l'inspecteur de propriétés :

```
function displayHelp(){
  dw.browseDocument('http://www.hooha.com/dw/inspectors/inspHelp.html');
}
```

inspectSelection()

Description

Actualise le contenu des zones de texte en fonction des attributs de la sélection en cours.

Arguments

maxOrMin

- L'argument *maxOrMin* est `max` ou `min`, selon que l'inspecteur de propriétés est en état développé ou compressé.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

L'occurrence suivante de la fonction `inspectSelection()` récupère la valeur de l'attribut `CONTENT` et l'utilise pour remplir un champ de formulaire appelé `keywords` :

```
function inspectSelection(){
  var dom = dreamweaver.getDocumentDOM();
  var theObj = dom.getSelectedNode();
  document.forms[0].keywords.value = ¬
  theObj.getAttribute("content");
}
```

CHAPITRE 13

Panneaux flottants

13

Vous pouvez créer tous types de panneaux flottants ou d'inspecteurs sans les limitations de taille ou de mise en page de l'inspecteur de propriétés.

Un inspecteur de propriétés personnalisé doit rester votre premier choix pour définir les propriétés de la sélection en cours. Toutefois, les panneaux flottants personnalisés offrent plus d'espace et de souplesse pour afficher des informations relatives à l'ensemble du document ou à plusieurs sélections.

Les fichiers de panneaux flottants personnalisés sont des fichiers HTML qui résident dans le dossier Configuration/Floaters du dossier de l'application Macromedia Dreamweaver 8. La section `BODY` d'un fichier de panneau flottant contient un formulaire HTML. Des gestionnaires d'événements associés aux éléments du formulaire peuvent appeler un code JavaScript pour effectuer des modifications arbitraires dans le document actif.

Dreamweaver possède plusieurs panneaux flottants intégrés accessibles depuis le menu Fenêtre (ces panneaux intégrés font partie du code principal de Dreamweaver ; vous ne trouverez donc pas de fichier de panneau flottant leur correspondant dans le dossier Configuration/Floaters).

Vous pouvez créer et ajouter vos propres panneaux au menu Fenêtre. Pour plus d'informations sur l'ajout d'éléments au système de menus, voir Chapitre 8, “Menus et commandes de menu,” on page 191.

Le tableau ci-dessous recense les fichiers utilisés pour créer un panneau flottant :

Chemin	Fichier	Description
Configuration/Floaters/	*nom_panneau*.htm	Spécifie le texte qui s'affiche dans la barre de titre du panneau flottant, définit ce dernier et contient les fonctions JavaScript requises.
Configuration/Menus/	menus.xml	Ajoute une commande à un menu.

Fonctionnement des fichiers de panneau flottant

Les panneaux flottants personnalisés peuvent être déplacés, redimensionnés et leurs onglets combinés, à l'instar des panneaux flottants intégrés à Dreamweaver. Les panneaux flottants personnalisés présentent les différences suivantes par rapport aux panneaux flottants intégrés :

- Les panneaux flottants personnalisés s'affichent en gris par défaut. La définition de l'attribut `BGCOLOR` dans la balise `BODY` n'a aucun effet.
- Tous les panneaux flottants personnalisés s'affichent toujours au premier plan dans la fenêtre de document ou bien flottent derrière celle-ci lorsqu'ils sont inactifs, suivant le paramètre de l'option Toutes autres palettes défini dans les Préférences de panneaux.

Les fichiers de panneau flottant diffèrent également des autres fichiers d'extension. Contrairement aux autres fichiers d'extension, Dreamweaver ne charge pas de fichiers de panneau flottant en mémoire au démarrage, sauf si les panneaux flottants étaient affichés la dernière fois que Dreamweaver a été fermé. Si aucun panneau flottant n'était visible lors de la fermeture de Dreamweaver, les fichiers définissant ces panneaux sont chargés uniquement lorsqu'ils sont référencés à partir de l'une des fonctions suivantes : `dreamweaver.getFloaterVisibility()`, `dreamweaver.setFloaterVisibility()` ou `dreamweaver.toggleFloater()`. Pour plus d'informations sur ces fonctions, voir le *Guide des API de Dreamweaver*.

Lorsque l'un des fichiers du dossier Configuration appelle les fonctions `dw.getFloaterVisibility(`*`floaterName`*`)`, `dw.setFloaterVisibility(`*`floaterName`*`)` ou `dw.toggleFloater(`*`floaterName`*`)`, les événements suivants se produisent :

1. Si l'argument *`floaterName`* ne correspond pas à l'un des noms de panneau flottant réservés, Dreamweaver recherche dans le dossier Configuration/Floaters un fichier nommé *`floaterName`*.htm (pour obtenir une liste complète des noms de panneau flottant réservés, voir la fonction `dreamweaver.getFloaterVisibility()` dans le *Guide des API de Dreamweaver*. Si *`floaterName`*.htm est introuvable, Dreamweaver recherche *`floaterName`*.html. Si aucun fichier de ce nom n'est trouvé, aucun autre événement ne se produit.
2. Si le fichier de panneau flottant est chargé pour la première fois, la fonction `initialPosition()` est appelée, si elle est définie, pour déterminer la position par défaut du panneau flottant à l'écran ; la fonction `initialTabs()` est également appelée, si elle est définie, pour déterminer le groupement d'onglets par défaut du panneau flottant.
3. Les fonctions `selectionChanged()` et `documentEdited()` sont appelées car il est supposé que des modifications ont été apportées pendant que le panneau flottant était masqué.

4. Lorsque le panneau flottant est visible, les événements suivants se produisent :
   - Lorsqu'un élément différent est sélectionné, la fonction `selectionChanged()` est appelée, si elle est définie.
   - Lorsque l'utilisateur modifie le document, la fonction `documentEdited()` est appelée, si elle est définie.
   - Les gestionnaires d'événements associés aux champs dans l'interface de panneau flottant s'exécutent au fur et à mesure que l'utilisateur les rencontre (par exemple, un bouton associé à un gestionnaire d'événements `onClick` exécutant `dw.getDocumentDOM().body.innerHTML=''` a pour effet, lorsque l'utilisateur clique dessus, de supprimer tous les éléments situés entre les balises `BODY` de début et de fin dans le document).

     Les panneaux flottants gèrent deux événements spéciaux relatifs à la balise `body` : `onShow()` et `onHide()`.
5. Lorsque l'utilisateur quitte Dreamweaver, la visibilité, la position et le groupement d'onglets actuellement définis pour le panneau flottant sont enregistrés. Au démarrage suivant, l'application Dreamweaver charge les fichiers de panneau flottant correspondant à tous les panneaux flottants qui étaient affichés au moment de la dernière fermeture de l'application et elle affiche les panneaux flottants au même endroit et avec le même groupement d'onglets.

Exemple de panneau flottant simple

Cet exemple implique la création de l'extension Script Editor, qui génère un panneau flottant pour afficher le code JavaScript inhérent à un marqueur de script sélectionné en mode Création. L'éditeur de script affiche le code JavaScript dans l'élément `textarea` d'un formulaire HTML, défini dans un panneau flottant nommé `scriptlayer`. Si vous apportez des modifications au code sélectionné dans le panneau flottant, l'extension appelle la fonction `updateScript()` pour enregistrer vos modifications. Si vous n'avez pas sélectionné de marqueur de script lorsque vous invoquez l'éditeur de script, l'extension affiche `(no script selected)` dans un panneau flottant nommé `blanklayer`.

Vous créez cette extension en exécutant les étapes suivantes :

- Création des panneaux flottants
- Ecriture du code JavaScript
- Création d'un élément de menu

Création des panneaux flottants

Le début du fichier HTML de cette extension comporte les informations d'en-tête de document standard et une balise `title` qui inscrit les mots Script Editor dans la barre de titre des panneaux flottants.

Pour créer l'en-tête du fichier HTML :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Script Editor</title>
<script language="JavaScript">
```

L'extension définit deux panneaux flottants qui affichent soit le message `(no script selected)` si l'utilisateur n'a pas sélectionné de marqueur de script ou le code JavaScript inhérent à un marqueur de script sélectionné. Le code suivant définit ces deux panneaux flottants, ou calques, nommés `blanklayer` et `scriptlayer` :

Pour créer deux panneaux flottants :

1. Ajoutez le code suivant après l'en-tête dans le fichier HTML :

```
<body>
<div id="blanklayer" style="position:absolute; width:422px; ¬
height:181px; z-index:1; left: 8px; top: 11px; ¬
visibility: hidden">
<center>
<br>
<br>
<br>
<br>
<br>
(no script selected)
</center>
</div>

<div id="scriptlayer" style="position:absolute; width:422px; ¬
height:181px; z-index:1; left: 8px; top: 11px; ¬
visibility: visible">
<form name="theForm">
<textarea name="scriptCode" cols="80" rows="20" wrap="VIRTUAL" ¬
onBlur="updateScript()"></textarea>
</form>
</div>
```

```
</body>
</html>
```

2. Enregistrez le fichier sous le nom Editeur de script.htm dans le dossier Configuration/Floaters.

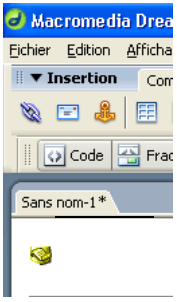
Les deux balises `div` utilisent l'attribut `style` pour spécifier la position (`absolute`), la taille (`width:422px` et `height:181px`) et le paramètre `visibility` par défaut (`visible`) des panneaux flottants. Le panneau `blanklayer` utilise l'attribut `center` ainsi qu'une série de balises de saut de ligne (`br`) pour positionner le texte au centre du panneau. Le panneau `scriptlayer` crée un formulaire contenant une balise `textarea` unique pour afficher le code JavaScript sélectionné. La balise `textarea` spécifie également que, lorsque survient un événement `onBlur` indiquant une modification du code sélectionné, la fonction `updateScript()` est appelée pour écrire à nouveau le texte modifié dans le document.

Ecriture du code JavaScript

Le code JavaScript pour l'éditeur de script consiste en un panneau flottant appelé par Dreamweaver, `selectionchanged()`, et une fonction définie par l'utilisateur, `updateScript()`.

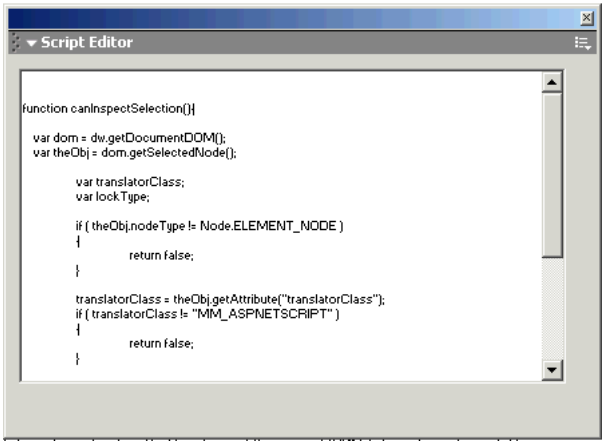
selectionChanged() : avez-vous sélectionné un marqueur de script ?

La fonction `selectionChanged()` détermine si un marqueur de script a été sélectionné en mode Création. Un marqueur de script s'affiche en mode Création si une routine JavaScript est présente dans la section `BODY` d'un document et si Scripts est sélectionné dans la section Eléments invisibles de la boîte de dialogue Préférences. La figure suivante représente une icône de marqueur de script :



Marqueur de

La fonction `selectionChanged()` commence par appeler la fonction `dw.getDocumentDOM()` pour obtenir le DOM (modèle d'objet de document) du document de l'utilisateur. Elle appelle ensuite la fonction `getSelectedNode()` pour voir si le nœud sélectionné pour ce document est un élément, mais également une balise `SCRIPT`. Si ces deux conditions sont réunies, la fonction `selectionChanged()` rend visible le panneau flottant `scripteditor` et le charge avec le code JavaScript inhérent. Elle règle également la propriété `visibility` du panneau flottant `blanklayer` sur la valeur `hidden`. La figure suivante représente le panneau flottant `scriptlayer` avec le code JavaScript sélectionné :



Si le nœud sélectionné n'est pas un élément ou qu'il ne s'agit pas d'une balise `SCRIPT`, la fonction `selectionChanged()` rend visible le panneau flottant `blanklayer` et cache le panneau `scriptlayer`. Le panneau flottant `blanklayer` affiche le texte `(no script selected)`, comme indiqué par la figure suivante :

Pour ajouter la fonction selectionChanged() :

1. Ouvrez le fichier Editeur de script.htm dans le dossier Configuration/Floaters.
2. Entrez le code suivant dans la section en-tête du fichier.

```
function selectionChanged(){
  /* get the selected node */
  var theDOM = dw.getDocumentDOM();
  var theNode = theDOM.getSelectedNode();

  /* check to see if the node is a script marker */
  if (theNode.nodeType == Node.ELEMENT_NODE && ¬
  theNode.tagName == "SCRIPT"){
    document.layers['scriptlayer'].visibility = 'visible';
    document.layers['scriptlayer'].document.theForm.¬
    scriptCode.value = theNode.innerHTML;
    document.layers['blanklayer'].visibility = 'hidden';
  }else{
    document.layers['scriptlayer'].visibility = 'hidden';
    document.layers['blanklayer'].visibility = 'visible';
  }
}
```

3. Enregistrez le fichier.

updateScript() : écriture en retour des modifications

La fonction `updateScript()` écrit à nouveau le script sélectionné lorsqu'un événement `onBlur` survient dans la zone `textarea` du panneau `scriptlayer`. L'élément de formulaire `textarea` contient le code JavaScript et l'événement `onBlur` se produit lorsque `textarea` n'est plus la zone sélectionnée.

Pour ajouter la fonction updateScript() :

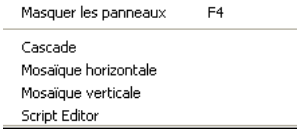
1. Ouvrez le fichier Editeur de script.htm, qui réside dans le dossier Configuration/Floaters.
2. Entrez le code suivant dans la section en-tête du fichier.

```
/* update the document with any changes made by
   the user in the textarea */

  function updateScript(){
  var theDOM = dw.getDocumentDOM();
  var theNode = theDOM.getSelectedNode();
  theNode.innerHTML = document.layers['scriptlayer'].document.¬
  theForm.scriptCode.value;
}

</script>
</head>
```

3. Enregistrez le fichier.

Création d'un élément de menu

Il ne suffit pas d'enregistrer le code d'éditeur de script dans le dossier Configuration/Floaters. Vous devez également appeler la fonction `dw.setFloaterVisibility('scriptEditor',true)` ou la fonction `dw.toggleFloater('scriptEditor')` pour charger et rendre visible le panneau flottant. L'emplacement le plus logique pour invoquer l'éditeur de script est le menu Fenêtre, défini dans le fichier menus.xml. Vous devez créer la balise `menuitem`, qui génère une entrée pour l'extension d'éditeur de script dans le menu Fenêtre, comme indiqué dans la figure suivante :



Si vous sélectionnez un marqueur de script en mode Création pour le document en cours, puis que vous sélectionnez l'élément de menu d'éditeur de script, le panneau flottant d'éditeur de script est invoqué et le code JavaScript inhérent au marqueur de script s'affiche. Si vous sélectionnez l'élément de menu lorsque aucun marqueur de script n'a été sélectionné, le panneau `blanklayer` contenant le texte `(no script selected)` s'affiche.

Pour ajouter l'élément de menu :

1. Ouvrez le fichier menus.xml, qui réside dans le dossier Configuration/Menus.
2. Localisez la balise débutant par `<menuitem name="Tile _Vertically"` et placez le curseur après le symbole `/>` de fermeture de la balise.
3. Insérez le code suivant sur une nouvelle ligne :

```
<menuitem name="Script Editor" enabled="true" ¬
command="dw.toggleFloater('scriptEditor')"¬
checked="dw.getFloaterVisibility('scriptEditor')" />
```

4. Enregistrez le fichier.

API du panneau flottant

Toutes les fonctions personnalisées de l'API de panneau flottant sont facultatives.

Certaines fonctions de cette section ne fonctionnent que sous Windows. Leur description indique si tel est le cas.

displayHelp()

Description

Si cette fonction est définie, un bouton Aide s'affiche sous les boutons OK et Annuler dans la boîte de dialogue. Cette fonction est appelée lorsque l'utilisateur clique sur le bouton Aide.

Arguments

Néant

Valeurs renvoyées

Dreamweaver n'attend rien.

Exemple

```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
  var myHelpFile = dw.getConfigurationPath() +
    '/ExtensionsHelp/superDuperHelp.htm';
  dw.browseDocument(myHelpFile);
}
```

documentEdited()

Description

Cette fonction est appelée lorsque le panneau flottant s'affiche et une fois que la série de modifications en cours est terminée, ce qui signifie que plusieurs modifications peuvent être effectuées avant que cette fonction ne soit appelée. Cette fonction ne doit être définie que si le panneau flottant doit assurer le suivi des modifications apportées au document.

REMARQUE	Ne définissez la fonction `documentEdited()` que si vous en avez besoin, car elle a un impact sur les performances.

Arguments

Néant

Valeurs renvoyées

Dreamweaver n'attend rien.

Exemple

Dans l'exemple suivant, la fonction `documentEdited()` recherche des calques dans le document et met à jour un champ de texte affichant le nombre de calques du document :

```
function documentEdited(){
  /* create a list of all the layers in the document */
  var theDOM = dw.getDocumentDOM();
  var layersInDoc = theDOM.getElementsByTagName("layer");
  var layerCount = layersInDoc.length;

  /* update the numOfLayers field with the new layer count */
  document.theForm.numOfLayers.value = layerCount;
}
```

getDockingSide()

Disponibilité

Dreamweaver MX.

Description

Indique où vous pouvez ancrer les panneaux flottants. Cette fonction renvoie une chaîne contenant une combinaison des mots `"left"`, `"right"`, `"top"` et `"bottom"` (à savoir, à gauche, à droite, en haut et en bas). Vous pouvez ancrer le panneau flottant sur le côté indiqué dans la chaîne. Si aucune indication n'apparaît, vous ne pouvez pas ancrer de panneau flottant.

Cette fonction vous permet d'interdire l'ancrage de certains panneaux sur un côté spécifique de l'espace de travail de Dreamweaver ou sur un autre panneau.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne contenant les mots `"left"`, `"right"`, `"top"` ou `"bottom"`, ou une combinaison de ces mots indiquant où Dreamweaver peut ancrer le panneau flottant.

Exemple

```
getDockingSide()
{
  return dock_side = “left top”;
}
```

initialPosition()

Description

Détermine la position initiale du panneau flottant lorsqu'il est appelé pour la première fois. Si cette fonction n'est pas définie, la position par défaut est le centre de l'écran.

Arguments

`platform`

- L'argument *`platform`* a la valeur `Mac` ou `Win`, selon la plate-forme de l'utilisateur.

Valeurs renvoyées

Dreamweaver attend une chaîne au format `"leftPosInPixels,topPosInPixels"`.

Exemple

Dans l'exemple suivant, la fonction `initialPosition()` spécifie que lorsque le panneau flottant s'affiche pour la première fois, il doit se trouver à 420 pixels du côté gauche de l'écran et à 20 pixels du haut de l'écran sous Windows ou à 390 pixels du côté gauche et à 20 pixels du haut de l'écran sur Macintosh :

```
function initialPosition(platform){
  var initPos = "420,20";
  if (platform == "macintosh"){
    initPos = "390,20";
  }
  return initPos;
}
```

initialTabs()

Description

Détermine les autres panneaux flottants dont les onglets sont combinés à ceux de ce panneau, lorsqu'il s'affiche pour la première fois. Si l'un des panneaux flottants répertoriés s'est affiché précédemment, il n'est pas inclus dans le groupement d'onglets. Pour que les onglets de deux panneaux flottants personnalisés soient combinés, chacun d'eux doit référencer l'autre dans sa fonction `initialTabs()`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne au format `"floaterName1,floaterName2,...floaterNameN"`.

Exemple

Dans l'exemple suivant, la fonction `initialTabs()` spécifie que lorsque le panneau flottant s'affiche pour la première fois, ses onglets doivent être combinés à ceux du panneau flottant scriptEditor :

```
function initialTabs(){
  return "scriptEditor";
}
```

isATarget()

Disponibilité

Dreamweaver MX (sous Windows uniquement)

Description

Détermine si d'autres panneaux peuvent s'ancrer à ce panneau flottant. Si la fonction `isATarget()` n'est pas spécifiée, Dreamweaver empêche les autres panneaux de s'ancrer à celui-ci. Dreamweaver appelle cette fonction lorsque l'utilisateur essaie de combiner ce panneau avec d'autres.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si d'autres panneaux flottants peuvent s'ancrer à celui-ci ; `false` dans le cas contraire.

Exemple

```
IsATarget()
{
  return true;
}
```

isAvailableInCodeView()

Description

Détermine si le panneau flottant doit être activé lorsque le mode Code est sélectionné. Si cette fonction n'est pas définie, le panneau flottant est désactivé en mode Code.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le panneau flottant doit être activé en mode Code ; `false` dans le cas contraire.

isResizable()

Disponibilité

Dreamweaver 4.

Description

Détermine si l'utilisateur peut redimensionner le panneau flottant. Si cette fonction n'est pas définie, ou si elle renvoie la valeur `true`, l'utilisateur peut redimensionner le panneau flottant. Si la fonction renvoie la valeur `false`, l'utilisateur ne peut pas redimensionner le panneau flottant.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si l'utilisateur peut redimensionner le panneau flottant ; `false` dans le cas contraire.

Exemple

Dans l'exemple suivant, l'utilisateur ne peut pas redimensionner le panneau flottant :

```
function isResizable()
{
  return false;
}
```

selectionChanged()

Description

Appelée lorsque le panneau flottant s'affiche et à chaque changement de sélection (lorsqu'un nouveau document devient actif ou lorsque le pointeur d'insertion passe à un nouvel emplacement dans le document actif). Cette fonction ne doit être définie que si le panneau flottant doit assurer le suivi de la sélection.

REMARQUE	Ne définissez la fonction `selectionChanged()` que si vous en avez absolument besoin, car elle a un impact sur les performances.

Arguments

Aucun.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

Dans l'exemple suivant, la fonction `selectionChanged()` affiche un calque différent dans le panneau flottant selon que la sélection est un marqueur de script. Si la sélection est un marqueur de script, Dreamweaver rend le calque de script visible. Sinon, Dreamweaver rend le calque vierge visible :

```
function selectionChanged(){
   /* get the selected node */
   var theDOM = dw.getDocumentDOM();
   var theNode = theDOM.getSelectedNode();

   /* check to see if the node is a script marker */
   if (theNode.nodeType == Node.ELEMENT_NODE && ¬
   theNode.tagName == "SCRIPT"){
     document.layers['blanklayer'].visibility = 'hidden';
     document.layers['scriptlayer'].visibility = 'visible';}
  else{
     document.layers['scriptlayer'].visibility = 'hidden';
     document.layers['blanklayer'].visibility = 'visible';
   }
}
```

A propos des performances

La déclaration des fonctions `selectionChanged()` ou `documentEdited()` dans vos panneaux flottants personnalisés risque d'avoir des répercussions négatives sur les performances de Dreamweaver. Cela est dû au fait que les fonctions `documentEdited()` et `selectionChanged()` sont appelées après chaque frappe d'une touche et chaque clic de la souris si Dreamweaver est resté inactif plus d'un dixième de seconde. Il est important de tester votre panneau flottant avec divers scénarios, en utilisant des documents volumineux (100 Ko ou plus de code HTML) autant que possible.

Pour éviter les pertes de performances, la fonction `setTimeout()` a été implémentée en tant que méthode globale dans Dreamweaver 3. Tout comme dans les navigateurs, la fonction `setTimeout()` accepte deux arguments : le code JavaScript devant être appelé et le délai d'attente avant cet appel en millisecondes.

La méthode `setTimeout()` vous permet d'introduire dans votre traitement des pauses au cours desquelles l'utilisateur peut poursuivre son interaction avec l'application. Vous devez incorporer ces pauses de façon explicite étant donné que l'écran se fige pendant le traitement des scripts et, par conséquent, vous empêche d'effectuer d'autres modifications (et de mettre à jour l'interface ou le panneau flottant).

L'exemple suivant correspond à un panneau flottant qui affiche des informations sur chacun des calques du document. Il utilise la méthode `setTimeout()` pour faire une pause d'une demi-seconde après le traitement de chaque calque.

```
/* create a flag that specifies whether an edit is being processed, and set
   it to false. */
document.running = false;

/* this function called when document is edited */
function documentEdited(){
  /* create a list of all the layers to be processed */
  var dom = dw.getDocumentDOM();
  document.layers = dom.getElementsByTagName("layer");
  document.numLayers = document.layers.length;
  document.numProcessed = 0;

  /* set a timer to call processLayer(); if we didn't get
   * to finish processing the previous edit, then the timer
   * is already set. */
  if (document.running = false){
    setTimeout("processLayer()", 500);
  }

  /* set the processing flag to true */
  document.running = true;
}

/* process one layer */
function processLayer(){
  /* display information for the next unprocessed layer.
      displayLayer() is a function you would write to
      perform the "magic".  */
   displayLayer(document.layers[document.numProcessed]);

  /* if there's more work to do, set a timeout to process
   * the next layer.  If we're finished, set the document.running
   * flag to false. */
  document.numProcessed = document.numProcessed + 1;
  if (document.numProcessed < document.numLayers){
    setTimeout("processLayer()", 500);
  }else{
    document.running = false;
  }
}
```

CHAPITRE 14

Comportements

14

Les comportements permettent aux utilisateurs de créer des pages HTML interactives. Ils donnent aux concepteurs de sites Web la possibilité d'affecter facilement des actions à des éléments de page en remplissant un formulaire HTML.

Le terme *comportement* fait référence à la combinaison d'un événement (tel que `onClick`, `onLoad` ou `onSubmit`) et d'une action (par exemple, Vérifier le plug-in, Atteindre l'URL, Permuter une image). Le navigateur détermine quels éléments HTML acceptent quels événements. Les fichiers répertoriant les événements que chaque navigateur prend en charge sont stockés dans le dossier Configuration/Behaviors/Events du dossier de l'application Macromedia 8.

Les actions constituent la partie d'un comportement que vous pouvez contrôler ; ainsi, lorsque vous rédigez un comportement, vous rédigez en réalité un fichier d'action. Les actions sont des fichiers HTML. La section `BODY` d'un fichier d'action contient généralement un formulaire HTML qui accepte les paramètres de l'action (par exemple, les paramètres indiquant les calques à afficher ou masquer). La section `HEAD` d'un fichier d'action contient des fonctions JavaScript qui traitent les entrées de formulaire du contenu `BODY` et contrôlent les fonctions, arguments et gestionnaires d'événements insérés dans le document de l'utilisateur.

Vous devez écrire des actions de comportement lorsque vous souhaitez partager des fonctions avec des utilisateurs ou lorsque vous souhaitez insérer la même fonction JavaScript à plusieurs reprises tout en modifiant les paramètres à chaque fois.

REMARQUE

Vous ne pouvez pas utiliser les comportements pour insérer des fonctions VBScript directement ; vous pouvez toutefois ajouter indirectement une fonction VBScript en modifiant le modèle d'objet de document (DOM) dans la fonction `applyBehavior()`.

Le tableau ci-dessous recense les fichiers utilisés pour créer une action de comportement :

Chemin	Fichier	Description
Configuration/Behaviors/Actions/	*action_comportement*.htm	La section `BODY` du fichier inclut un formulaire HTML contenant les paramètres de l'action. La section `HEAD` du fichier contient les fonctions JavaScript.

REMARQUE

Pour plus d'informations sur les comportements des serveurs qui offrent une fonctionnalité d'application Web, voir Chapitre 15, "Comportements de serveur," on page 335.

Fonctionnement des comportements

Lorsqu'un utilisateur sélectionne un élément HTML dans un document Dreamweaver et clique sur le bouton Plus (+) dans le panneau Comportements, les événements suivants se produisent :

1. Dreamweaver appelle la fonction `canAcceptBehavior()` dans chaque fichier d'action pour vérifier si l'action est adaptée au document ou à l'élément sélectionné.

   Si cette fonction renvoie la valeur `false`, Dreamweaver estompe l'action dans le menu déroulant des actions (par exemple, l'action Contrôler Shockwave ou Flash est estompée lorsque le document de l'utilisateur ne contient pas de fichiers SWF.) Si la valeur renvoyée est une liste d'événements, Dreamweaver les compare un à un aux événements valides de l'élément HTML actuellement sélectionné et du navigateur cible jusqu'à ce qu'une correspondance soit trouvée. Dreamweaver ajoute au menu déroulant Evénements l'événement correspondant, extrait de la fonction `canAcceptBehavior()` en tête de liste. Si aucune correspondance n'est établie, l'événement associé par défaut à l'élément HTML (marqué d'un astérisque [*]) dans le fichier d'événements) est placé en tête de liste. Les autres événements du menu sont assemblés à partir du fichier d'événements.
2. L'utilisateur sélectionne une action dans le menu déroulant des actions.
3. Dreamweaver appelle la fonction `windowDimensions()` pour déterminer la taille de la boîte de dialogue des paramètres. Si la fonction `windowDimensions()` n'est pas définie, la taille est déterminée automatiquement.

   Une boîte de dialogue s'affiche toujours, les boutons OK et Annuler figurant sur le côté droit, quel que soit le contenu de l'élément `BODY`.

4. Dreamweaver affiche une boîte de dialogue contenant les éléments `BODY` du fichier d'action. Si la balise `BODY` du fichier d'action contient un gestionnaire `onLoad`, Dreamweaver l'exécute.
5. L'utilisateur entre les paramètres de l'action. Dreamweaver exécute les gestionnaires d'événements associés aux champs du formulaire au fur et à mesure que l'utilisateur les rencontre.
6. L'utilisateur clique sur OK.
7. Dreamweaver appelle les fonctions `behaviorFunction()` et `applyBehavior()` dans le fichier d'action sélectionné. Ces fonctions renvoient des chaînes qui sont insérées dans le document de l'utilisateur.
8. Si l'utilisateur double-clique sur l'action dans la colonne Actions, Dreamweaver rouvre la boîte de dialogue des paramètres et exécute le gestionnaire `onLoad`. Dreamweaver appelle alors la fonction `inspectBehavior()` dans le fichier d'action sélectionné, ce qui a pour effet de renseigner les champs à l'aide des données que l'utilisateur a entrées précédemment.

Insertion de plusieurs fonctions dans le fichier de l'utilisateur

Les actions peuvent insérer plusieurs fonctions (la fonction de comportement principale et un nombre illimité de fonctions d'assistance) dans la section `HEAD`. Plusieurs comportements peuvent même partager des fonctions d'assistance, à condition que les fonctions soient définies exactement de la même façon dans chaque fichier d'action. Une façon de garantir que les fonctions partagées sont identiques consiste à stocker chaque fonction d'assistance dans un fichier JavaScript externe et à insérer ce dernier dans les fichiers d'action appropriés à l'aide de `<SCRIPT SRC="externalFile.js">`.

Lorsque l'utilisateur supprime un comportement, Dreamweaver tente de supprimer toutes les fonctions d'assistance inutilisées associées à ce comportement. Si d'autres comportements utilisent une fonction d'assistance, celle-ci n'est pas supprimée. En raison de l'extrême prudence de l'algorithme de suppression des fonctions d'assistance, Dreamweaver peut occasionnellement laisser une fonction inutilisée dans le document de l'utilisateur.

Procédure à suivre lorsqu'une action exige une valeur renvoyée

Il peut arriver qu'un gestionnaire d'événements exige une valeur renvoyée (par exemple, `onMouseOver="window.status='This is a link'; return true"`). Mais si Dreamweaver insère l'action `"return behaviorName(args)"` dans le gestionnaire d'événements, les autres comportements de la liste sont ignorés.

Pour contourner cette limitation, affectez la valeur de retour souhaitée à la variable `document.MM_returnValue`, dans la chaîne renvoyée par la fonction `behaviorFunction()`. Ce paramétrage entraîne l'insertion par Dreamweaver de `return document.MM_returnValue` à la fin de la liste des actions du gestionnaire d'événements. Pour accéder à un exemple d'utilisation de la variable `MM_returnValue` , voir le fichier Validate Form.js, qui réside dans le dossier Configuration/Behaviors/Actions au sein du dossier de l'application Dreamweaver.

Exemple de comportement simple

Pour mieux comprendre le fonctionnement des comportements et la façon de le créer, il peut être utile d'examiner un exemple. Le dossier Configuration/Behaviors/Actions, qui réside au sein du dossier de l'application Dreamweaver, contient des exemples. La plupart d'entre eux sont cependant très complexes. Cet exemple est plus simple, vous permettant ainsi de vous familiariser avec la création de comportements. Le fichier d'action le plus simple pour commencer est Call JavaScript.htm (avec son homologue, Call JavaScript.js, qui contient toutes les fonctions JavaScript).

Pour créer un comportement, exécutez la procédure ci-dessous :

- Création de l'extension de comportement
- Création des fichiers HTML à consulter
- Test du comportement

Création de l'extension de comportement

Le code suivant présente un exemple relativement simple. Il vérifie le type du navigateur et atteint une page spécifique s'il s'agit de Netscape Navigator et une page différente s'il s'agit de Microsoft Internet Explorer. Ce code peut facilement être étendu pour vérifier d'autres types de navigateurs (tels qu'Opera et WebTV) et modifié pour exécuter d'autres actions que l'action d'atteindre des URL.

Pour créer l'extension de comportement :

1. Créez un document vierge.
2. Insérez le code suivant dans le fichier :

```
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine 5.0 ¬
  //dialog">
<html>
<head>
<title>behavior "Check Browser Brand"</title>
<meta http-equiv="Content-Type" content="text/html">
<script language="JavaScript">

// The function that will be inserted into the
// HEAD of the user's document
function checkBrowserBrand(netscapeURL,explorerURL) {
  if (navigator.appName == "Netscape") {
    if (netscapeURL) location.href = netscapeURL;
  }else if (navigator.appName == "Microsoft Internet Explorer") {
    if (explorerURL) location.href = explorerURL;
  }
}

//****************** API **********************

function canAcceptBehavior(){
  return true;
}

// Return the name of the function to be inserted into
// the HEAD of the user's document
function behaviorFunction(){
    return "checkBrowserBrand";
}

// Create the function call that will be inserted
// with the event handler
function applyBehavior() {
  var nsURL = escape(document.theForm.nsURL.value);
  var ieURL = escape(document.theForm.ieURL.value);
  if (nsURL && ieURL) {
    return "checkBrowserBrand(\'" + nsURL + "\',\'" + ieURL + ¬
    "\')";
  }else{
    return "Please enter URLs in both fields."
  }
}

// Extract the arguments from the function call
// in the event handler and repopulate the
// parameters form
```

```
function inspectBehavior(fnCall){
  var argArray = getTokens(fnCall, "()',");
  var nsURL = unescape(argArray[1]);
  var ieURL = unescape(argArray[2]);
  document.theForm.nsURL.value = nsURL;
  document.theForm.ieURL.value = ieURL;
}

//***************** LOCAL FUNCTIONS  ******************

// Put the pointer in the first text field
// and select the contents, if any
function initializeUI(){
  document.theForm.nsURL.focus();
  document.theForm.nsURL.select();
}

// Let the user browse to the Navigator and
// IE URLs
function browseForURLs(whichButton){
  var theURL = dreamweaver.browseForFileURL();
  if (whichButton == "nsURL"){
    document.theForm.nsURL.value = theURL;
  }else{
    document.theForm.ieURL.value = theURL;
  }
}

//*************** END OF JAVASCRIPT *****************
</script>
</head>
<body>
<form method="post" action="" name="theForm">
<table border="0" cellpadding="8">
<tr>
<td nowrap="nowrap">  Go to this URL if the browser is ¬
Netscape Navigator:<br>
<input type="text" name="nsURL" size="50" value="">  
<input type="button" name="nsBrowse" value="Browse..." ¬
onClick="browseForURLs('nsURL')"></td>
</tr>
<tr>
<td nowrap="nowrap">  Go to this URL is the browser is ¬
Microsoft Internet Explorer:<br>
<input type="text" name="ieURL" size="50" value="">  
<input type="button" name="ieBrowse" value="Browse..." ¬
onClick="browseForURLs('ieURL')"></td>
</tr>
</table>
</form>
```

```
</body>
</html>
```

3. Enregistrez le fichier sous le nom Configuration/Behaviors/Actions/BrowserDependentURL.htm.

Création des fichiers HTML à consulter

Vous créez ensuite les fichiers HTML à consulter, selon que votre navigateur correspond à Internet Explorer ou Netscape Navigator.

Pour créer les fichiers HTML à consulter :

1. Créez un fichier au contenu suivant :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Internet Explorer Only</title>
</head>

<body>
This is the page to go to if you are using Internet Explorer.
</body>
</html>
```

2. Enregistrez le fichier sous le nom iecontent.htm sur votre ordinateur.
3. Créez un autre fichier au contenu suivant :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Netscape Navigator content</title>
</head>

<body>
This is the page to go to if you are using Netscape Navigator.
</body>
</html>
```

4. Enregistrez le fichier sous le nom netscapecontent.htm dans le même répertoire que le fichier iecontent.htm.
5. Redémarrez Dreamweaver.
6. Créez un fichier HTML au contenu suivant :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Which browser</title>
</head>

<body>
</body>
</html>
```

7. Enregistrez le fichier sous le nom whichbrowser.htm dans le même répertoire que le fichier iecontent.htm.
8. Cliquez sur le bouton Plus (+) du panneau Comportements et sélectionnez le comportement Check Browser Band (Vérifier type de navigateur).
9. Cliquez sur le bouton Parcourir en regard de la zone de texte Go to the URL if the browser is Netscape Navigator (Accéder à l'URL associée au navigateur Netscape Navigator) et sélectionnez le fichier netscapecontent.htm.
10. Cliquez sur le bouton Parcourir en regard de la zone de texte Go to the URL if the browser is Internet Explorer (Accéder à l'URL associée au navigateur Internet Explorer) et sélectionnez le fichier iecontent.htm.
11. Cliquez sur OK.

    Dreamweaver ajoute le code JavaScript spécifié au fichier whichbrowser.htm. Ce dernier se présente alors comme suit :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Which browser</title>
<script language="JavaScript" type="text/JavaScript">
<!--
function checkBrowserBrand(netscapeURL,explorerURL) {
  if (navigator.appName == "Netscape") {
    if (netscapeURL) location.href = netscapeURL;
  }else if (navigator.appName == "Microsoft Internet Explorer") {
    if (explorerURL) location.href = explorerURL;
  }
}
//-->
</script>
</head>

<body
  onLoad="checkBrowserBrand('netscaptecontent.htm','iecontent.htm')">
```

```
</body>
</html>
```

Test du comportement

Vous pouvez enfin tester le comportement.

Pour tester le comportement :

1. Affichez le fichier whichbrowser.htm dans votre navigateur.

   Le fichier qui s'affiche (iecontent.htm ou netscapecontent.htm) varie selon le navigateur utilisé.

API de comportements

Deux fonctions d'API de comportements sont obligatoires : `applyBehavior()` et `behaviorFunction()` ; les autres sont facultatives.

applyBehavior()

Description

Cette fonction insère, dans le document de l'utilisateur, un gestionnaire d'événements qui appelle la fonction insérée par la fonction `behaviorFunction()`. La fonction `applyBehavior()` peut également exécuter d'autres modifications dans le document de l'utilisateur, mais elle ne doit pas supprimer l'objet auquel le comportement est appliqué ou qui reçoit l'action.

Arguments

`uniqueName`

- Cet argument est un identifiant unique parmi toutes les instances de tous les comportements du document de l'utilisateur. Son format est *`functionNameInteger`*, où *`functionName`* correspond au nom de la fonction insérée par `behaviorFunction()`. Cet argument peut être utile si vous insérez une balise dans le document de l'utilisateur et souhaitez affecter une valeur unique à son attribut `NAME`.

Valeurs renvoyées

Dreamweaver renvoie une chaîne contenant l'appel de fonction qui doit être inséré dans le document de l'utilisateur, généralement après acceptation des paramètres entrés par l'utilisateur. Si la fonction `applyBehavior()` détermine que l'utilisateur a effectué une entrée non valide, la fonction peut renvoyer une chaîne d'erreur au lieu de l'appel de fonction. Dreamweaver ne signale aucune erreur si la chaîne est vide (`return "";`), mais si la chaîne n'est pas vide et qu'elle n'est pas non plus un appel de fonction, Dreamweaver affiche une boîte de dialogue contenant le texte : Invalid Input supplied for this behavior: [the string returned from `applyBehavior()`] (l'entrée spécifiée n'est pas valide pour ce comportement : [chaîne renvoyée par applyBehavior()]). Si la valeur renvoyée est `null` (`return;`), Dreamweaver indique qu'une erreur s'est produite mais sans fournir plus de détails.

REMARQUE Les guillemets (`""`) contenus dans la chaîne renvoyée doivent être précédés d'une barre oblique inversée (\) afin d'éviter que l'interpréteur JavaScript ne signale des erreurs.

Exemple

Dans l'exemple suivant, la fonction `applyBehavior()` renvoie un appel à la fonction `MM_openBrWindow()` et transmet les paramètres fournis par l'utilisateur (hauteur et largeur de la fenêtre ; affichage des barres de défilement, une barre d'outils, une barre d'emplacement et d'autres fonctions ; URL devant s'ouvrir dans la fenêtre) :

```
function applyBehavior() {
  var i,theURL,theName,arrayIndex = 0;
  var argArray = new Array(); //use array to produce correct ¬
  number of commas w/o spaces
  var checkBoxNames = new Array("toolbar","location",¬
  "status","menubar","scrollbars","resizable");

  for (i=0; i<checkBoxNames.length; i++) {
    theCheckBox = eval("document.theForm." + checkBoxNames[i]);
    if (theCheckBox.checked) argArray[arrayIndex++] = ¬
    (checkBoxNames[i] + "=yes");
  }
  if (document.theForm.width.value)
    argArray[arrayIndex++] = ("width=" + ¬
    document.theForm.width.value);
  if (document.theForm.height.value)
    argArray[arrayIndex++] = ("height=" + ¬
    document.theForm.height.value);
  theURL = escape(document.theForm.URL.value);
  theName = document.theForm.winName.value;
  return "MM_openBrWindow('"+theURL+"',¬
```

```
  '"+theName+"','"+argArray.join()+"')";
}
```

behaviorFunction()

Description

Cette fonction insère une ou plusieurs fonctions (placées entre les balises suivantes, si elles n'existent pas encore) dans la section `HEAD` du document utilisateur :

```
<SCRIPT LANGUAGE="JavaScript"></SCRIPT>
```

Arguments

Néant

Valeurs renvoyées

Dreamweaver renvoie soit une chaîne contenant les fonctions JavaScript, soit une chaîne contenant les noms des fonctions à insérer dans le document de l'utilisateur. Cette valeur doit être chaque fois identique (elle ne peut pas dépendre des données entrées par l'utilisateur). Les fonctions sont insérées une seule fois, quel que soit le nombre de fois que l'action est appliquée aux éléments du document.

REMARQUE	Les guillemets (" ") contenus dans la chaîne renvoyée doivent être précédés d'une barre oblique inversée (\) afin d'éviter que l'interpréteur JavaScript ne signale des erreurs.

Exemple

L'instance suivante de la fonction `behaviorFunction()` renvoie la fonction `MM_popupMsg()` :

```
function behaviorFunction(){
  return ""+
  "function MM_popupMsg(theMsg) { //v1.0\n"+
  "  alert(theMsg);\n"+
  "}";
}
```

Le code suivant est équivalent à la déclaration de fonction `behaviorFunction()` précédente et correspond à la méthode utilisée pour déclarer la fonction `behaviorFunction()` dans tous les comportements intégrés à Dreamweaver :

```
function MM_popupMsg(theMsg){ //v1.0
  alert(theMsg);
}
```

```
function behaviorFunction(){
  return "MM_popupMsg";
}
```

canAcceptBehavior()

Description

Cette fonction indique si l'action est autorisée pour l'élément HTML sélectionné et définit l'événement par défaut qui doit la déclencher. Elle peut également rechercher la présence de certains objets (tels que des fichiers SWF) dans le document de l'utilisateur et interdire l'action si ces objets sont absents.

Arguments

HTMLElement

- L'argument est l'élément HTML sélectionné.

Valeurs renvoyées

Dreamweaver renvoie l'une des valeurs suivantes :

- La valeur `true` si l'action est autorisée mais n'est associée à aucun événement préféré.
- Une liste des événements préférés (par ordre décroissant de préférence) de cette action. La définition d'événements préférés remplace l'événement par défaut (signalé par un astérisque [*] dans le fichier d'événements) de l'objet sélectionné. Voir l'étape 1 dans la section *Fonctionnement des comportements*, page 318.
- La valeur `false` si l'action n'est pas autorisée.

Si la fonction `canAcceptBehavior()` renvoie la valeur `false`, l'action est estompée dans le menu contextuel Actions du panneau Comportements.

Exemple

Dans l'exemple suivant, la fonction `canAcceptBehavior()` renvoie une liste des événements préférés pour le comportement si le document contient des images nommées :

```
function canAcceptBehavior(){
  var theDOM = dreamweaver.getDocumentDOM();
  // Get an array of all images in the document
  var allImages = theDOM.getElementsByTagName('IMG');
  if (allImages.length > 0){
    return "onMouseOver, onClick, onMouseDown";
  }else{
    return false;
  }
}
```

displayHelp()

Description

Si cette fonction est définie, un bouton Aide apparaît sous les boutons OK et Annuler dans la boîte de dialogue des paramètres. Cette fonction est appelée lorsque l'utilisateur clique sur le bouton Aide.

Arguments

Néant

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
  var myHelpFile = dw.getConfigurationPath() +
    '/ExtensionsHelp/superDuperHelp.htm';
  dw.browseDocument(myHelpFile);
}
```

deleteBehavior()

Description

Cette fonction annule toute modification effectuée par la fonction `applyBehavior()`.

REMARQUE Dreamweaver supprime automatiquement la déclaration de fonction et le gestionnaire d'événements associés à un comportement lorsque l'utilisateur supprime ce comportement dans le panneau Comportements. Vous ne devez définir la fonction `deleteBehavior()` que si la fonction `applyBehavior()` a effectué des modifications supplémentaires dans le document de l'utilisateur (telle une insertion de balise).

Arguments

`applyBehaviorString`

- Cet argument correspond à la chaîne renvoyée par la fonction `applyBehavior()`.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

identifyBehaviorArguments()

Description

Cette fonction associe les arguments d'un appel de fonction de comportement à des liens de navigation, fichiers dépendants, URL, références de style Netscape 4.0 ou noms d'objet pour que les URL utilisées dans les comportements puissent être mises à jour si l'utilisateur enregistre le document dans un autre emplacement et pour que les fichiers référencés puissent s'afficher dans la carte du site et être considérés comme des fichiers dépendants afin de pouvoir être téléchargés vers et depuis un serveur.

Arguments

`theFunctionCall`

- Cet argument correspond à la chaîne renvoyée par la fonction `applyBehavior()`.

Valeurs renvoyées

Dreamweaver renvoie une chaîne contenant une liste des types d'arguments de l'appel de fonction séparés par des virgules. La longueur de la liste doit être égale au nombre d'arguments transmis lors de l'appel de fonction. Les types d'argument doivent correspondre à l'un des types suivants :

- Le type d'argument `nav` indique que l'argument est une URL de navigation et qu'il doit donc être affiché dans la carte du site.
- Le type d'argument `dep` indique que l'argument est une URL de fichier dépendant et qu'il doit donc être inclus avec tous les autres fichiers dépendants lorsqu'un document contenant ce comportement est téléchargé depuis ou vers un serveur.
- Le type d'argument `URL` indique que l'argument est à la fois une URL de navigation et une URL dépendante (ou une URL de type inconnu), et qu'il doit donc être affiché dans la carte du site et considéré comme un fichier dépendant lors du téléchargement depuis ou vers un serveur.
- Le type d'argument `NS4.0ref` indique que l'argument est une référence à un objet de style Netscape Navigator 4.0.
- Le type d'argument `IE4.0ref` indique que l'argument est une référence à un objet de style DOM 4.0 d'Internet Explorer.
- Le type d'argument `objName` indique que l'argument est un nom d'objet simple, tel qu'il est spécifié dans l'attribut `NAME` de l'objet. Ce type a été ajouté à Dreamweaver 3.
- Le type d'argument `other` indique que l'argument n'appartient à aucun des types ci-dessus.

Exemple

Cet exemple simple de fonction `identifyBehaviorArguments()` fonctionne avec l'action de comportement Ouvrir une fenêtre du navigateur qui renvoie une fonction comportant toujours trois arguments (l'URL à ouvrir, le nom de la nouvelle fenêtre et la liste des propriétés de la fenêtre) :

```
function identifyBehaviorArguments(fnCallStr) {
  return "URL,other,other";
}
```

Une version plus complexe de la fonction `identifyBehaviorArguments()` est nécessaire pour les fonctions de comportement qui gèrent un nombre variable d'arguments (telles qu'Afficher-Masquer les calques). Pour cet exemple de version de la fonction `identifyBehaviorArguments()`, il existe un nombre minimum d'arguments et les arguments supplémentaires se présentent toujours en groupes dont la taille est un multiple de ce nombre minimum. En d'autres termes, une fonction dont le nombre d'arguments minimum est 4 peut avoir 4, 8 ou 12 arguments, mais elle n'en aura jamais 10 :

```
function identifyBehaviorArguments(fnCallStr) {
  var listOfArgTypes;
  var itemArray = dreamweaver.getTokens(fnCallStr, '(),');

  // The array of items returned by getTokens() includes the
  // function name, so the number of *arguments* in the array
  // is the length of the array minus one. Divide by 4 to get the
  // number of groups of arguments.
  var numArgGroups = ((itemArray.length - 1)/4);
  // For each group of arguments
  for (i=0; i < numArgGroups; i++){

  // Add a comma and "NS4.0ref,IE4.0ref,other,dep" (because this
  // hypothetical behavior function has a minimum of four
  // arguments the Netscape object reference, the IE object
  // reference, a dependent URL, and perhaps a property value
  // such as "show" or "hide") to the existing list of argument
  // types, or if no list yet exists, add only
  // "NS4.0ref,IE4.0ref,other,dep"
  var listOfArgTypes += ((listOfArgTypes)?",":"") + ¬
  "NS4.0ref,IE4.0ref,other,dep";
  }
}
```

inspectBehavior()

Description

Cette fonction recherche dans l'appel de fonction un comportement déjà appliqué au document de l'utilisateur et définit les valeurs des options de la boîte de dialogue des paramètres en conséquence. Si la fonction `inspectBehavior()` n'est pas définie, les valeurs par défaut des options s'affichent.

REMARQUE La fonction `inspectBehavior()` doit uniquement utiliser les informations qui lui sont transmises via l'argument *`applyBehaviorString`*. N'essayez pas d'obtenir d'autres informations sur le document de l'utilisateur (par exemple, en utilisant la fonction `dreamweaver.getDocumentDOM()`) dans cette fonction.

Arguments

`applyBehaviorString`

- Cet argument correspond à la chaîne renvoyée par la fonction `applyBehavior()`.

REMARQUE Si l'élément HTML contient du code similaire à '`onClick="someBehavior(); return document.MM_returnValue;"`' et que vous ajoutez un nouveau comportement extrait du menu des comportements, Dreamweaver appelle `inspectBehavior()` dès que l'interface utilisateur du nouveau comportement apparaît et transmet le paramètre sous forme de chaîne vide. De ce fait, vérifiez le paramètre `applyBehaviorString`, illustré dans l'exemple ci-dessous :

```
function inspectBehavior(enteredStr){
  if(enteredStr){
  //do your work here
  }
}
```

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

L'instance suivante de la fonction `inspectBehavior()`, extraite du fichier Display Status Message.htm, renseigne le champ Message de la boîte de dialogue des paramètres avec le message que l'utilisateur a sélectionné lors de la première application du comportement :

```
function inspectBehavior(msgStr){
  var startStr = msgStr.indexOf("'") + 1;
  var endStr = msgStr.lastIndexOf("'");
  if (startStr > 0 && endStr > startStr) {
    document.theForm.message.value = ¬
    unescQuotes(msgStr.substring(startStr,endStr));
```

```
  }
}
```

REMARQUE

Pour plus d'informations sur la fonction `unescQuotes()`, voir le fichier dwscripts.js du dossier Configuration/Shared/Common/Scripts/CMN.

windowDimensions()

Description

Cette fonction définit les dimensions de la boîte de dialogue des paramètres. Si cette fonction n'est pas définie, les dimensions de la fenêtre sont calculées automatiquement.

REMARQUE

Ne définissez cette fonction que si vous souhaitez utiliser une boîte de dialogue de paramètres ayant des dimensions supérieures à 640 x 480 pixels.

Arguments

`platform`

- La valeur de l'argument est soit `"macintosh"`, soit `"windows"`, selon la plate-forme utilisée par l'utilisateur.

Valeurs renvoyées

Dreamweaver renvoie une chaîne au format `"widthInPixels,heightInPixels"`.

Les dimensions renvoyées sont inférieures à la taille totale de la boîte de dialogue parce qu'elles n'incluent pas la zone des boutons OK et Annuler. Si les dimensions renvoyées ne permettent pas de faire apparaître toutes les options, des barres de défilement s'affichent.

Exemple

Dans l'exemple suivant, la fonction `windowDimensions()` définit les dimensions de la boîte de dialogue des paramètres à 648 x 520 pixels :

```
function windowDimensions(){
  return "648,520";
}
```

CHAPITRE 15

Comportements de serveur

15

Macromedia Dreamweaver 8 permet aux utilisateurs d'ajouter des comportements de serveur à leurs documents afin d'exécuter des tâches côté serveur, par exemple pour filtrer des enregistrements en fonction de critères préalablement définis, parcourir des enregistrements, lier des listes de résultats à des pages d'informations détaillées et insérer des enregistrements dans un jeu de résultats. Si l'utilisateur insère sans cesse le même code d'exécution dans les documents, il a tout intérêt à créer une nouvelle extension afin d'automatiser la mise à jour de documents à l'aide de blocs de code couramment utilisés. Voir Ajout de comportements de serveur personnalisés dans le guide *Bien démarrer avec Dreamweaver* pour plus de détails sur l'utilisation de l'interface de l'outil Créateur de comportements de serveur pour créer des comportements de serveur personnalisés. Revenez ensuite au présent chapitre pour des détails sur l'utilisation de fichiers prenant en charge les comportements de serveur et les fonctions interagissant avec les comportements de serveur existants. Pour plus d'informations sur chaque fonction, voir Fonctions relatives aux comportements de serveur et Fonctions du gestionnaire de données d'extension dans le *Guide des API de Dreamweaver*. Dreamweaver prend actuellement en charge les extensions de comportement de serveur qui ajoutent du code d'exécution pour les modèles de serveurs suivants : ASP.Net/C#, ASP.Net/VisualBasic, ASP/JavaScript, ASP/VBScript, ColdFusion, JSP et PHP/MySQL.

Le présent chapitre utilise les termes suivants :

- **Extension de comportement de serveur** L'extension de comportement de serveur est l'interface entre le code côté serveur et Dreamweaver. Une extension de comportement de serveur est composée d'un code JavaScript, d'un code HTML et d'un code EDML (Extension Data Markup Language) qui est du code XML créé spécifiquement pour les données d'extension. Des fichiers d'exemple, organisés en fonction du type de serveur utilisé, sont localisés dans le dossier Configuration/ServerBehaviors, à l'intérieur de votre dossier d'installation. Lorsque vous rédigez une extension, vous devez utiliser la fonction `dwscripts.applySB()` pour demander à Dreamweaver de lire les fichiers EDML, d'extraire les composants de votre extension et d'insérer les blocs de code requis dans le document de l'utilisateur.

- **Instance de comportement de serveur** L'ajout par Dreamweaver de blocs de code au document utilisateur constitue une instance de comportement de serveur. La plupart des comportements de serveur pouvant être appliqués plusieurs fois, les instances peuvent être multiples. Chaque instance de comportement de serveur est répertoriée dans le panneau Comportements de serveur de l'interface Dreamweaver.
- **Code d'exécution** Ensemble des blocs de code ajoutés à un document lorsqu'un comportement de serveur est appliqué. Ces blocs de code sont d'ordinaire un code côté serveur, par exemple, un script ASP compris entre des balises `<% ... %>`.
- **Participants** L'extension de comportement de serveur insère des blocs de code dans le document utilisateur. Un bloc de code est un bloc de script unique et continu, tel qu'une balise côté serveur, une balise HTML, voire un attribut qui ajoute une fonctionnalité côté serveur à une page Web. Un fichier EDML désigne chaque bloc de code par le terme de participant. Tous les participants relatifs à un comportement de serveur donné forment un groupe de participants.

REMARQUE

Pour plus d'informations sur les fichiers Participant et Groupe, ainsi que sur la façon dont les fichiers EDML de Dreamweaver sont structurés, voir *Extension Data Markup Language (EDML)*, page 337.

Architecture de Dreamweaver

Lorsque vous utilisez l'outil Créateur de comportements de serveur pour créer une extension spécifique Dreamweaver, Dreamweaver génère plusieurs fichiers (fichiers de script EDML et HTML) qui prennent en charge l'insertion du code de comportement de serveur dans un document Dreamweaver (certains comportements référencent également les fichiers JavaScript pour une fonctionnalité optimale). L'architecture simplifie votre implémentation de l'API et sépare votre code d'exécution de la méthode d'application de Dreamweaver. Ce chapitre explique comment modifier ces fichiers.

Dossiers et fichiers de comportements de serveur

L'interface utilisateur pour chaque comportement de serveur réside dans le dossier Configuration/ServerBehaviors/*ServerModelName*, où *ServerModelName* correspond à l'un des types de serveur suivants : ASP.NET_Csharp, ASP.NET_VB (Visual Basic), ASP_Js (JavaScript), ASP_Vbs (VBScript), ColdFusion, JSP, PHP_MySQL ou Shared (implémentations inter-modèles de serveur).

Extension Data Markup Language (EDML)

Dreamweaver génère deux fichiers EDML lorsque vous utilisez l'outil Créateur de comportements de serveur : un fichier Groupe EDML et un fichier Participant EDML correspondant aux noms indiqués dans le Créateur de comportements de serveur. Le fichier Groupe définit les participants appropriés, qui représentent des blocs de code, et les groupes définissent les participants devant être combinés pour former un comportement de serveur individuel.

Fichiers Groupe

En général, les fichiers Groupe contiennent une liste de participants et les fichiers Participant comportent toutes les données sur le code spécifiques au modèle de serveur. Les fichiers Participant peuvent être utilisés par plusieurs extensions pour permettre à plusieurs fichiers Groupe de référencer le même fichier Participant.

L'exemple suivant présente une vue de haut niveau du fichier EDML du groupe de comportements de serveur. Pour obtenir une liste complète des éléments et des attributs, voir *Balises de fichiers EDML Groupe*, page 356.

```
<group serverBehavior="Go To Detail Page.htm" dataSource="Recordset.htm">
  <groupParticipants selectParticipant="goToDetailPage_attr">
    <groupParticipant name="moveTo_declareParam"  partType="member"/>
    <groupParticipant name="moveTo_keepParams"  partType="member"/>
    <groupParticipant name="goToDetailPage_attr" partType="identifier" />
  </groupParticipants>
</group>
```

Dans la balise de bloc `groupParticipants`, chaque balise `groupParticipant` indique le fichier Participant EDML contenant le bloc de code à utiliser. La valeur de l'attribut `name` est le nom du fichier Participant moins l'extension .edml (par exemple, l'attribut `moveTo_declareParam`).

Fichiers Participant

Un participant représente un bloc de code unique sur la page, tel une balise de serveur, une balise HTML ou un attribut. Pour pouvoir être utilisé par un auteur Dreamweaver, le fichier Participant doit figurer dans le fichier Groupe. Plusieurs groupes de fichiers peuvent utiliser un seul fichier Participant.

Par exemple, le fichier moveTo_declareParam.edml contient le code suivant :

```
<participant>
     <quickSearch><![CDATA[MM_paramName]]></quickSearch>
     <insertText location="aboveHTML+80">
<![CDATA[
```

```
<% var MM_paramName = ""; %>
]]>
     </insertText>
     <searchPatterns whereToSearch="directive">
       <searchPattern><![CDATA[/var\s*MM_paramName/]]></searchPattern>
     </searchPatterns>
</participant>
```

Pour ajouter un comportement de serveur à un document, Dreamweaver a besoin d'informations détaillées, notamment l'endroit où insérer le code, l'aspect du code et les paramètres remplacés par l'auteur ou par les données à l'exécution. C'est ce que chaque fichier EDML Participant décrit pour chaque bloc de code. Plus particulièrement, le fichier Participant décrit les données suivantes :

- Ce code et l'emplacement de l'instance unique sont définis par les paramètres de la balise `insertText`, comme illustré dans l'exemple suivant :

  ```
  <insertText location="aboveHTML+80">
  ```

- Comment reconnaître les instances déjà sur la page qui sont définies par la balise `searchPatterns`, comme illustré dans l'exemple suivant :

  ```
  <searchPatterns whereToSearch="directive">
         <searchPattern><![CDATA[/var\s*MM_paramName/]]></searchPattern>
  </searchPatterns>
  ```

Dans la balise du bloc`searchPatterns`, chaque balise `searchPattern` contient un modèle utilisé pour rechercher des instances de code d'exécution et pour extraire des paramètres spécifiques. Pour plus d'informations, voir *Techniques de comportements de serveur*, page 387.

Fichier Script

Chaque comportement de serveur dispose également d'un fichier HTML contenant les fonctions et les liens relatifs aux scripts qui gèrent l'intégration du code de comportement de serveur à l'interface Dreamweaver. Les fonctions pouvant être modifiées sont traitées dans *Fonctions d'implémentation des comportements de serveur*, page 350.

Exemple de comportement de serveur simple

L'exemple suivant illustre le processus de création d'un nouveau comportement de serveur et vous permet de voir et de gérer les fichiers générés par Dreamweaver. Pour plus d'informations sur l'utilisation de l'interface du Créateur de comportements de serveur, voir Ajout de comportements de serveur personnalisés dans *Bien démarrer avec Dreamweaver*. L'exemple affiche « Hello World » du serveur ASP. Le comportement « Hello World » contient un seul participant (une simple balise ASP) et rien n'est modifié, ni ajouté sur la page.

Pour créer un comportement, exécutez la procédure ci-dessous :

- Création du document contenant des pages dynamiques
- Définition du nouveau comportement de serveur
- Définition du code à insérer

Création du document contenant des pages dynamiques

Commencez par créer un document ASP.

Procédez comme suit pour créer un document contenant des pages dynamiques :

1. Dans Dreamweaver, choisissez Fichier > Nouveau.
2. Dans la boîte de dialogue Nouveau document, sélectionnez Catégorie : Page dynamique et Page dynamique : ASP JavaScript.
3. Cliquez sur Créer.

Définition du nouveau comportement de serveur

Définissez ensuite le nouveau comportement de serveur.

Procédez comme suit pour définir le nouveau comportement de serveur à l'aide de l'outil Créateur de comportements de serveur.

REMARQUE	Si le panneau Comportements de serveur n'est ni ouvert, ni visible, choisissez Fenêtre > Comportements de serveur.

1. Dans le panneau Comportements de serveur, sélectionnez le bouton plus (+) et l'option Nouveau comportement de serveur.
2. Dans la boîte de dialogue Nouveau comportement de serveur, sélectionnez Type de document : ASP JavaScript et nom : Hello World

   (Ne cochez pas la case Copier un comportement de serveur existant.)
3. Cliquez sur OK.

Définition du code à insérer

Définissez enfin le code à insérer.

Pour définir le code à insérer :

1. Sélectionnez le bouton plus (+) pour l'option Blocs de code à insérer.
2. Dans la boîte de dialogue Créer un nouveau bloc de code, entrez `Hello_World_block1` (Dreamweaver entre cette information automatiquement dans certains cas).
3. Cliquez sur OK.
4. Dans le champ de texte Bloc de code, entrez `<% Response.Write("Hello World") %>`.
5. Dans le menu déroulant Insérer code, sélectionnez Relatif à la sélection pour permettre à l'utilisateur de contrôler l'emplacement du code dans le document.
6. Dans le menu déroulant Position relative, sélectionnez Après la sélection.
7. Cliquez sur OK.

Dans le panneau Comportements de serveur, vous constatez que le menu plus (+) contient le nouveau comportement de serveur dans la liste déroulante. Dans le dossier d'installation de vos fichiers Dreamweaver, le dossier Configuration/ServerBehaviors/ASP_Js contient désormais les trois fichiers suivants :

- Le fichier Groupe : Hello World.edml
- Le fichier Participant : Hello World_block1.edml
- Un fichier de script : Hello World.htm

REMARQUE	Si vous travaillez dans un environnement multiutilisateur, ces fichiers s'affichent dans votre dossier Application Data.

Comment appeler les fonctions de l'API de comportement de serveur

Les fonctions de l'API de comportement de serveur sont appelées dans les situations suivantes :

- La fonction `findServerBehaviors()` est appelée à l'ouverture du document et à chaque fois qu'un participant est modifié. Elle recherche des instances du comportement de serveur dans le document utilisateur. Pour chaque instance trouvée, la fonction `findServerBehaviors()` crée un objet JavaScript et lui associe des informations d'état à l'aide de propriétés JavaScript.
- Lorsqu'elle est mise en œuvre, Dreamweaver appelle la fonction `analyzeServerBehavior()` pour chaque comportement trouvé dans le document utilisateur après que toutes les fonctions `findServerBehaviors()` ont été appelées.

  Lorsque la fonction `findServerBehaviors()` crée un objet de comportement, elle définit normalement les quatre propriétés (`incomplete`, `participants`, `selectedNode` et `title`). Cependant, il est parfois plus facile de retarder la configuration de certaines des propriétés jusqu'à ce que tous les autres comportements de serveur aient trouvé des instances d'eux-mêmes. Par exemple, le comportement Déplacer vers l'enregistrement suivant est composé de deux participants ; un objet lien et un objet jeu d'enregistrements. Plutôt que de rechercher l'objet du jeu d'enregistrements dans sa fonction `findServerBehaviors()`, attendez que la fonction `findServerBehaviors()` du comportement du jeu d'enregistrements soit exécutée, car le jeu d'enregistrements recherche toutes ses propres instances.

  Lorsque la fonction `analyzeServerBehavior()` du comportement Déplacer vers l'enregistrement suivant est appelée, un tableau lui est transmis, contenant tous les objets de comportement de serveur du document. La fonction peut rechercher l'objet de jeu d'enregistrements dans le tableau.

En cours d'analyse, il arrive que deux comportements ou plus identifient une seule balise dans le document utilisateur comme étant une instance de ce comportement. Supposons, par exemple, que la fonction `findServerBehaviors()` pour le comportement Attribut dynamique détecte une instance du comportement Attribut dynamique associée à une balise `input` dans le document utilisateur. Simultanément, la fonction `findServerBehaviors()` pour le comportement Champ de texte dynamique peut analyser la même balise `input` et détecter une instance du comportement Champ de texte dynamique. Le panneau Comportements de serveur affiche alors le bloc Attribut dynamique et le comportement Champ de texte dynamique. Pour corriger ce problème, utilisez les fonctions `analyzeServerBehavior()` pour supprimer tous les comportements de serveur redondants sauf un.

Pour supprimer un comportement de serveur, une fonction `analyzeServerBehavior()` peut définir la propriété `deleted` dudit comportement sur la valeur `true`. Si la propriété `deleted` a toujours la valeur `true` lorsque Dreamweaver finit d'appeler les fonctions `analyzeServerBehavior()`, le comportement disparaît de la liste.

- Lorsque l'utilisateur clique sur le bouton plus (+) du panneau Comportements de serveur, le menu contextuel s'affiche.

  Pour déterminer le contenu du menu, Dreamweaver recherche d'abord un fichier ServerBehaviors.xml dans le même répertoire que celui des comportements. ServerBehaviors.xml fait référence aux fichiers HTML devant apparaître dans le menu.

  Si le fichier HTML référencé contient une balise de titre (title), le contenu de cette balise s'affiche dans le menu. Par exemple, si le fichier ServerBehaviors/ASP_Js/GetRecords.htm contient la balise `<title>Get More Records</title>`, le texte Get More Records s'affiche dans le menu.

  Si le fichier ne contient pas de balise de titre, c'est le nom du fichier qui apparaît dans le menu. Par exemple, si GetRecords.htm ne contient pas de balise de titre, la mention GetRecords s'affiche dans le menu.

  S'il n'y a pas de fichier ServerBehaviors.xml ou si le répertoire contient un ou plusieurs fichiers HTML non mentionnés dans ServerBehaviors.xml, Dreamweaver recherche une balise title dans chaque fichier et fait figurer dans le menu le nom de cette balise ou celui du fichier.

  Si vous ne voulez pas qu'un fichier du répertoire ServerBehaviors s'affiche dans le menu, ajoutez l'instruction suivante sur la première ligne du fichier HTML :

  ```
  <!-- MENU-LOCATION=NONE -->
  ```

- La fonction `canApplyServerBehavior()` est appelée lorsque l'utilisateur sélectionne un élément dans le menu. Si cette fonction renvoie une valeur `true`, une boîte de dialogue s'affiche. Lorsque l'utilisateur clique sur OK, la fonction `applyServerBehavior()` est appelée.
- Si l'utilisateur double-clique sur un comportement de serveur existant pour le modifier, Dreamweaver affiche la boîte de dialogue, exécute le gestionnaire `onLoad` sur la balise `BODY` (si elle existe), puis appelle la fonction `inspectServerBehavior()`. La fonction `inspectServerBehavior()` renseigne les éléments du formulaire avec les valeurs des paramètres en cours. Lorsque l'utilisateur clique sur OK, Dreamweaver appelle à nouveau la fonction `applyServerBehavior()`.
- Si l'utilisateur clique sur le bouton moins (-), la fonction `deleteServerBehavior()` est appelée. La fonction `deleteServerBehavior()` supprime les comportements correspondants du document.
- Lorsque l'utilisateur sélectionne un comportement de serveur et utilise les commandes Couper ou Copier, Dreamweaver transmet l'objet représentant le comportement de serveur à sa fonction `copyServerBehavior()`. La fonction `copyServerBehavior()` ajoute à l'objet de comportement les autres propriétés nécessaires à son collage ultérieur.

  Une fois la fonction `copyServerBehavior()` renvoyée, Dreamweaver convertit l'objet de comportement en un formulaire qui peut être placé dans le Presse-papiers. Pendant la conversion de l'objet, Dreamweaver supprime toutes les propriétés faisant référence à des objets ; toute propriété de l'objet autre qu'un nombre, un opérateur booléen ou une chaîne est perdue.

  Lorsque l'utilisateur appuie sur la commande Coller, Dreamweaver décompresse le contenu du Presse-papiers et génère un nouvel objet de comportement de serveur. Ce nouvel objet est identique à l'original, sauf qu'il ne contient aucune propriété faisant référence à des objets. Dreamweaver transmet le nouvel objet de comportement à la fonction `pasteServerBehavior()`. La fonction `pasteServerBehavior()` ajoute le comportement au document utilisateur. Une fois la fonction `pasteServerBehavior()` renvoyée, Dreamweaver appelle les fonctions `findServerBehaviors()` pour obtenir la nouvelle liste des comportements de serveur présents dans le document utilisateur.

Les utilisateurs peuvent copier et coller des comportements d'un document à un autre. Pour échanger des informations, les fonctions `copyServerBehavior()` et `pasteServerBehavior()` doivent se baser uniquement sur les propriétés de l'objet de comportement.

API de comportement de serveur

Les fonctions suivantes sont les API que vous pouvez utiliser pour gérer des comportements de serveur.

analyzeServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction permet aux comportements de serveur de définir les propriétés `incomplete` et `deleted`.

Une fois la fonction `findServerBehaviors()` appelée pour tous les comportements de serveur sur la page, un tableau de tous les comportements s'affiche dans le document utilisateur. Pour chaque objet JavaScript présent dans le tableau, la fonction `analyzeServerBehavior()` est appelée. Par exemple, pour le comportement Texte dynamique, Dreamweaver appelle la fonction `analyzeServerBehavior()` dans le fichier DynamicText.htm (ou DynamicText.js).

La fonction `analyzeServerBehavior()` a notamment pour but de terminer la définition de toutes les propriétés (`incomplete`, `participants`, `selectedNode` et `title`) sur l'objet de comportement. Il est parfois plus facile d'exécuter cette tâche après que la fonction `findServerBehaviors()` a généré une liste exhaustive des comportements de serveur présents dans le document utilisateur.

L'autre but de la fonction `analyzeServerBehavior()` est de rechercher si plusieurs comportements se réfèrent à la même balise dans le document utilisateur. Dans ce cas, la propriété `deleted` est utilisée pour supprimer du tableau tous les comportements sauf un.

Supposons que les comportements de serveur Recordset1, DynamicText1 et DynamicText2 se trouvent sur une page. Les deux comportements de serveur DynamicText requièrent Recordset1 pour exister sur la page. Une fois les comportements de serveur détectés par le biais de la fonction `findServerBehaviors()`, Dreamweaver appelle la fonction `analyzeServerBehavior()` pour les trois comportements de serveur. Lorsque la fonction `analyzeServerBehavior()` est appelée pour DynamicText1, elle inspecte le tableau de tous les objets de comportement de serveur présents sur la page, à la recherche de celui qui appartient à Recordset1. Si elle ne trouve pas l'objet de comportement de serveur qui appartient à Recordset1, la propriété `incomplete` est affectée de la valeur `true` et un point d'exclamation est affiché dans le panneau Comportements de serveur, indiquant à l'utilisateur qu'il y a un problème. De même, lorsque la fonction `analyzeServerBehavior()` est appelée pour DynamicText2, elle recherche l'objet appartenant à Recordset1. Comme Recordset1 ne dépend d'aucun autre comportement de serveur, il n'a pas besoin de définir la fonction `analyzeServerBehavior()` dans cet exemple.

Arguments

`serverBehavior`, *`[serverBehaviorArray]`*

- L'argument *`serverBehavior`* est un objet JavaScript représentant le comportement à analyser.
- L'argument `[serverBehaviorArray]` est un tableau d'objets JavaScript représentant l'ensemble des comportements de serveur trouvés sur une page.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

applyServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction consulte les valeurs des éléments de formulaire dans la boîte de dialogue et ajoute le comportement au document utilisateur. Dreamweaver appelle cette fonction lorsque l'utilisateur clique sur dans la boîte de dialogue Comportements de serveur. Si la fonction est exécutée avec succès, la boîte de dialogue Comportements de serveur est fermée. Si elle échoue, la boîte de dialogue Comportements de serveur reste ouverte et un message d'erreur s'affiche à l'écran. Cette fonction peut modifier un document utilisateur.

Pour plus d'informations, consultez la section *dwscripts.applySB()*, page 351.

Arguments

serverBehavior

- *serverBehavior* est un objet JavaScript représentant le comportement de serveur ; il est nécessaire de modifier un comportement existant. S'il s'agit d'un nouveau comportement, l'argument est `null`.

Valeurs renvoyées

Dreamweaver attend une chaîne vide en cas de succès ou un message d'erreur en cas d'échec.

canApplyServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction détermine si un comportement peut être appliqué. Dreamweaver appelle cette fonction avant d'afficher la boîte de dialogue Comportements de serveur. Si la fonction renvoie la valeur `true`, la boîte de dialogue Comportements de serveur s'affiche. Si la fonction renvoie la valeur `false`, la boîte de dialogue Comportements de serveur ne s'affiche pas et la tentative d'ajout d'un comportement de serveur est interrompue.

Arguments

serverBehavior

- *serverBehavior* est un objet JavaScript représentant le comportement ; il est nécessaire de modifier un comportement existant. S'il s'agit d'un nouveau comportement, l'argument est `null`.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le comportement peut être appliqué, sinon `false`.

copyServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

L'utilisation de la fonction `copyServerBehavior()` est facultative. Cette fonction permet aux utilisateurs de copier des instances du comportement de serveur spécifié. Dans l'exemple suivant, cette fonction est utilisée pour les jeux d'enregistrements. Lorsqu'un utilisateur sélectionne un jeu d'enregistrements dans les panneaux Comportements de serveur ou Liaisons de données, l'utilisation de la commande Copier copie le comportement dans le Presse-papiers, tandis que la commande Couper le supprime de ce dernier. Les commandes Copier et Couper n'ont aucun effet sur les comportements de serveur qui ne mettent pas en œuvre cette fonction. Pour plus d'informations, consultez la section *Comment appeler les fonctions de l'API de comportement de serveur*, page 341.

Pour échanger des informations, les fonctions `copyServerBehavior()` et `pasteServerBehavior()` doivent uniquement se baser sur les propriétés de l'objet de comportement capables d'être converties en chaînes. Le Presse-papiers ne contient que du texte brut ; il est, pour cette raison, nécessaire de résoudre les nœuds `participant` dans le document et d'enregistrer le texte brut qui en résulte dans une propriété secondaire.

REMARQUE

Il est également important de mettre en œuvre la fonction `pasteServerBehavior()` pour permettre à l'utilisateur de coller le comportement dans un document Dreamweaver.

Arguments

`serverBehavior`

- *`serverBehavior`* est un objet JavaScript représentant le comportement.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le comportement est copié avec succès dans le Presse-papiers, sinon `false`.

deleteServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction supprime le comportement du document utilisateur. Elle est appelée lorsque l'utilisateur clique sur le bouton moins (-) du panneau Comportements de serveur. Elle permet de modifier un document utilisateur.

Pour plus d'informations, consultez la section *dwscripts.deleteSB()*, page 352.

Arguments

serverBehavior

- *serverBehavior* est un objet JavaScript représentant le comportement.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

displayHelp()

Description

Si cette fonction est définie, un bouton Aide s'affiche sous les boutons OK et Annuler dans la boîte de dialogue. Cette fonction est appelée lorsque l'utilisateur clique sur le bouton Aide.

Arguments

Néant

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

```
// Dans l'exemple suivant, la fonction displayHelp() ouvre
// dans un navigateur un fichier expliquant comment utiliser
// l'extension.
function displayHelp(){
  var myHelpFile = dw.getConfigurationPath() +
    '/ExtensionsHelp/superDuperHelp.htm';
  dw.browseDocument(myHelpFile);
}
```

findServerBehaviors()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction parcourt le document utilisateur à la recherche d'instances d'elle-même. Pour chaque instance trouvée, la fonction `findServerBehaviors()` crée un objet JavaScript et lui associe des informations d'état en tant que propriétés JavaScript de l'objet.

Les quatre propriétés obligatoires sont `incomplete`, `participants`, `title` et `selectedNode`. Vous pouvez définir d'autres propriétés en fonction de vos besoins.

Pour plus d'informations, voir dwscripts.findSBs() et `dreamweaver.getParticipants()` dans le *Guide des API de Dreamweaver*.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau d'objets JavaScript ; la longueur du tableau correspond au nombre d'instances du comportement trouvées dans la page.

inspectServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction détermine les paramètres de la boîte de dialogue Comportements de serveur en fonction de l'objet de comportement spécifié. Dreamweaver appelle la fonction `inspectServerBehavior()` lorsqu'un utilisateur ouvre une boîte de dialogue de comportement de serveur. Dreamweaver appelle cette fonction uniquement lorsque l'utilisateur modifie un comportement existant.

Arguments

`serverBehavior`

- L'argument *`serverBehavior`* est un objet JavaScript représentant le comportement de serveur. Il s'agit du même objet que celui renvoyé par`findServerBehaviors()`.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

pasteServerBehavior()

Disponibilité

Dreamweaver UltraDev 1.

Description

Si cette fonction est implémentée, les utilisateurs peuvent coller des instances du comportement de serveur spécifié à l'aide de la fonction `pasteServerBehavior()`. Lorsque l'utilisateur colle le comportement de serveur, Dreamweaver organise le contenu du Presse-papiers et génère un nouvel objet de comportement. Ce nouvel objet est identique à l'original, sauf qu'il ne contient aucune propriété de pointeur. Dreamweaver transmet le nouvel objet de comportement à la fonction `pasteServerBehavior()`. La fonction `pasteServerBehavior()` se base sur les propriétés de l'objet de comportement pour déterminer ce qui doit être ajouté au document utilisateur. La fonction `pasteServerBehavior()` ajoute ensuite le comportement au document utilisateur. Une fois la fonction `pasteServerBehavior()` renvoyée, Dreamweaver appelle la fonction `findServerBehaviors()` pour obtenir la nouvelle liste des comportements de serveur présents dans le document utilisateur.

L'utilisation de la fonction `pasteServerBehavior()` est facultative. Pour plus d'informations, consultez la section *Comment appeler les fonctions de l'API de comportement de serveur*, page 341.

REMARQUE

Si vous implémentez cette fonction, vous devez également implémenter la fonction `copyServerBehavior()`.

Arguments

`comportement`

- L'objet JavaScript *`behavior`* représente le comportement.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le comportement est collé avec succès depuis le Presse-papiers, sinon `false`.

Fonctions d'implémentation des comportements de serveur

Vous pouvez ajouter ou modifier ces fonctions dans les fichiers de script HTML, ou dans les fichiers JavaScript répertoriés dans le fichier de script HTML.

dwscripts.findSBs()

Disponibilité

Dreamweaver MX (cette fonction remplace la fonction `findSBs()` des versions précédentes de Dreamweaver).

Description

Cette fonction recherche toutes les instances d'un comportement de serveur sur la page en cours, ainsi que tous les participants. Elle définit le titre, le type, les tableaux des participants, des poids et des types, la valeur `selectedNode` et l'indicateur « incomplete ». Elle crée également un objet de paramètre contenant un tableau de propriétés définissables par l'utilisateur (le jeu d'enregistrements, le nom et le nom d'une colonne, par exemple). Vous pouvez renvoyer ce tableau depuis la fonction `findServerBehaviors()`.

Arguments

`serverBehaviorTitle`

- L'argument *`serverBehaviorTitle`* est une chaîne de titre facultative utilisée si aucun titre n'est spécifié dans le titre EDML, utile pour la localisation.

Valeurs renvoyées

Dreamweaver attend un tableau d'objets JavaScript, avec les propriétés requises définies. Cette fonction renvoie un tableau vide si aucun comportement de serveur n'apparaît sur la page.

Exemple

L'exemple suivant recherche dans le document utilisateur en cours toutes les instances d'un comportement de serveur particulier :

```
function findServerBehaviors() {
  allMySBs = dwscripts.findSBs();
  return allMySBs;
}
```

dwscripts.applySB()

Disponibilité

Dreamweaver MX (cette fonction remplace la fonction `applySB()` des versions précédentes de Dreamweaver).

Description

Cette fonction insère ou met à jour le code d'exécution du comportement de serveur. Si le paramètre *sbObj* a une valeur `null`, la fonction insère un nouveau code d'exécution ; dans le cas contraire, elle met à jour le code d'exécution existant indiqué par l'objet *sbObj*. Définissez les paramètres utilisateur sous la forme de propriétés sur un objet JavaScript, puis transmettez-les en tant que *paramObj*. Ils doivent correspondre à tous les paramètres déclarés sous la forme `@@paramName@@` dans le texte d'insertion XML.

Arguments

paramObj, sbObj

- L'argument *paramObj* est l'objet contenant les paramètres utilisateur.
- L'argument *sbObj* est l'objet du comportement de serveur précédent si vous mettez à jour un comportement de serveur existant, sinon `null`.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si le comportement est copié avec succès dans le document utilisateur, sinon `false`.

Exemple

Dans l'exemple suivant, insérez dans l'objet `paramObj` les entrées utilisateur et appelez la fonction `dwscripts.applySB()` en transmettant les données et votre comportement de serveur, `sbObj` :

```
function applyServerBehaviors(sbObj) {

  // get all UI values here...
  paramObj = new Object();
  paramObj.rs      = rsName.value;
  paramObj.col     = colName.value;
  paramObj.url     = urlPath.value;
  paramObj.form__tag = formObj;

  dwscripts.applySB(paramObj, sbObj);
}
```

dwscripts.deleteSB()

Disponibilité

Dreamweaver MX (cette fonction remplace la fonction `deleteSB()` des versions précédentes de Dreamweaver).

Description

Cette fonction supprime tous les participants de l'instance du comportement de serveur `sbObj`. Le participant dans son ensemble est supprimé, à moins que le fichier EDML n'indique des instructions de suppression spéciales à l'aide de la balise `delete`. Les participants appartenant à plusieurs instances de comportement de serveur (comptage de référence > 1) ne sont pas supprimés.

Arguments

`sbObj`

- L'argument *`sbObj`* est l'instance de l'objet du comportement de serveur à supprimer du document utilisateur.

Valeurs renvoyées

Dreamweaver ne renvoie rien.

Exemple

L'exemple suivant illustre la suppression de tous les participants du comportement de serveur `sbObj`, hormis ceux qui sont protégés par la balise `delete` du fichier EDML.

```
function deleteServerBehavior(sbObj) {
  dwscripts.deleteSB(sbObj);
}
```

Modification des fichiers EDML

Vous devez respecter les conventions de codage Dreamweaver lorsque vous modifiez un fichier. Faites très attention à la dépendance d'un élément par rapport à un autre. Par exemple, si vous mettez à jour les balises que vous insérez, il est possible que vous deviez également mettre à jour les modèles de recherche.

REMARQUE	Les fichiers EDML ont été ajoutés avec Dreamweaver MX. Si vous travaillez avec des comportements de serveur existants, reportez-vous aux versions précédentes des manuels *Extension de Dreamweaver*.

Expressions régulières

Vous devez connaître les expressions régulières implémentées dans JavaScript 1.5. Vous devez également savoir à quel moment il convient de les utiliser dans les fichiers EDML de comportements de serveur. Par exemple, les expressions régulières ne peuvent pas être utilisées dans les valeurs `quickSearch`. En revanche, elles sont employées dans le contenu de la balise `searchPattern` pour la recherche et l'extraction de données.

Les expressions régulières décrivent des chaînes de texte en utilisant des caractères qui ont une signification spéciale (métacaractères) pour représenter le texte, le décomposer et le traiter selon des règles prédéfinies. Les expressions régulières sont des outils d'analyse et de traitement puissants car elles fournissent une méthode générique de représentation d'un modèle.

Tout bon manuel de référence sur JavaScript 1.5 comporte une section ou un chapitre consacré aux expressions régulières. Dans cette section, nous examinons la façon dont les fichiers EDML de comportements de serveur Dreamweaver utilisent les expressions régulières pour rechercher des paramètres dans votre code d'exécution et pour extraire leurs valeurs. Chaque fois qu'un utilisateur modifie un comportement de serveur, les valeurs des paramètres doivent être extraites des instances du code d'exécution. Utilisez des expressions régulières pour la procédure d'extraction.

Vous devez connaître quelques métacaractères et métaséquences (regroupements de caractères spéciaux) utiles dans les fichiers EDML de comportements de serveur, comme ceux décrits dans le tableau suivant :

Expression régulière	Description
\	Arrêter le mode caractères spéciaux. Exemple : \. rétablit le métacaractère à un point littéral, \/ rétablit la barre oblique à son sens littéral et \) rétablit la parenthèse à son sens littéral.
/ ... /i	Ne pas tenir compte de la casse lors de la recherche d'une métaséquence.
(...)	Créer une sous-expression entre parenthèses dans la métaséquence.
\s*	Rechercher des espaces blancs.

La balise EDML `<searchPatterns whereToSearch="directive">` indique qu'une recherche doit être effectuée dans le code d'exécution. Chaque sous-balise `<searchPattern>...</searchPattern>` définit un modèle dans le code d'exécution devant être identifié. L'exemple Redirect If Empty contient deux modèles.

Dans l'exemple suivant, pour extraire des valeurs de paramètres de `<% if (@@rs@@.EOF) Response.Redirect("@@new__url@@"); %>`, rédigez une expression régulière qui identifie les chaînes `rs` et `new__url` :

```
<searchPattern paramNames="rs,new__url">
  /if d ((\w+)\.EOF\) Response\.Redirect\("([^\r\n]*)"\)/i
</searchPattern>
```

Ce processus effectue une recherche dans le document de l'utilisateur et, si une correspondance est trouvée, les valeurs de paramètre sont extraites. La première sous-expression entre parenthèses (`\w+`) extrait la valeur de `rs`. La deuxième sous-expression (`[^\r\n]*`) extrait la valeur de `new_url`.

REMARQUE

La séquence de caractères "`[^\r\n]*`" correspond à tout caractère sauf à un saut de ligne, sur Macintosh et Windows.

Quelques mots sur la structure EDML

Vous devez utiliser un nom de fichier unique pour identifier votre groupe de comportements de serveur. Si un fichier Participant associé n'est utilisé que par un fichier Groupe, vous devez donner à votre fichier Participant un nom correspondant à celui du groupe. Cette convention permet au fichier Groupe de comportements de serveur `updateRecord.edml` de fonctionner avec le fichier Participant `updateRecord_init.edml`. Lorsque les fichiers Participant sont destinés au partage entre plusieurs groupes de comportements de serveur, vous devez leur donner un nom descriptif et unique.

REMARQUE

L'espace de nom EDML étant partagé indépendamment de la structure de dossier, assurez-vous d'utiliser des noms de fichiers uniques. Les noms de fichiers ne doivent pas comporter plus de 31 caractères (y compris l'extension .edml) en raison des limitations Macintosh.

Le code d'exécution de votre comportement de serveur réside à l'intérieur de fichiers EDML. Le programme d'analyse EDML ne doit confondre aucun de vos codes d'exécution avec les balises EDML. C'est pourquoi la balise `CDATA` doit envelopper votre code d'exécution. La balise `CDATA` représente des données de caractère et correspond à tout texte autre qu'une balise EDML. Lorsque vous utilisez la balise `CDATA`, le programme d'analyse EDML n'essaie pas de l'interpréter en tant que balise, mais l'assimile plutôt à un bloc de texte brut. Les blocs marqués par la balise `CDATA` commencent par `<![CDATA[` et se terminent par `]]>`.

Si vous insérez le texte « `Hello, World` », vous pouvez spécifier aisément votre EDML comme suit :

```
<insertText>Hello, World</insertText>
```

Cependant, si vous insérez un texte comportant des balises, tel `<img src='foo.gif'>`, cela peut être source de confusion pour le programme d'analyse EDML. Dans ce cas, incorporez votre texte dans la construction `CDATA` comme indiqué dans l'exemple suivant :

```
<insertText><![CDATA[<img src='foo.gif'>]]></insertText>
```

Le code d'exécution ASP est enveloppé par la balise `CDATA` tag, comme illustré dans l'exemple suivant :

```
<![CDATA[
  <% if (@@rs@@.EOF) Response.Redirect("@@new__url@@"); %>
]]
```

En raison de la balise `CDATA`, les balises `<%= %>`, ainsi que tout contenu interne, ne sont pas traitées. Au lieu de cela, le gestionnaire de données d'extension (EDM) reçoit le texte non interprété, comme indiqué dans l'exemple suivant :

```
<% if (Recordset1.EOF) Response.Redirect("http://www.macromedia.com"); %>
```

Dans les définitions EDML suivantes, les emplacements où la balise `CDATA` est recommandée sont indiqués dans les exemples.

Balises de fichiers EDML Groupe

Ces balises et attributs sont valides dans les fichiers Groupe EDML.

<group>

Description

Cette balise contient toutes les spécifications relatives à un groupe de participants.

Parent

Néant

Type

Balise de bloc.

Obligatoire

Oui.

Attributs de ‹group›

Les éléments suivants sont des attributs valides de la balise group.

version

Description

Cet attribut définit le traitement de comportement de serveur de Dreamweaver cible du comportement de serveur actif. Pour Dreamweaver 8, le numéro de version est 7. Si aucune version n'est spécifiée, Dreamweaver suppose qu'il s'agit de la version 7. Pour cette version de Dreamweaver, l'attribut de version est réglé sur 7.0 pour tous les groupes et participants générés par le Créateur de comportements de serveur. La version de groupe de cet attribut n'a aucun effet actuellement.

Parent

`group`

Type

Attribut.

Obligatoire

Non.

serverBehavior

Description

L'attribut `serverBehavior` indique le comportement de serveur qui peut utiliser le groupe. Lorsque l'une des chaînes `quickSearch` du participant du groupe est trouvée dans le document, le comportement de serveur indiqué par l'attribut `serverBehavior` demande à Dreamweaver d'appeler la fonction `findServerBehaviors()`.

Dans certains cas, si plusieurs groupes sont associés à un seul et même comportement de serveur, ce dernier doit déterminer le groupe qu'il veut utiliser.

Parent

`group`

Type

Attribut.

Obligatoire

Non.

Valeur

La valeur est le nom exact (sans chemin) d'un fichier HTML de comportement de serveur dans un dossier Configuration/ServerBehaviors, comme indiqué dans l'exemple suivant :

```
<group serverBehavior="redirectIfEmpty.htm">
```

dataSource

Description

Cette fonction avancée prend en charge les nouvelles sources de données qui peuvent être ajoutées à Dreamweaver.

Les versions d'un comportement de serveur peuvent différer, selon la source de données utilisée. Par exemple, le comportement de serveur Région répétée est conçu pour la source de données Recordset.htm standard. Si Dreamweaver est développé afin de prendre en charge un nouveau type de source de données (un objet COM, par exemple), vous pouvez configurer `dataSource="COM.htm"` dans un fichier Groupe avec une implémentation différente de Région répétée. Le comportement de serveur Région répétée applique alors la nouvelle implémentation de Région répétée si la nouvelle source de données est sélectionnée.

Parent

`group`

Type

Attribut.

Obligatoire

Non.

Valeur

Le nom exact d'un fichier source de données d'un dossier Configuration/DataSources, comme indiqué dans l'exemple suivant :

```
<group serverBehavior="Repeat Region.htm" ¬
dataSource="myCOMdataSource.htm">
```

Ce groupe définit une nouvelle implémentation du comportement de serveur Région répétée à utiliser si la source de données COM est utilisée. Dans la fonction `applyServerBehaviors()`, vous pouvez indiquer que ce groupe doit être appliqué en définissant la propriété `MM_dataSource` sur l'objet de paramètre, comme indiqué dans l'exemple suivant :

```
function applyServerBehavior(ssRec) {
  var paramObj = new Object();
  paramObj.rs = getComObjectName();
```

```
  paramObj.MM_dataSource = "myCOMdataSource.htm";

  dwscripts.applySB(paramObj, sbObj);
}
```

subType

Description

Cette fonction avancée prend en charge plusieurs implémentations d'un comportement de serveur.

Les versions d'un comportement de serveur peuvent différer, selon la sélection de l'utilisateur. Lorsqu'un comportement de serveur est appliqué mais que plusieurs fichiers Groupe sont appropriés, le fichier Groupe correct peut être sélectionné par transmission d'une valeur `subType`. Le groupe ayant cette valeur `subType` spécifique est appliqué.

Parent

`group`

Type

Attribut.

Obligatoire

Non.

Valeur

La valeur est une chaîne unique déterminant le groupe à appliquer, comme indiqué dans l'exemple suivant :

```
<group serverBehavior="myServerBehavior.htm" ¬
subType="longVersion">
```

Cet attribut de groupe définit la version longue du sous-type `myServerBehavior`. Il existe également une version avec l'attribut `subType="shortVersion"`. Dans la fonction `applyServerBehaviors()`, vous pouvez indiquer quel groupe doit être appliqué en définissant la propriété `MM_subType` sur l'objet de paramètre, comme indiqué dans l'exemple suivant :

```
function applyServerBehavior(ssRec) {
  var paramObj = new Object();
  if (longVersionChecked) {
    paramObj.MM_subType = "longVersion";
  } else {
    paramObj.MM_subType = "shortVersion";
  }
```

```
  dwscripts.applySB(paramObj, sbObj);
}
```

‹title›

Description

Cette chaîne s'affiche dans le panneau Comportements de serveur pour chaque instance de comportement de serveur trouvée dans le document actif.

Parent

`group`

Type

Balise de bloc.

Obligatoire

Non.

Valeur

La valeur est une chaîne de texte brut pouvant comporter des noms de paramètre pour rendre chaque instance unique, comme indiqué dans l'exemple suivant :

```
<title>Redirect If Empty (@@recordsetName@@)</title>
```

‹groupParticipants›

Description

Cette balise contient un tableau de déclarations `groupParticipant`.

Parent

`group`

Type

Balise de bloc.

Obligatoire

Oui.

Attributs de ‹groupParticipants›

Les éléments suivants sont des attributs valides de la balise groupParticipants.

selectParticipant

Description

Indique quel participant doit être sélectionné et mis en surbrillance dans le document lorsqu'une instance est sélectionnée dans le panneau Comportements de serveur. Les instances de comportements de serveur énumérées dans ce panneau étant classées en fonction du participant sélectionné, vous devez définir l'attribut `selectParticipant` même si le participant n'est pas visible.

Parent

`groupParticipants`

Type

Attribut.

Obligatoire

Non.

Valeur

La valeur *`participantName`* est le nom exact (moins l'extension .edml) d'un seul fichier Participant référencé comme un participant Groupe, comme indiqué dans l'exemple suivant
Voir *name*, page 362.

```
<groupParticipants selectParticipant="redirectIfEmpty_link">
```

‹groupParticipant›

Description

Cette balise représente l'inclusion d'un participant unique dans le groupe.

Parent

`groupParticipants`

Type

Balise.

Obligatoire

Oui (au moins un).

Attributs de ‹groupParticipant›

Les éléments suivants sont des attributs valides de la balise groupParticipant.

name

Description

Cet attribut désigne un participant particulier à inclure dans le groupe. L'attribut `name` figurant dans la balise `groupParticipant` doit être le même que le nom de fichier du participant (sans l'extension .edml).

Parent

`groupParticipant`

Type

Attribut.

Obligatoire

Oui.

Valeur

La valeur est le nom exact (sans l'extension .edml) d'un fichier Participant quelconque, comme indiqué dans l'exemple suivant :

```
<groupParticipant name="redirectIfEmpty_init">
```

Cet exemple fait référence au fichier redirectIfEmpty_init.edml.

partType

Description

Cet attribut indique le type de participant.

Parent

`groupParticipant`

Type

Attribut.

Obligatoire

Non.

Valeurs

`identifier`, *`member`*, *`option`*, *`multiple`*, *`data`*

- La valeur *identifier* est un participant qui identifie le groupe entier. Si ce participant est trouvé dans le document, le groupe est considéré comme existant, que d'autres participants du groupe soient trouvés ou non. Il s'agit de la valeur par défaut si l'attribut `partType` n'est pas spécifié.
- La valeur *member* est un membre normal d'un groupe. Si trouvé seul, il n'identifie pas un groupe. S'il n'est pas trouvé dans un groupe, le groupe est jugé incomplet.
- La valeur *option* indique que le participant est facultatif. S'il n'est pas trouvé, le groupe est quand même jugé complet et aucun indicateur « incomplete » n'est affiché dans le panneau Comportements de serveur.
- La valeur *multiple* indique que le participant est facultatif et que plusieurs de ses copies peuvent être associées au comportement de serveur. Aucun des paramètres uniques à ce participant ne sera utilisé lors du regroupement de participants car ils peuvent avoir des valeurs différentes.
- La valeur *data* est un participant non standard utilisé par les programmeurs en tant que référentiel des données de groupe supplémentaires. Elle n'est prise en compte par rien d'autre.

Fichiers EDML Participant

Ces balises et attributs sont valides dans les fichiers EDML Participant.

<participant>

Description

Cette balise contient toutes les spécifications d'un participant unique.

Parent

Néant

Type

Balise de bloc.

Obligatoire

Oui.

Attributs de <participant>

Les éléments suivants sont des attributs valides de la balise participant.

version

Description

Cet attribut définit le traitement de comportement de serveur de Dreamweaver cible du comportement de serveur actif. Pour Dreamweaver 8, le numéro de version est 7. Si aucune version n'est spécifiée, Dreamweaver suppose qu'il s'agit de la version 7. Pour cette version de Dreamweaver, l'attribut de version est réglé sur 7.0 pour tous les groupes et participants générés par le Créateur de comportements de serveur.

REMARQUE	L'attribut de version participant annule l'attribut de version groupe si ceux-ci sont différents. Néanmoins, le fichier participant utilisera uniquement l'attribut de version groupe si le participant n'en spécifie aucun.

Pour les fichiers Participant, cet attribut détermine s'il faut effectuer la fusion de blocs de code. Pour les participants qui ne disposent pas de cet attribut (ou si l'attribut est défini sur la version 4 ou précédente), les blocs de code insérés ne sont pas fusionnés avec les autres blocs de code sur la page. Les participants dont l'attribut est défini sur version 5 ou plus, sont fusionnés avec les autres blocs de code sur la page, lorsque possible. Notez que la fusion de blocs de code s'effectue uniquement pour les participants au-dessus et en dessous de la balise HTML.

Parent

`participant`

Type

Attribut.

Obligatoire

Non.

‹quickSearch›

Description

Cette balise est une simple chaîne de recherche utilisée pour améliorer les performances. Il ne peut pas s'agir d'une expression régulière. Si la chaîne est trouvée dans le document actif, les autres modèles de recherche sont appelés pour rechercher des instances spécifiques. La chaîne peut être vide pour toujours utiliser le modèle de recherche.

Parent
`participant`

Type
Balise de bloc.

Obligatoire
Non.

Valeur
La valeur *`searchString`*, chaîne littérale qui existe sur la page si le participant existe. La chaîne doit être la plus unique possible afin d'améliorer les performances, mais elle ne doit pas nécessairement être unique. Bien que cette chaîne ne soit pas sensible à la casse, prenez garde aux espaces non essentiels qui peuvent être modifiés par l'utilisateur, comme indiqué dans l'exemple suivant :

```
<quickSearch>Response.Redirect</quickSearch>
```

Si la balise `quickSearch` est vide, elle est considérée comme correspondante et des recherches plus précises utilisent les expressions régulières définies dans les balises `searchPattern`. Cela peut être utile lorsqu'une simple chaîne ne peut pas être utilisée pour exprimer un modèle de recherche fiable et que des expressions régulières sont obligatoires.

<insertText>

Description
Cette balise donne des indications sur ce qui doit être inséré dans le document et à quel endroit. Elle contient le texte à insérer. Les parties du texte qui sont personnalisées doivent être indiquées à l'aide du format `@@parameterName@@`.

Dans certains cas, par exemple pour un participant traducteur uniquement, vous n'avez probablement pas besoin de cette balise.

Parent
`implementation`

Type
Balise de bloc.

Obligatoire
Non.

Valeur

La valeur est le texte à insérer dans le document. Si des parties du texte doivent être personnalisées, elles peuvent être transmises ultérieurement en tant que paramètres. Les paramètres doivent être incorporés entre deux signes (@@). Ce texte pouvant interférer avec la structure EDML, il doit utiliser la construction `CDATA`, comme indiqué dans l'exemple suivant :

```
<insertText location="aboveHTML">
  <![CDATA[<%= @@recordset@@).cursorType %>]]>
</insertText>
```

Lorsque le texte est inséré, le paramètre `@@recordset@@` est remplacé par un nom de jeu d'enregistrements fourni par l'utilisateur. Pour plus d'informations sur les blocs de code conditionnels et répétitifs, voir le chapitre Ajout de comportements de serveur personnalisés dans *Bien démarrer avec Dreamweaver*.

Attributs de <insertText>

Les éléments suivants sont des attributs valides de la balise insertText.

emplacement

Description

Cet attribut indique où le texte participant doit être inséré. L'emplacement d'insertion étant relatif à l'attribut `whereToSearch` de la balise `searchPatterns`, veillez à définir ces deux paramètres soigneusement (voir `whereToSearch`, page 370).

Parent

`insertText`

Type

Attribut.

Obligatoire

Oui.

Valeurs

aboveHTML[+weight], *belowHTML[+weight]*, *beforeSelection*, *replaceSelection*, *wrapSelection*, *afterSelection*, *beforeNode*, *replaceNode*, *afterNode*, *firstChildOfNode*, *lastChildOfNode*, *nodeAttribute[+attribute]*

- La valeur *aboveHTML[+weight]* insère le texte au-dessus de la balise HTML (appropriée pour le code serveur uniquement). Le poids peut être un nombre entier compris entre 1 et 99 et est utilisé pour préserver l'ordre relatif entre les différents participants. Par convention, les jeux d'enregistrements ont un poids de 50. Si un participant fait référence à des variables de jeu d'enregistrements, il doit par conséquent être affecté d'un poids plus élevé (par exemple, 60) de façon à ce que le code soit inséré au-dessous du jeu d'enregistrements, comme indiqué dans l'exemple suivant :

  ```
  <insert location="aboveHTML+60">
  ```

  Si aucun poids n'est indiqué, un poids de 100 est attribué en interne et ajouté au-dessous de tous les participants de poids spécifique, comme indiqué dans l'exemple suivant :

  ```
  <insert location="aboveHTML">
  ```

- La valeur *belowHTML[+weight]* est similaire à l'emplacement aboveHTML, si ce n'est que les participants sont ajoutés au-dessous de la balise /HTML de fermeture.
- La valeur *beforeSelection* insère le texte avant le point d'insertion ou la sélection actuelle. S'il n'y a pas de sélection, il est inséré à la fin de la balise BODY.
- La valeur *replaceSelection* remplace la sélection actuelle par le texte. S'il n'y a pas de sélection, il est inséré à la fin de la balise BODY.
- La valeur *wrapSelection* équilibre la sélection actuelle, insère une balise de bloc avant la sélection et ajoute la balise de fin appropriée après la sélection.
- La valeur *afterSelection* insère le texte après la sélection actuelle ou le point d'insertion. S'il n'y a pas de sélection, il est inséré à la fin de la balise BODY.
- La valeur *beforeNode* insère le texte avant un nœud (emplacement spécifique dans le DOM). Lorsqu'une fonction telle que dwscripts.applySB() est appelée pour effectuer l'insertion, le pointeur de nœud doit être transmis en tant que paramètre de *paramObj*. Le nom de ce paramètre définissable par l'utilisateur doit être spécifié par l'attribut nodeParamName (voir la section *nodeParamName*, page 368).

  En résumé, si votre emplacement comporte le mot node, veillez à déclarer la balise nodeParamName.
- La valeur *replaceNode* remplace un nœud par le texte.
- La valeur *afterNode* insère le texte après un nœud.
- La valeur *firstChildOfNode* insère le texte en tant que premier enfant d'une balise de bloc. Par exemple, pour insérer un élément au début d'une balise FORM.

- *firstChildOfNode* insère le texte en tant que dernier enfant d'une balise de bloc. Par exemple, si vous souhaitez insérer du code à la fin d'une balise FORM (utile pour l'ajout de champs de formulaire masqués).
- *nodeAttribute[+attribute]* définit un attribut d'un nœud de balise. Si l'attribut n'existe pas déjà, cette valeur le crée.

  Par exemple, utilisez <insert location="nodeAttribute+ACTION" nodeParamName="form"> pour définir l'attribut ACTION d'un formulaire. Cette variante convertit la balise FORM de l'utilisateur de <form> en <form action="myText">.

  Si vous ne spécifiez pas d'attribut, l'emplacement *nodeAttribute* provoque l'ajout direct du texte à la balise ouverte. Par exemple, utilisez insert location="nodeAttribute" pour ajouter un attribut facultatif à une balise. Vous pouvez procéder ainsi pour convertir la balise INPUT d'un utilisateur de <input type="checkbox"> en <input type="checkbox" <%if(foo)Reponse.Write("CHECKED")%>> .

REMARQUE	Pour la valeur d'attribut location="nodeAttribute", le dernier modèle de recherche détermine les points de début et de fin de l'attribut. Veillez à ce que le dernier modèle trouve l'instruction entière.

nodeParamName

Description

Cet attribut est uniquement utilisé pour les emplacements d'insertion relatifs à un nœud. Il indique le nom du paramètre utilisé pour transmettre le nœud au moment de l'insertion.

Parent

insertText

Type

Attribut.

Obligatoire

Cet attribut est uniquement requis si l'emplacement d'insertion contient le mot node.

Valeur

La valeur *tagtype__Tag* est un nom spécifié par l'utilisateur pour le paramètre de nœud qui est transmis avec l'objet de paramètre à la fonction `dwscripts.applySB()`. Par exemple, si vous insérez du texte dans un formulaire, vous pouvez utiliser un paramètre *form__tag*. Dans votre fonction `applyServerBehavior()` de comportement de serveur, vous pouvez utiliser le paramètre *form__tag* pour indiquer le formulaire précis à mettre à jour, comme indiqué dans l'exemple suivant :

```
function applyServerBehavior(ssRec) {
  var paramObj = new Object();
  paramObj.rs = getRecordsetName();
  paramObj.form__tag = getFormNode();
  dwscripts.applySB(paramObj, sbObj);
}
```

Dans votre fichier EDML, vous pouvez spécifier le paramètre de nœud `form__tag`, comme indiqué dans l'exemple suivant :

```
<insertText location="lastChildOfNode" nodeParamName="form__tag">
  <![CDATA[<input type="hidden" name="MY_DATA">]]>
</insertText>
```

Le texte est inséré en tant que valeur `lastChildOfNode` et le nœud spécifique est transmis à l'aide de la propriété `form__tag` de l'objet de paramètre.

‹searchPatterns›

Description

Cette balise donne des indications sur la façon de rechercher le texte du participant dans le document et contient une liste de modèles utilisés lors de la recherche d'un participant. Si plusieurs modèles de recherche sont définis, ils doivent tous être trouvés dans le texte analysé (les modèles de recherche ont une relation AND logique), à moins qu'ils ne soient marqués comme facultatifs à l'aide de l'indicateur `isOptional`.

Parent

`implementation`

Type

Balise de bloc.

Obligatoire

Non.

Attributs de <searchPatterns>

Les éléments suivants sont des attributs valides de la balise searchPatterns.

whereToSearch

Description

Cet attribut indique où rechercher le texte du participant. Il est relatif à l'emplacement d'insertion, veillez donc à définir ces deux paramètres soigneusement (voir *emplacement*, page 366).

Parent

`searchPatterns`

Type

Attribut.

Obligatoire

Oui.

Valeurs

`directive`, *`tag+tagName`*, *`tag+*`*, *`comment`*, *`text`*

- La valeur *`directive`* recherche toutes les directives de serveur (balises spécifiques au serveur). Pour ASP et JSP, cela signifie une recherche dans tous les blocs de script `<% ... %>`.

  REMARQUE

  La recherche n'est pas effectuée dans les attributs de balise, même s'ils contiennent des directives.

- La valeur *`tag+tagName`* effectue une recherche dans le contenu de la balise donnée, comme indiqué dans l'exemple suivant :

  ```
  <searchPatterns whereToSearch="tag+FORM">
  ```

  Cet exemple n'effectue de recherche que dans les balises de formulaire. Par défaut, la recherche est effectuée dans tout le nœud `outerHTML`. Pour les balises `INPUT`, indiquez le type après une barre oblique (/). Dans l'exemple suivant, utilisez ce code pour effectuer une recherche dans tous les boutons d'envoi :

  ```
  <searchPatterns whereToSearch="tag+INPUT/SUBMIT">.
  ```

- La valeur *tag+** effectue une recherche dans le contenu de n'importe quelle balise, comme indiqué dans l'exemple suivant :

  ```
  <searchPatterns whereToSearch="tag+*">
  ```

  Cet exemple effectue une recherche dans toutes les balises.
- La valeur *comment* limite la recherche aux seuls commentaires HTML `<! ... >`, comme indiqué dans l'exemple suivant :

  ```
  <searchPatterns whereToSearch="comment">
  ```

  Cet exemple recherche des balises telles que `<!-- my comment here -->`.
- La valeur *text* limite la recherche aux sections de texte brut, comme indiqué dans l'exemple suivant :

  ```
  <searchPatterns whereToSearch="text">
    <searchPattern>XYZ</searchPattern>
  </searchPatterns>
  ```

  Cet exemple recherche un nœud de texte contenant le texte `XYZ`.

<searchPattern>

Description

Cette balise est un modèle qui identifie le texte participant et en extrait chaque valeur de paramètre. Chaque sous-expression de paramètre doit être mise entre parenthèses ().

Vous pouvez avoir des modèles sans paramètres (utilisés pour identifier le texte participant), des modèles avec un paramètre ou des modèles avec plusieurs paramètres. Tous les modèles obligatoires doivent être trouvés et chaque paramètre doit être nommé et trouvé exactement une fois.

Pour plus d'informations sur l'utilisation de la balise `searchPattern`, voir *Recherche des comportements de serveur*, page 387.

Parent

`searchPatterns`

Type

Balise de bloc.

Obligatoire

Oui.

Valeurs

`searchString`, */`regularExpression`/*, *`<empty>`*

- La valeur *searchString* est une simple chaîne de recherche, sensible à la casse. Elle ne peut pas être utilisée pour l'extraction de paramètres.
- La valeur */regularExpression/* est un modèle de recherche d'expression régulière.
- La valeur *<empty>* est utilisée si aucun modèle n'est spécifié. Dans ce cas, il est toujours considéré qu'il y a correspondance et la valeur entière est affectée au premier paramètre.

  Dans l'exemple suivant, pour identifier le texte participant `<%= RS1.Field.Items("author_id") %>`, vous pouvez définir un modèle simple, suivi d'un modèle précis extrayant également les deux valeurs de paramètre :

```
<searchPattern>Field.Items</searchPattern>
<searchPattern paramNames="rs,col">
  <![CDATA[
  /<%=\s*(\w+)\.Field\.Items\("(\w+)"\)/
  ]]>
</searchPattern>
```

  Dans cet exemple, une correspondance précise est établie avec le modèle. La valeur de la première sous-expression (`\w+`) est affectée au paramètre `rs` et celle de la deuxième sous-expression (`\w+`) au paramètre `col`.

REMARQUE Les expressions régulières doivent impérativement commencer et se terminer par une barre oblique (/). Sinon, elles sont utilisées en tant que recherches de chaîne littérale. Les expressions régulières peuvent être suivies du modificateur d'expression régulière `i` pour indiquer la non-sensibilité à la casse (comme dans `/pattern/i`). Par exemple, VBScript n'est pas sensible à la casse et doit donc utiliser `/pattern/i`. JavaScript est sensible à la casse et doit utiliser `/pattern/`.

  Si vous désirez affecter le contenu entier de l'emplacement de recherche limitée à un paramètre, n'indiquez aucun modèle, comme indiqué dans l'exemple suivant :

```
<searchPatterns whereToSearch="tag+OPTION">
  <searchPattern>MY_OPTION_NAME</searchPattern>
  <searchPattern paramNames="optionLabel" limitSearch="innerOnly">
  </searchPattern>
</searchPatterns>
```

  Cet exemple règle le paramètre `optionLabel` sur le contenu complet de `innerHTML` d'une balise `OPTION` .

Attributs de <searchPattern>

Les éléments suivants sont des attributs valides de la balise searchPattern.

paramNames

Description

Cet attribut est une liste séparée à l'aide de virgules des noms de paramètre dont les valeurs sont extraites. Ces paramètres sont affectés dans l'ordre de la sous-expression. Vous pouvez affecter des paramètres uniques ou utiliser une liste séparée par virgule pour affecter plusieurs paramètres. Si d'autres expressions entre parenthèses sont utilisées mais n'indiquent pas de paramètres, des virgules supplémentaires peuvent être utilisées en tant qu'espaces réservés dans la liste des noms de paramètres.

Les noms de paramètres doivent correspondre à ceux spécifiés dans le texte d'insertion et dans les paramètres de mise à jour.

Parent

`searchPattern`

Type

Attribut.

Obligatoire

Oui.

Valeurs

`paramName1`, *`paramName2`*, ...

Chaque nom de paramètre doit être le nom exact d'un paramètre utilisé dans le texte d'insertion. Par exemple, si le texte d'insertion contient `@@p1@@`, vous devez définir exactement un paramètre avec ce nom :

```
<searchPattern paramNames="p1">patterns</searchPattern>
```

Pour extraire plusieurs paramètres utilisant un seul modèle, utilisez une liste de noms de paramètre séparés par des virgules, dans l'ordre dans lequel les sous-expressions apparaissent dans le modèle. Supposons que l'exemple suivant indique votre modèle de recherche :

```
<searchPattern paramName="p1,,p2">/(\w+)_(BIG|SMALL)_(\w+)/¬
</searchPattern>
```

Il y a deux paramètres (séparés par du texte) à extraire. Avec le texte suivant : `<%= a_BIG_b %>`, la première sous-expression du modèle de recherche correspond à `"a"`, donc `p1="a"`. La deuxième sous-expression est ignorée (remarquez le `,,` dans la valeur `paramName`). La troisième sous-expression correspond à `"b"`, donc `p2="b"`.

limitSearch

Description

Cet attribut limite la recherche à une partie de la balise `whereToSearch`.

Parent

`searchPattern`

Type

Attribut.

Obligatoire

Non.

Valeurs

`all`, *`attribute+attribName`*, *`tagOnly`*, *`innerOnly`*

- La valeur *`all`* (valeur par défaut) effectue une recherche dans la balise entière qui est spécifiée dans l'attribut `whereToSearch`.
- La valeur *`attribute+attribName`* effectue la recherche dans la seule valeur de l'attribut spécifié, comme indiqué dans l'exemple suivant :

```
<searchPatterns whereToSearch="tag+FORM">
  <searchPattern limitSearch="attribute+ACTION">
    /MY_PATTERN/
  </searchPattern>
</searchPatterns>
```

  Cet exemple indique que la recherche doit se limiter à la seule valeur de l'attribut `ACTION` des balises `FORM`. Si cet attribut n'est pas défini, la balise est ignorée.
- La valeur *`tagOnly`* effectue la recherche dans la seule balise extérieure et ignore la balise `innerHTML`. Cette valeur est valide uniquement lorsque `whereToSearch` est une balise.
- La valeur *`innerOnly`* effectue la recherche dans la seule balise `innerHTML` et ignore la balise extérieure. Cette valeur est valide uniquement lorsque `whereToSearch` est une balise.

isOptional

Description

Cet attribut est un indicateur signifiant que le modèle de recherche n'est pas obligatoire pour trouver le participant. Cela est utile pour les participants plus complexes pouvant avoir des paramètres non essentiels à extraire. Vous pouvez créer des modèles servant à identifier de façon distincte un participant et avoir des modèles facultatifs pour l'extraction de paramètres non essentiels.

Parent

`searchPattern`

Type

Attribut.

Obligatoire

Non.

Valeurs

`true`, *`false`*

- La valeur est *`true`* si le `searchPattern` n'est pas indispensable à l'identification du participant.
- La valeur est *`false`* (valeur par défaut) si la balise `searchPattern` est requise.

Considérez par exemple la chaîne de jeux d'enregistrements simple suivante :

```
<%
var Recordset1 = Server.CreateObject("ADODB.Recordset");
Recordset1.ActiveConnection = "dsn=andescoffee;";
Recordset1.Source = "SELECT * FROM PressReleases";
Recordset1.CursorType = 3;
Recordset1.Open();
%>
```

Les modèles de recherche doivent identifier le participant et extraire plusieurs paramètres. Cependant, si un paramètre tel que *`cursorType`* n'est pas trouvé, il doit toujours être considéré comme un jeu d'enregistrements. Le paramètre *`cursor`* est facultatif. Dans l'EDML, les modèles de recherche peuvent ressembler à l'exemple suivant :

```
<searchPattern paramNames="rs">/var (\w+) = Server.CreateObject/
</searchPattern>
<searchPattern paramNames="src">/ActiveConnection = "([^\r\n]*)"/¬
</searchPattern>
<searchPattern paramNames="conn">/Source = "([^\r\n]*)"/¬
</searchPattern>
<searchPattern paramNames="cursor" isOptional="true">¬
```

```
/CursorType = (\d+)/
</searchPattern>
```

Les trois premiers modèles sont obligatoires pour identifier le jeu d'enregistrements. Si le dernier paramètre n'est pas trouvé, le jeu d'enregistrements est quand même identifié.

‹updatePatterns›

Description

Cette fonction facultative avancée permet des mises à jour précises du participant. Sans cette balise, le participant est automatiquement mis à jour par remplacement du texte participant entier à chaque fois. Si vous spécifiez une balise `<updatePatterns>`, elle doit contenir des modèles précis pour trouver et remplacer chaque paramètre du participant.

Cette balise peut s'avérer utile si l'utilisateur apporte des modifications au texte. Seules les parties du texte qui ont besoin d'être modifiées seront mises à jour avec précision.

Parent

`implementation`

Type

Balise de bloc.

Obligatoire

Non.

‹updatePattern›

Description

Cette balise est un type d'expression régulière spécifique permettant des mises à jour précises du texte du participant. Il doit y avoir au moins une définition de modèle de mise à jour pour chaque paramètre unique déclaré dans le texte d'insertion (de la forme `@@paramName@@`).

Parent

`updatePatterns`

Type

Balise de bloc.

Obligatoire

Oui (au moins une, si vous déclarez la balise `updatePatterns`).

Valeurs

La valeur est une expression régulière trouvant un paramètre placé entre deux sous-expressions entre parenthèses, dans le formulaire */(pre-pattern)parameter-pattern(post-pattern)/*. Vous devez définir au moins un modèle de mise à jour pour chaque `@@paramName@@` unique dans votre texte d'insertion. L'exemple suivant indique à quoi peut ressembler votre texte d'insertion :

```
<insertText location="afterSelection">
  <![CDATA[<%= @@rs@@.Field.Items("@@col@@") %>]]>
</insertText>
```

Une instance particulière du texte d'insertion sur une page peut ressembler à l'exemple suivant :

```
<%= RS1.Field.Items("author_id") %>
```

Il y a deux paramètres, `rs` et `col`. Pour mettre à jour ce texte une fois qu'il est inséré sur la page, vous avez besoin de deux définitions de modèle de mise à jour :

```
<updatePattern paramName="rs" >
  /(\b)\w+(\.Field\.Items)/
</updatePattern>
<updatePattern paramName="col">
  /(\bItems\(")\w+("\))/
</updatePattern>
```

Les parenthèses littérales, ainsi que d'autres caractères d'expressions régulières spéciaux, n'ont pas été pris en compte en raison de la barre oblique inverse les précédant (\). L'expression du milieu, définie comme `\w+`, est mise à jour avec la dernière valeur transmise pour les paramètres `rs` et `col`, respectivement. Les valeurs `RS1` et `author_id` peuvent être mises à jour avec de nouvelles valeurs de façon précise.

Plusieurs instances du même modèle peuvent être mises à jour en même temps en utilisant l'indicateur global d'expression régulière `g` après la barre oblique de fermeture (par exemple, dans */pattern/g*).

Si le texte du participant est long et complexe, vous devrez utiliser plusieurs modèles pour mettre à jour un seul paramètre, comme indiqué dans l'exemple suivant :

```
<% ...
  Recordset1.CursorType = 0;
  Recordset1.CursorLocation = 2;
  Recordset1.LockType = 3;
%>
```

Pour mettre à jour le nom du jeu d'enregistrements dans les trois positions, vous devez disposer de trois modèles de mise à jour pour un seul paramètre, comme indiqué dans l'exemple suivant :

```
<updatePattern paramName="rs">
```

```
  /(\b)\w+(\.CursorType)/
</updatePattern>
<updatePattern paramName="rs">
  /(\b)\w+(\.CursorLocation)/
</updatePattern>
<updatePattern paramName="rs">
  /(\b)\w+(\.LockType)/
</updatePattern>
```

Vous pouvez maintenant transmettre une nouvelle valeur pour le jeu d'enregistrements, qui est mis à jour avec précision à trois emplacements.

Attributs de <updatePattern>

Les éléments suivants sont des attributs valides de la balise updatePattern.

paramName

Description

Cet attribut indique le nom du paramètre dont la valeur est utilisée pour mettre à jour le participant. Ce paramètre doit correspondre à ceux spécifiés dans le texte d'insertion et dans les paramètres de recherche.

Parent

`updatePattern`

Type

Attribut.

Obligatoire

Oui.

Valeurs

La valeur est le nom exact d'un paramètre utilisé dans le texte d'insertion. Dans l'exemple suivant, si le texte d'insertion contient une valeur `@@rs@@`, vous devez avoir un paramètre du nom :

```
<updatePattern paramName="rs">pattern</updatePattern>
```

‹delete›

Description

Cette balise est une fonction facultative avancée qui vous permet de contrôler comment un participant est supprimé. Sans cette balise, vous devez effacer le participant entièrement pour le supprimer, mais seulement si aucun comportement de serveur n'y fait référence. En définissant une balise `<delete>`, vous pouvez spécifier qu'il ne doit jamais être supprimé ou que seules certaines parties doivent l'être.

Parent

`implementation`

Type

Balise.

Obligatoire

Non.

Attributs de ‹delete›

Les éléments suivants sont des attributs valides de la balise delete.

deleteType

Description

Cet attribut indique le type de suppression à effectuer. Il possède différentes significations, selon que le participant est une directive, une balise ou un attribut. Par défaut, le participant entier est supprimé.

Parent

`delete`

Type

Attribut.

Obligatoire

Non.

Valeurs

`all`, *`none`*, *`tagOnly`*, *`innerOnly`*, *`attribute+attribName`*, *`attribute+*`*

- La valeur *all* (valeur par défaut) supprime la directive ou la balise entière. Pour les attributs, elle supprime la définition entière.
- La valeur *none* n'est jamais automatiquement supprimée.
- La valeur *tagOnly* supprime uniquement la balise extérieure mais laisse le contenu de la balise innerHTML intact. Pour les attributs, elle supprime également la balise extérieure s'il s'agit d'une balise de bloc. Sans signification pour les directives.
- La valeur *innerOnly* supprime uniquement le contenu (la balise innerHTML) lorsqu'elle est appliquée à des balises. Pour les attributs, elle supprime uniquement la valeur. Sans signification pour les directives.
- La valeur *attribute+attribName* supprime uniquement l'attribut spécifié lorsqu'elle est appliquée à des balises. Sans signification pour les directives et les attributs.
- La valeur *attribute+** supprime tous les attributs des balises. Sans signification pour les directives et les attributs.

Si votre comportement de serveur convertit le texte sélectionné en lien, vous pouvez supprimer le lien en supprimant la balise extérieure uniquement, comme indiqué dans l'exemple suivant :

```
<delete deleteType="tagOnly"/>
```

Cet exemple convertit un participant de lien de `<A HREF="...">HELLO</A>` en `HELLO`.

‹translator›

Description

Cette balise donne des informations pour la traduction d'un participant de façon à ce qu'elle puisse être rendue différemment et puisse avoir un inspecteur de propriétés personnalisé.

Parent

`implementation`

Type

Balise de bloc.

Obligatoire

Non.

‹searchPatterns›

Description

Cette balise permet à Dreamweaver de trouver les instances spécifiées dans un document. Si plusieurs modèles de recherche sont définis, ils doivent tous être trouvés dans le texte analysé (les modèles de recherche ont une relation AND logique), à moins qu'ils ne soient marqués comme facultatifs à l'aide de l'indicateur `isOptional`.

Parent

`translator`

Type

Balise de bloc.

Obligatoire

Oui.

‹translations›

Description

Cette balise contient une liste d'instructions de traduction, chacune indiquant où rechercher le participant et ce qu'il convient d'en faire.

Parent

`translator`

Type

Balise de bloc.

Obligatoire

Non.

‹translation›

Description

Cette balise contient une instruction de traduction unique indiquant l'emplacement du participant, quel type de traduction effectuer et par quoi remplacer le texte du participant.

Parent

`translations`

Type

Balise de bloc.

Obligatoire

Non.

Attributs de ‹translation›

Les éléments suivants sont des attributs valides de la balise translation.

whereToSearch

Description

Cet attribut indique où rechercher le texte, par rapport à l'emplacement d'insertion. Veillez donc à définir soigneusement chaque emplacement (voir *emplacement*, page 366).

Parent

`translation`

Type

Attribut.

Obligatoire

Oui.

limitSearch

Description

Cet attribut limite la recherche à une partie de la balise `whereToSearch`.

Parent

`translation`

Type

Attribut.

Obligatoire

Non.

translationType

Description

Cet attribut indique le type de traduction à réaliser. Ces types sont prédéfinis et donnent à la traduction une fonctionnalité précise. Par exemple, si vous spécifiez `dynamic data`, toute donnée traduite doit se comporter comme une donnée dynamique Dreamweaver ; c'est-à-dire ressembler à un espace réservé de données dynamiques dans le mode de mise en page (notation entre accolades ({}) avec couleur d'arrière-plan dynamique) et figurer dans le panneau Comportements de serveur.

Parent

`translation`

Type

Attribut.

Obligatoire

Oui.

Valeurs

`dynamic data`, *`dynamic image`*, *`dynamic source`*, *`tabbed region start`*, *`tabbed region end`*, *`custom`*

- Cette valeur *`dynamic data`* indique que les directives traduites ressemblent à des données dynamiques Dreamweaver et se comportent de la même façon, comme indiqué dans l'exemple suivant :

```
<translation whereToSearch="tag+IMAGE"
  limitSearch="attribute+SRC"
  translationType="dynamic data">
```

- La valeur *`dynamic image`* indique que les attributs traduits doivent ressembler à des images dynamiques Dreamweaver et se comporter de la même façon, comme indiqué dans l'exemple suivant :

```
<translation whereToSearch="IMAGE+SRC"
  translationType="dynamic image">
```

- La valeur *`dynamic source`* indique que les directives traduites doivent se comporter comme des sources dynamiques Dreamweaver, comme indiqué dans l'exemple suivant :

```
<translation whereToSearch="directive"
  translationType="dynamic source">
```

- La valeur *`tabbed region start`* indique que les balises `<CFLOOP>` traduites définissent le début d'un contour tabulé, comme indiqué dans l'exemple suivant :

```
<translation whereToSearch="CFLOOP"
  translationType="tabbed region start">
```

- La valeur *tabbed region end* indique que les balises </CFLOOP> traduites définissent la fin d'un contour tabulé, comme indiqué dans l'exemple suivant :

```
<translation whereToSearch="CFLOOP"
  translationType="tabbed region end">
```

- La valeur *custom* est le cas par défaut, dans lequel aucune fonctionnalité Dreamweaver interne n'est ajoutée à la traduction. Il est courant de l'utiliser lors de la spécification d'une balise à insérer pour un inspecteur de propriétés personnalisé, comme indiqué dans l'exemple suivant :

```
<translation whereToSearch="directive"
  translationType="custom">
```

<openTag>

Description

Cette balise facultative peut être insérée au début de la section de traduction. Cette balise permet à d'autres extensions, par exemple les inspecteurs de propriétés personnalisés, de trouver la traduction.

Parent

translation

Type

Balise de bloc.

Obligatoire

Non.

Valeurs

La valeur *tagName* est un nom de balise valide. Il doit être unique pour empêcher les conflits avec les types de balise connus. Par exemple, si vous spécifiez <openTag>MM_DYNAMIC_CONTENT</openTag> les données dynamiques sont traduites dans la balise MM_DYNAMIC_CONTENT.

<attributs>

Description

Cette balise contient une liste d'attributs à ajouter à la balise traduite spécifiée dans `openTag`. Sinon, si la balise `openTag` n'est pas définie et que la balise `searchPattern` spécifie `tag`, cette balise contient une liste d'attributs traduits à ajouter à la balise trouvée.

Parent

`translation`

Type

Balise de bloc.

Obligatoire

Non.

<attribute>

Description

Cette balise spécifie un attribut unique (ou un attribut traduit) à ajouter à la balise traduite.

Parent

`attributs`

Type

Balise de bloc.

Obligatoire

Oui (au moins un).

Valeurs

La spécification *`attributeName="attributeValue"`* définit un attribut à une valeur. Le nom d'attribut est généralement fixe et la valeur contient des références de paramètres extraites par les modèles de paramètres, comme indiqué dans l'exemple suivant :

```
<attribute>SOURCE="@@rs@@"</attribute>
<attribute>BINDING="@@col@@"</attribute>
```

ou

```
<attribute>
  mmTranslatedValueDynValue="VALUE={@@rs@@.@@col@@}"
</attribute>
```

<display>

Description

Cette balise est une chaîne d'affichage facultative devant être insérée dans la traduction.

Parent

`translation`

Type

Balise de bloc.

Obligatoire

Non.

Valeurs

Cette valeur *`displayString`* se réfère à toute chaîne contenant du texte et du HTML. Peut comporter des références de paramètres extraites par les modèles de paramètres. Par exemple, `<display>{@@rs@@.@@col@@}</display>` entraîne une présentation de la traduction sous la forme `{myRecordset.myCol}`.

<closeTag>

Description

Cette balise facultative doit être insérée à la fin de la section traduite. Cette balise permet à certaines autres extensions, par exemple les inspecteurs de propriétés personnalisés, de trouver la traduction.

Parent

`translation`

Type

Balise de bloc.

Obligatoire

Non.

Valeurs

La valeur *`tagName`* est un nom de balise valide ; elle doit correspondre à une balise de traduction *`openTag`*.

Exemple

Si vous spécifiez la valeur `<closeTag>MM_DYNAMIC_CONTENT</closeTag>`, les données dynamiques sont traduites pour terminer avec la balise `</MM_DYNAMIC_CONTENT>`.

Techniques de comportements de serveur

Cette section explique les techniques courantes et avancées qui sont utilisées pour créer et modifier les comportements de serveur. La plupart des suggestions requièrent des paramètres spécifiques dans les fichiers EDML.

Recherche des comportements de serveur

Rédaction de modèles de recherche Pour que des comportements de serveur soient mis à jour ou supprimés, vous devez fournir à Dreamweaver les moyens de trouver chaque instance dans un document. Pour cela, une balise `quickSearch` et au moins une balise `searchPattern`, contenue dans la balise `searchPatterns`, sont requises.

La balise `quickSearch` tag doit être une chaîne et non une expression régulière, indiquant l'éventuelle existence du comportement de serveur sur la page. Elle ne respecte pas la casse. Elle doit être courte et unique et ne contenir aucun espace ni autre section susceptible d'être modifiée par l'utilisateur. L'exemple suivant montre un participant composé de la simple balise ASP JavaScript :

```
<% if (Recordset1.EOF) Response.Redirect("some_url_here") %>
```

Dans l'exemple suivant, la chaîne `quickSearch` vérifie la présence de la balise suivante :

```
<quickSearch>Response.Redirect</quickSearch>
```

Pour des raisons de performances, le processus de recherche des instances de comportement de serveur commence par le modèle `quickSearch`. Si cette chaîne figure dans le document et que le participant identifie un comportement de serveur (dans le fichier groupe, `partType="identifier"` pour ce participant), les fichiers associés sont alors chargés et la fonction `findServerBehaviors()` est appelée. A défaut de chaînes fiables à rechercher dans le participant (ou pour des raisons de débogage), vous pouvez laisser la chaîne `quickSearch` vide, comme indiqué dans l'exemple suivant :

```
<quickSearch></quickSearch>
```

Dans cet exemple, le comportement de serveur est toujours chargé et la recherche du document exécutée.

La balise `searchPattern` est ensuite utilisée pour analyser le document plus minutieusement que la balise `quickSearch` et pour extraire des valeurs de paramètre du code du participant. Les modèles de recherche spécifient l'emplacement de la recherche (l'attribut `whereToSearch`) et exploitent une série de balises `searchPattern` contenant des modèles spécifiques. Ces modèles peuvent utiliser de simples chaînes ou des expressions régulières. Le code de l'exemple précédent est une directive ASP, à savoir la spécification `whereToSearch="directive"` et une expression régulière identifie la directive et extrait les paramètres, comme indiqué dans l'exemple suivant :

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
  <searchPattern paramNames="rs,new__url">
    /if\s*\((\w+)\.EOF\)\s*Response\.Redirect\("([^\r\n]*)"\)/i
  </searchPattern>
</searchPatterns>
```

La chaîne recherchée est définie comme une expression régulière, commençant et finissant par une barre oblique (/) et suivie d'un `i` indiquant le non-respect de la casse. Dans l'expression régulière, les caractères spéciaux, tels que les parenthèses (), points (.), etc. sont spécifiés en les faisant précéder d'une barre oblique inverse (\). Les deux paramètres `rs` et `new__url` sont extraits de la chaîne à l'aide de sous-expressions entre parenthèses (les paramètres doivent être entre parenthèses). Dans cet exemple, ils sont indiqués par `(\w+)` et `([^\r\n]*)` : ces valeurs correspondent aux valeurs des expressions régulières normalement renvoyées par `$1` et `$2`.

Modèles de recherche facultatifs Il arrive parfois que vous vouliez identifier un participant, en dépit de paramètres demeurés introuvables. Un des participants peut éventuellement stocker des informations facultatives, telles qu'un numéro de téléphone. Pour cet exemple, vous pouvez utiliser le code ASP suivant :

```
<% //address block
  LNAME = "joe";
  FNAME = "smith";
  PHONE = "123-4567";
%>
```

Vous pouvez utiliser les modèles de recherche suivants :

```
<quickSearch>address</quickSearch>
<searchPatterns whereToSearch="directive">
  <searchPattern paramNames="lname">/LNAME\s*=\s*"([^\r\n]*)"/i¬
  </searchPattern>
  <searchPattern paramNames="fname">/FNAME\s*=\s*"([^\r\n]*)"/i¬
  </searchPattern>
<searchPattern paramNames="phone">/PHONE\s*=\s*"([^\r\n]*)"/i¬
</searchPattern>
</searchPatterns>
```

Dans l'exemple précédent, le numéro de téléphone doit être spécifié. Cependant, vous pouvez rendre le numéro de téléphone facultatif en ajoutant l'attribut `isOptional`, comme indiqué dans l'exemple suivant :

```
<quickSearch>address</quickSearch>
<searchPatterns whereToSearch="directive">
  <searchPattern paramNames="lname">/LNAME\s*=\s*"([^\r\n]*)"/i¬
  </searchPattern>
  <searchPattern paramNames="fname">/FNAME\s*=\s*"([^\r\n]*)"/i¬
  </searchPattern>
  <searchPattern paramNames="phone" isOptional="true">¬
  /PHONE\s*=\s*"([^\r\n]*)"/i
  </searchPattern>
</searchPatterns>
```

Le participant est maintenant reconnu, même si le numéro de téléphone n'est pas trouvé.

Méthode de correspondance des participants Si un comportement de serveur comporte plusieurs participants, ils doivent être identifiés dans le document utilisateur et mis en correspondance. Si l'utilisateur applique plusieurs instances du comportement de serveur à un document, chaque groupe de participants doit être mis en correspondance en conséquence. Pour que la mise en correspondance des participants s'effectue correctement, vous devrez peut-être modifier ou ajouter des paramètres, puis créer des participants qui soient identifiables de façon unique.

La mise en correspondance nécessite quelques règles : les participants sont mis en correspondance si l'ensemble des paramètres portant le même nom ont la même valeur. Au-dessus et en dessous de la balise `html` ne peut figurer qu'une seule instance d'un participant, avec un jeu donné de valeurs de paramètre. Entre les balises `html . . . /html`, les participants sont également mis en correspondance en fonction de leur position par rapport à la sélection ou aux nœuds communs utilisés pour l'insertion.

Les participants sans paramètres sont automatiquement retenus, comme le montre cet exemple d'un comportement de serveur avec un fichier groupe :

```
<group serverBehavior="test.htm">
  <title>Test</title>
  <groupParticipants>
    <groupParticipant name="test_p1" partType="identifier" />
    <groupParticipant name="test_p2" partType="identifier" />
  </groupParticipants>
</group>
```

L'exemple suivant insère deux participants simples au-dessus de la balise `html` :

```
<% //test_p1 %>
<% //test_p2 %>
<html>
```

Ces participants sont localisés et mis en correspondance, et le mot Test s'affiche une fois dans le panneau Comportements de serveur. Si vous ajoutez de nouveau ce comportement de serveur, rien n'est ajouté étant donné que les participants existent déjà.

Si les paramètres des participants sont uniques, de multiples instances peuvent être insérées au-dessus de la balise `html`. Par exemple, en ajoutant un paramètre de nom au participant, un utilisateur peut saisir un nom unique dans la boîte de dialogue du comportement de serveur Test. Si l'utilisateur saisit le nom « `aaa` », les participants suivants sont insérés :

```
<% //test_p1 name="aaa" %>
<% //test_p2 name="aaa" %>
<html>
```

Si vous ajoutez de nouveau le comportement de serveur et saisissez un autre nom, soit « `bbb` », le document se présente comme suit :

```
<% //test_p1 name="aaa" %>
<% //test_p2 name="aaa" %>
<% //test_p1 name="bbb" %>
<% //test_p2 name="bbb" %>
<html>
```

Deux instances du mot Test sont répertoriées dans le panneau Comportements de serveur. Si l'utilisateur tente d'ajouter une troisième instance à la page et qu'il la nomme « `aaa` », rien n'est ajouté étant donné qu'elle existe déjà.

Dans la balise `html`, le processus de mise en correspondance peut également avoir recours à des informations de position. Dans l'exemple suivant, nous avons deux participants, l'un ajouté avant la sélection et l'autre après, comme suit :

```
<% if (expression) { //mySBName %>
  Random HTML selection here
<% } //end mySBName %>
```

Ces deux participants ne sont pourvus d'aucun paramètre et sont donc regroupés. Il est toutefois possible d'ajouter une autre instance de ce comportement de serveur ailleurs dans le code HTML, comme indiqué dans l'exemple suivant :

```
<% if (expression) { //mySBName %>
  Random HTML selection here
<% } //end mySBName %>
  More HTML here...
<% if (expression) { //mySBName %>
  Another HTML selection here
<% } //end mySBName %>
```

A présent, les deux instances de chaque participant sont identiques, ce qui est permis dans le code HTML, et sont mises en correspondance selon l'ordre dans lequel elles apparaissent dans le document.

Voici un exemple type d'un problème de mise en correspondance et des solutions possibles pour l'éviter. Vous pouvez créer un participant capable de calculer la taxe de certaines données dynamiques et d'afficher le résultat à la sélection.

```
<% total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<html>
<body>
  The total (with taxes) is $<%=total%>
</body>
</html>
```

Les deux participants sont mis en correspondance car ils n'ont aucun paramètre en commun. Toutefois, si vous ajoutez une deuxième instance de ce comportement de serveur, vous devez avoir le code suivant :

```
<% total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<% total = Recordset1.Fields.Item("salePrice").Value * 1.0825 %>
<html>
<body>
  The total  (with taxes) is $<%=total%>
  Sale price (with taxes) is $<%=total%>
</body>
</html>
```

Ce comportement de serveur ne fonctionne plus correctement, car un seul paramètre est nommé`total`. Pour résoudre le problème, vérifiez qu'il existe bien un paramètre dont la valeur est unique pouvant être utilisé pour mettre en correspondance les participants. Dans l'exemple suivant, vous pouvez faire en sorte que le nom de la variable `total` soit unique en utilisant le nom de la colonne :

```
<% itemPrice_total = Recordset1.Fields.Item("itemPrice").¬
Value * 1.0825 %>
<% salePrice_total = Recordset1.Fields.Item("salePrice").¬
Value * 1.0825 %>
<html>
<body>
  The total  (with taxes) is $<%=itemPrice_total%>
  Sale price (with taxes) is $<%=salePrice_total%>
</body>
</html>
```

A présent, les modèles de recherche peuvent identifier de façon unique les participants et les mettre en correspondance.

Résolution du modèle de recherche

Dreamweaver prend en charge les actions suivantes en utilisant la fonctionnalité `searchPatterns` du participant :

- dépendance du transfert de fichier,
- mise à jour des chemins de fichier pour toutes les références de fichier (tels que les chemins des fichiers inclus).

Lors de la création de modèles de serveur, Dreamweaver génère des listes de modèles en recherchant les attributs `paramNames` spéciaux dans tous les participants. Pour rechercher des URL servant à vérifier la dépendance de fichiers et corriger le nom de chemin, Dreamweaver utilise les balises `searchPattern` qui contiennent des attributs `paramNames` avec l'extension `_url`. Vous pouvez spécifier plusieurs URL dans une même balise `searchPattern`.

Pour chaque balise `searchPattern` de traducteur comportant une valeur d'attribut `paramNames` avec l'extension `_includeUrl`, Dreamweaver MX utilise cette balise `searchPattern` pour traduire les instructions de fichier inclus de la page. Dreamweaver utilise un suffixe différent pour identifier les URL de fichier inclus, car toutes les références URL ne sont pas traduites. De même, une seule URL peut être traduite comme fichier inclus.

Pour résoudre une balise `searchPatterns`, Dreamweaver utilise l'algorithme suivant :

1. Recherche de l'attribut `whereToSearch` dans la balise `searchPatterns`.
2. Si la valeur de l'attribut commence par `tag+`, le reste de la chaîne est considéré comme le nom de la balise (les espaces ne sont pas acceptés dans les noms de balise).
3. Recherche de l'attribut `limitSearch` dans la balise `searchPattern`.
4. Si la valeur de l'attribut commence par `attribute+`, le reste de la chaîne est considéré comme le nom de l'attribut (les espaces ne sont pas acceptés dans les noms d'attribut).

Si ces quatre étapes se déroulent sans anomalie, Dreamweaver suppose qu'il s'agit d'une combinaison balise/attribut. Sinon, Dreamweaver recherche les balises `searchPattern` contenant un attribut `paramName` qui comporte un suffixe `_url` et une expression régulière définie. (Pour plus d'informations sur les expressions régulières, voir *Expressions régulières*, page 354.)

L'exemple suivant de balise `searchPatterns` ne contient pas de modèle de recherche car il combine une balise (`cfinclude`) avec un attribut (`template`) pour isoler l'URL et vérifier la dépendance de fichier, la correction de chemin, etc. :

```
<searchPatterns whereToSearch="tag+cfinclude">
   <searchPattern paramNames="include_url" limitSearch="attribute+template"
  />
</searchPatterns>
```

La combinaison balise/attribut (voir l'exemple précédent) ne s'applique pas à la traduction, car Dreamweaver traduit directement en texte dans le calque JavaScript. La vérification de dépendance de fichier, la correction de chemin, etc. s'effectuent dans le calque C. Dans le calque C, Dreamweaver partage le document en interne en directives (texte direct) et en balises (structurées en une arborescence efficace).

Mise à jour des comportements de serveur

Mise à jour des participants Par défaut, aucune balise `<updatePatterns>` ne figure dans les fichiers EDML participants et les instances du participant sont mises à jour dans le document par un remplacement complet. Lorsqu'un utilisateur modifie un comportement de serveur existant et clique sur OK, tout participant contenant un paramètre dont la valeur a changé sera supprimé, puis rajouté avec une nouvelle valeur au même endroit.

Si l'utilisateur personnalise le code du participant dans le document, le participant peut ne plus être reconnu si les modèles de recherche recherchent l'ancien code. Des modèles de recherche plus courts permettent à l'utilisateur de personnaliser le code du participant dans le document ; toutefois, la mise à jour de l'instance du comportement de serveur risque d'entraîner le remplacement du participant et la perte des modifications personnalisées.

Mise à jour de précision Dans certains cas, il peut s'avérer préférable de laisser l'utilisateur personnaliser le code du participant après insertion dans le document. Pour ce faire, il suffit de limiter les modèles de recherche et de fournir des modèles de mise à jour dans le fichier EDML. Après avoir ajouté le participant à la page, le comportement de serveur n'en modifie que certaines parties. L'exemple suivant indique un simple participant doté de deux paramètres :

```
<% if (Recordset1.EOF) Response.Redirect("some_url_here") %>
```

Dans cet exemple, les modèles de recherche peuvent se présenter comme suit :

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
  <searchPattern paramNames="rs,new__url">
    /if\s*\((\w+)\.EOF\)\s*Response\.Redirect\("([^\r\n]*)"\)/i
  </searchPattern>
</searchPatterns>
```

Le cas échéant, l'utilisateur peut ajouter un autre test à une instance particulière de ce code, comme indiqué dans l'exemple suivant :

```
<% if (Recordset1.EOF || x > 2) Response.Redirect("some_url_here") %>
```

Les modèles de recherche échouent, car ils recherchent une parenthèse après le paramètre `EOF`. Pour rendre les modèles de recherche plus flexibles, vous pouvez les raccourcir en les partageant, comme indiqué dans l'exemple suivant :

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
  <searchPattern paramNames="rs">/(\w+)\.EOF/</searchPattern>
  <searchPattern paramNames="new__url">
    /if\s*\([^\r\n]*\)\s*Response\.Redirect\("([^\r\n]*)"/i
  </searchPattern>
</searchPatterns>
```

Ces modèles de recherche raccourcis étant souples, l'utilisateur peut effectuer des ajouts au code. Toutefois, si le comportement de serveur modifie l'URL, lorsque l'utilisateur clique sur , le participant est remplacé et les personnalisations sont perdues. Pour une mise à jour plus précise, ajoutez une balise `updatePatterns` contenant un modèle pour la mise à jour de chaque paramètre :

```
<updatePatterns>
  <updatePattern paramNames="rs">/(\b)\w+(\.EOF)/¬
  </updatePattern>
  <updatePattern paramNames="new__url">
    /(Response\.Redirect\(")[^\r\n]*("/i
  </updatePattern>
</updatePatterns>
```

Dans les modèles de mise à jour, les parenthèses sont inversées et placées autour du texte avant et après le paramètre. Pour les modèles de recherche, utilisez le paramètre `textBeforeParam(param)textAfterParam`. Pour les modèles de mise à jour, utilisez le paramètre `(textBeforeParam)param(textAfterParam)`. Tout le texte qui se trouve entre les deux sous-expressions entre parenthèses sera remplacé par la nouvelle valeur du paramètre.

Suppression de comportements de serveur

Suppression par défaut et comptes de dépendance L'utilisateur peut supprimer une instance qu'il a sélectionnée dans le panneau Comportements de serveur en cliquant sur le bouton moins (-) ou en appuyant sur Suppr. Tous les participants sont éliminés, hormis ceux qui sont communs à d'autres comportements de serveur. En particulier, si plusieurs comportements de serveur ont un pointeur de participant relié au même nœud, le nœud n'est pas supprimé.

Par défaut, vous supprimez les participants en éliminant une balise entière. Si l'emplacement d'insertion correspond à `"wrapSelection"`, seule la balise extérieure est supprimée. La déclaration d'attribut dans son ensemble est supprimée. L'exemple suivant suppose un participant sur l'attribut `ACTION` d'une balise `form` :

```
<form action="<% my_participant %>">
```

Après suppression de l'attribut, seul `form` demeure.

Utilisation d'indicateurs pour limiter la suppression des participants Il s'avère parfois utile de limiter la façon dont les participants sont supprimés. Pour cela, il suffit d'ajouter une balise de suppression (delete) au fichier EDML. L'exemple suivant montre un participant correspondant à l'attribut `href` d'un lien :

```
<a href="<%=MY_URL%>">Link Text</a>
```

Lorsque ce participant d'attribut est supprimé, la balise devient `<a>Link Text</a>`, qui n'apparaît plus sous forme de lien dans Dreamweaver. Vous pouvez éventuellement ne supprimer que la valeur de l'attribut, en ajoutant la balise suivante au fichier EDML participant :

```
<delete deleteType="innerOnly"/>
```

Vous pouvez aussi éliminer la balise entière lorsque l'attribut est supprimé en entrant `<delete deleteType="tagOnly"/>`. Le texte résultant est `Link Text`.

Comment éviter les conflits dans les fichiers JavaScript contenant la directive de partage de mémoire

Si plusieurs fichiers HTML font référence à un fichier JavaScript particulier, Dreamweaver charge JavaScript dans un emplacement central à partir duquel les fichiers HTML peuvent partager la même source JavaScript. Ces fichiers contiennent la directive suivante :

```
//SHARE-IN-MEMORY=true
```

Si un fichier JavaScript contient la directive `SHARE-IN-MEMORY` et qu'un fichier HTML y fait référence (à l'aide de la balise `SCRIPT` avec l'attribut `SRC`), Dreamweaver le charge dans un emplacement centralisé afin que le code soit, par la suite, implicitement inclus dans tous les fichiers HTML.

REMARQUE

Les fichiers JavaScript chargés à cet emplacement centralisé partageant la mémoire, ils ne peuvent dupliquer aucune déclaration. Un conflit de noms risque de se produire s'il y a définition simultanée d'une même variable ou fonction à la fois dans votre fichier chargé dans la mémoire commune et dans un autre fichier JavaScript. Par conséquent, prenez en compte ces fichiers et leur convention d'attribution de nom lorsque vous rédigez des fichiers JavaScript.

CHAPITRE 16

Sources de données

16

Les fonctions API de sources de données de Macromedia Dreamweaver 8 vous permettent d'ajouter des sources de données qui s'affichent dans le menu plus (+) du panneau Liaisons. Pour plus d'informations, voir la fonction `dreamweaver.dbi.getDataSources()` dans le *Guide des API de Dreamweaver*.

Les fichiers de source de données sont stockés dans le dossier Configuration/DataSources/*nom_modèle_serveur*. A ce jour, Dreamweaver prend en charge les modèles de serveurs suivants : ASP.Net/C#, ASP.Net/VisualBasic, ASP/JavaScript, ASP/VBScript, Macromedia ColdFusion, JSP et PHP/MySQL. Le sous-dossier de chaque modèle contient des fichiers HTML et EDML associés aux sources de données de ce modèle de serveur.

Le tableau ci-dessous recense les fichiers utilisés pour créer une source de données.

Chemin	Fichier	Description
Configuration/DataSources/Nom_modèle_serveur	*Nom_source_données*.htm	Indique le nom de la source de données et l'emplacement où résident les fichiers JavaScript de prise en charge.
Configuration/DataSources/Nom_modèle_serveur	*Nom_source_données*.edml	Définit le code inséré par Dreamweaver dans le document pour désigner la valeur de la source de données.
Configuration/DataSources/Nom_modèle_serveur	*Nom_source_données*.js	Contient les fonctions JavaScript requises pour ajouter, insérer et supprimer le code approprié dans un document.

Fonctionnement des sources de données

Dans Dreamweaver, l'ajout de données dynamiques se fait à partir du panneau Liaisons. Les objets de données dynamiques affichés dans le menu plus (+) sont basés sur le modèle de serveur spécifié pour la page. Par exemple, les utilisateurs peuvent insérer des jeux d'enregistrements, des commandes, des variables de demande, des variables de session et des variables d'application pour les applications ASP.

Les étapes suivantes décrivent les processus mis en jeu lors de l'ajout de données dynamiques :

1. Lorsque vous cliquez sur le menu plus (+) du panneau Liaisons, un menu contextuel s'affiche.

   Pour déterminer le contenu du menu, Dreamweaver commence par rechercher un fichier DataSources.xml dans le même dossier que les sources de données (Configuration/DataSources/ASP_Js/DataSources.xml, par exemple). Le fichier DataSources.xml décrit le contenu du menu contextuel ; il contient les références des fichiers HTML devant figurer dans le menu contextuel.

   Dreamweaver recherche une balise de titre dans chaque fichier HTML référencé. Si le fichier contient une balise de titre, le contenu de celle-ci est affiché dans le menu. Si le fichier ne contient pas de balise de titre, c'est le nom du fichier qui apparaît dans le menu.

   Une fois qu'il a fini de lire le fichier DataSources.xml ou si ce fichier n'existe pas, Dreamweaver effectue une recherche dans le reste du dossier pour vérifier s'il contient d'autres éléments devant apparaître dans le menu. S'il s'avère que le dossier principal contient des fichiers qui ne figurent pas dans le menu, Dreamweaver les y ajoute. Si des sous-dossiers contiennent des fichiers qui ne figurent pas dans le menu, Dreamweaver crée un sous-menu et les y ajoute.
2. Lorsque l'utilisateur sélectionne un élément dans le menu plus (+), Dreamweaver appelle la fonction `addDynamicSource()` afin que le code de la source de données soit ajouté au document de l'utilisateur.
3. Dreamweaver passe en revue tous les fichiers du dossier spécifique au modèle de serveur utilisé, en appelant la fonction `findDynamicSources()` dans chaque fichier. Pour chaque valeur du tableau renvoyé, Dreamweaver appelle la fonction `generateDynamicSourceBindings()` dans le même fichier afin d'obtenir une nouvelle liste de tous les champs dans chaque source de données pour le document de l'utilisateur. Ces champs sont présentés à l'utilisateur sous la forme d'une commande d'arborescence dans la boîte de dialogue Données dynamiques ou Texte dynamique, ou encore dans le panneau Liaisons. L'arborescence des sources de données d'un document ASP peut se présenter comme dans l'exemple suivant :

   ```
   Recordset (Recordset1)
   ```

```
    ColumnOneInRecordset
    ColumnTwoInRecordset
  Recordset (Recordset2)
    ColumnOfRecordset
  Request
    NameOfRequestVariable
    NameOfAnotherRequestVariable
  Session
    NameOfSessionVariable
```

4. Si l'utilisateur clique deux fois sur le nom d'une source de données dans le panneau Liaisons pour la modifier, Dreamweaver appelle la fonction `editDynamicSource()` pour qu'elle traite les modifications de l'utilisateur au sein de l'arborescence.
5. Si l'utilisateur clique sur le bouton Moins (-), Dreamweaver extrait le nœud sélectionné dans l'arborescence et le transmet à la fonction `deleteDynamicSource()`, qui supprime le code ajouté précédemment à l'aide de la fonction `addDynamicSource()`. S'il n'est pas possible de supprimer la sélection en cours, la fonction renvoie un message d'erreur. Après le retour de la fonction `deleteDynamicSource()`, Dreamweaver actualise l'arborescence des sources de données en appelant les fonctions `findDynamicSources()` et `generateDynamicSourceBindings()`.
6. Si l'utilisateur sélectionne une source de données et clique sur OK dans la boîte de dialogue Données dynamiques ou Texte dynamique, ou s'il clique sur Insérer ou sur Lier dans le panneau Liaisons, Dreamweaver appelle la fonction `generateDynamicDataRef()`. La valeur retournée est insérée dans le document au niveau du point d'insertion.
7. Si l'utilisateur affiche la boîte de dialogue Données dynamiques ou Texte dynamique pour modifier un objet de données dynamiques existant, la sélection dans l'arborescence des sources de données doit être initialisée sur l'objet de données dynamiques. Pour initialiser la commande d'arborescence, Dreamweaver passe en revue tous les fichiers du dossier spécifique au modèle de serveur utilisé (le dossier Configuration/DataSources/ASP_Js, par exemple), en appelant la fonction `inspectDynamicDataRef()` implémentée dans chaque fichier.

   Dreamweaver appelle la fonction `inspectDynamicDataRef()` pour que l'objet de données dynamiques soit reconverti, à partir du code du document de l'utilisateur, en un élément de l'arborescence (cette procédure est en fait inversée par rapport à celle produite lorsque la fonction `generateDynamicDataRef()` est appelée). Si la fonction `inspectDynamicDataRef()` renvoie un tableau contenant deux éléments, Dreamweaver indique, à l'aide d'un repère visuel, l'élément de l'arborescence lié à la sélection en cours.

8. Chaque fois que l'utilisateur modifie la sélection, Dreamweaver appelle la fonction `inspectDynamicDataRef()` pour déterminer si la nouvelle sélection correspond à un texte dynamique ou à une balise comportant un attribut dynamique. S'il s'agit de texte dynamique, Dreamweaver affiche les liaisons de la sélection en cours dans le panneau Liaisons.
9. Vous pouvez, à partir du panneau Liaisons ou des boîtes de dialogue Données dynamiques ou Texte dynamique, modifier le format des données pour un objet de texte dynamique ou un attribut dynamique que l'utilisateur a déjà ajouté à la page. Lorsque le format est modifié, Dreamweaver appelle la fonction `generateDynamicDataRef()` afin d'obtenir la chaîne à insérer dans un document utilisateur. Il transmet ensuite cette chaîne à la fonction `formatDynamicDataRef()` (voir *formatDynamicDataRef()*, page 423). La chaîne renvoyée par la fonction `formatDynamicDataRef()` est insérée dans le document utilisateur.

Exemple simple de source de données

Cette extension ajoute une source de données personnalisée au panneau Liaisons pour les documents Macromedia ColdFusion. Les utilisateurs peuvent spécifier la variable de leur choix pour la source de données.

Cet exemple permet de créer une source de données intitulée MaSourceDeDonnées. Cette source de données comprend un fichier JavaScript MaSourceDeDonnées.js, un fichier MaSourceDeDonnées_DataRef.edml et les fichiers de commandes MaSourceDeDonnées_Variable requis pour générer une boîte de dialogue dans laquelle les utilisateurs entrent le nom d'une variable spécifique. L'exemple MaSourceDeDonnées est basé sur l'implémentation des sources de données Cookie Variable et URL Variable. Les fichiers associés à ces sources de données résident dans le dossier Configuration/DataSources/ColdFusion.

Vous créez cette source de données en exécutant les étapes suivantes :

- Création du fichier de définition de source de données
- Création du fichier EDML
- Création du fichier JavaScript qui implémente les fonctions API de sources de données
- Création des fichiers de commande de prise en charge pour les entrées utilisateur
- Test de la nouvelle source de données

Création du fichier de définition de source de données

Ce fichier indique à Dreamweaver le nom de la source de données tel qu'il s'affichera dans le menu plus (+) du panneau Liaisons. Il indique également à Dreamweaver où se trouvent les fichiers JavaScript de prise en charge de l'implémentation de sources de données.

Lorsqu'un utilisateur clique sur le menu plus (+) du panneau Liaisons, Dreamweaver effectue des recherches dans le dossier DataSources associé au modèle de serveur actuel pour regrouper l'ensemble des sources de données définies dans les fichiers HTML (HTM) du dossier. Pour rendre une nouvelle source de données accessible aux utilisateurs, vous devez créer un fichier de définition de source de données qui fournit le nom de la source de données à l'aide de la balise `TITLE` et l'emplacement des fichiers JavaScript de prise en charge à l'aide de la balise `SCRIPT`.

En outre, plusieurs fichiers de prise en charge sont nécessaires pour l'implémentation de cette source de données. En général, il n'est pas nécessaire d'utiliser les fonctions de ces fichiers de prise en charge, mais ces dernières se révèlent souvent utiles (voire nécessaires dans cet exemple). Par exemple, le fichier dwscriptsServer.js contient la fonction `dwscripts.stripCFOutputTags()` utilisée pour l'implémentation de cette source de données. A l'aide du fichier DataSourceClass.js, vous créez une instance de la classe DataSource à utiliser comme valeur de retour pour la fonction `findDynamicSources()`.

Pour créer le fichier de définition de source de données :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<HTML>
<HEAD>
<TITLE>MyDatasource</TITLE>
<SCRIPT SRC="../../Shared/Common/Scripts/dwscripts.js"></SCRIPT>
<SCRIPT SRC="../../Shared/Common/Scripts/dwscriptsServer.js"></SCRIPT>
<SCRIPT SRC="../../Shared/Common/Scripts/DataSourceClass.js"></SCRIPT>
<SCRIPT SRC="MyDatasource.js"></SCRIPT>
</HEAD>
<body></body>
</HTML>
```

3. Enregistrez le fichier sous le nom MaSourceDeDonnées.htm dans le dossier Configuration/DataSources/ColdFusion.

Création du fichier EDML

Le fichier EDML définit le code inséré par Dreamweaver dans le document pour désigner la valeur de la source de données. (Pour plus d'informations sur les fichiers EDML, voir Chapitre 15, “Comportements de serveur,” on page 335.) Lorsqu'un utilisateur ajoute une valeur particulière à un document à partir d'une source de données, Dreamweaver insère le code qui sera converti dans la valeur réelle lors de l'exécution. Le fichier EDML participant définit le code du document (pour plus d'informations, voir *Fichiers EDML Participant*, page 363).

Pour la variable MaSourceDeDonnées, Dreamweaver doit insérer le code ColdFusion `<cfoutput>#MonXML.variable#</cfoutput>`, où *variable* est la valeur que l'utilisateur attend de la source de données.

Pour créer le fichier EDML :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<participant>
   <quickSearch><![CDATA[#]]></quickSearch>
   <insertText
  location="replaceSelection"><![CDATA[<cfoutput>#MyDatasource.@@bindin
  gName@@#</cfoutput>]]></insertText>
   <searchPatterns whereToSearch="tag+cfoutput">
     <searchPattern paramNames="sourceName,bindingName"><![CDATA[/
  #(?:\s*\w+\s*\()?(MyDatasource)\.(\w+)\b[^#]*#/i]]></searchPattern>
   </searchPatterns>
</participant>
```

3. Enregistrez le fichier sous le nom MaSourceDeDonnées_DataRef.edml dans le dossier Configuration/DataSources/ColdFusion.

Création du fichier JavaScript qui implémente les fonctions API de sources de données

Après avoir défini le nom de la source de données, le nom des fichiers de script de prise en charge et le code du document de travail Dreamweaver, vous devez spécifier les fonctions JavaScript afin que Dreamweaver permette à l'utilisateur d'ajouter, d'insérer et de supprimer le code nécessaire dans un document.

En vous basant sur la construction de la source de données Cookie Variable, vous pouvez mettre en œuvre la source de données MonXML, comme illustré dans l'exemple ci-dessous. (La commande `MaSourceDeDonnées_Variable` utilisée dans la fonction `addDynamicSource()` est définie dans *Création des fichiers de commande de prise en charge pour les entrées utilisateur*, page 406.)

Pour créer le fichier JavaScript :

1. Créez un document vierge.
2. Entrez le code suivant :

```
//************** GLOBALS VARS *****************
var MyDatasource_FILENAME = "REQ_D.gif";
var DATASOURCELEAF_FILENAME = "DSL_D.gif";

//****************** API **********************
function addDynamicSource()
{
  MM.retVal = "";
  MM.MyDatasourceContents = "";
  dw.popupCommand("MyDatasource_Variable");
  if (MM.retVal == "OK")
  {
    var theResponse = MM.MyDatasourceContents;
    if (theResponse.length)
    {
      var siteURL = dw.getSiteRoot();
      if (siteURL.length)
      {
        dwscripts.addListValueToNote(siteURL, "MyDatasource",
  theResponse);
      }
      else
      {
        alert(MM.MSG_DefineSite);
      }
    }
    else
    {
    alert(MM.MSG_DefineMyDatasource);
    }
  }
}

function findDynamicSources()
{
  var retList = new Array();

  var siteURL = dw.getSiteRoot()
```

```
  if (siteURL.length)
  {
    var bindingsArray = dwscripts.getListValuesFromNote(siteURL,
  "MyDatasource");
    if (bindingsArray.length > 0)
    {

  // Vous créez ici une instance de la classe DataSource telle qu'elle
  est définie dans le
  // fichier DataSourceClass.js pour stocker les valeurs renvoyées.

      retList.push(new DataSource("MyDatasource",
                                  MyDatasource_FILENAME,
                                  false,
                                  "MyDatasource.htm"))
    }
  }

  return retList;
}

function generateDynamicSourceBindings(sourceName)
{
  var retVal = new Array();

  var siteURL = dw.getSiteRoot();

  // Pour le nom d'objet localisé...
  if (sourceName != "MyDatasource")
  {
    sourceName = "MyDatasource";
  }

  if (siteURL.length)
  {
    var bindingsArray = dwscripts.getListValuesFromNote(siteURL,
  sourceName);
    retVal = getDataSourceBindingList(bindingsArray,
                                      DATASOURCELEAF_FILENAME,
                                      true,
                                      "MyDatasource.htm");
  }

  return retVal;
}

function generateDynamicDataRef(sourceName, bindingName, dropObject)
{
  var paramObj = new Object();
```

```
  paramObj.bindingName = bindingName;
  var retStr = extPart.getInsertString("", "MyDatasource_DataRef",
  paramObj);

  // Les balises cfoutput doivent être supprimées si une balise CFOUTPUT
  est insérée
  // ou si les attributs d'une balise ColdFusion doivent être liés. Nous
  utilisons donc la fonction
  // dwscripts.canStripCfOutputTags() de dwscriptsServer.js

  if (dwscripts.canStripCfOutputTags(dropObject, true))
  {
    retStr = dwscripts.stripCFOutputTags(retStr, true);
  }

  return retStr;
}

function inspectDynamicDataRef(expression)
{
  var retArray = new Array();

  if(expression.length)
  {
    var params = extPart.findInString("MyDatasource_DataRef",
  expression);
    if (params)
    {
      retArray[0] = params.sourceName;
      retArray[1] = params.bindingName;
    }
  }

  return retArray;
}

function deleteDynamicSource(sourceName, bindingName)
{
  var siteURL = dw.getSiteRoot();

  if (siteURL.length)
  {
    //Pour le nom d'objet localisé
    if (sourceName != "MyDatasource")
    {
      sourceName = "MyDatasource";
    }

    dwscripts.deleteListValueFromNote(siteURL, sourceName, bindingName);
```

```
    }
}
```

3. Enregistrez le fichier sous le nom MaSourceDeDonnées.js dans le dossier Configuration/DataSources/ColdFusion.

Création des fichiers de commande de prise en charge pour les entrées utilisateur

La fonction `addDynamicSource()` contient la commande `dw.popupCommand("MyDatasrouce_Variable")`, qui ouvre une boîte de dialogue dans laquelle l'utilisateur entre un nom spécifique de variable. Vous devez cependant créer la boîte de dialogue pour la variable MaSourceDeDonnées.

Pour fournir une boîte de dialogue à l'utilisateur, vous devez créer un nouvel ensemble de fichiers de commandes : un fichier de définition de commande HTML et un fichier d'implémentation de commande JavaScript (pour plus d'informations sur les fichiers de commandes, voir *Fonctionnement des commandes*, page 177).

Le fichier de définition de commande indique à Dreamweaver l'emplacement des fichiers JavaScript de prise en charge d'implémentation ainsi que la forme de la boîte de dialogue visible par l'utilisateur. Le fichier JavaScript de prise en charge détermine les boutons de la boîte de dialogue et indique comment affecter les données saisies par l'utilisateur à partir de la boîte de dialogue.

Pour créer le fichier de définition de commande :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension layout-engine 5.0//
  dialog">
<html>
<head>
<title>MyDatasource Variable</title>
<script src="MyDatasource_Variable.js"></script>
<SCRIPT SRC="../Shared/MM/Scripts/CMN/displayHelp.js"></SCRIPT>
<SCRIPT SRC="../Shared/MM/Scripts/CMN/string.js"></SCRIPT>
<link href="../fields.css" rel="stylesheet" type="text/css">
</head>
<body>
<form>
  <div ALIGN="center">
    <table border="0" cellpadding="2" cellspacing="4">
      <tr>
        <td align="right" valign="baseline" nowrap>Name:</td>
        <td valign="baseline" nowrap>
```

```
          <input name="theName" type="text" class="medTField">
        </td>
      </tr>
    </table>
  </div>
</form>
</body>
</html>
```

3. Enregistrez le fichier dans le dossier Configuration/Commands sous le nom MaSourceDeDonnées_Variable.htm.

REMARQUE	Le fichier MaSourceDeDonnées.js correspond au fichier de mise en œuvre créé dans le cadre de la procédure suivante.

Pour créer le fichier JavaScript de prise en charge :

1. Créez un document vierge.
2. Entrez le code suivant :

```
//******************* API **********************

function commandButtons(){
  return new
  Array(MM.BTN_OK,"okClicked()",MM.BTN_Cancel,"window.close()");
}

//***************** LOCAL FUNCTIONS  ******************

function okClicked(){
  var nameObj = document.forms[0].theName;

  if (nameObj.value) {
    if (IsValidVarName(nameObj.value)) {
      MM.MyDatasourceContents = nameObj.value;
      MM.retVal = "OK";
      window.close();
    } else {
      alert(nameObj.value + " " + MM.MSG_InvalidParamName);
    }
  } else {
    alert(MM.MSG_NoName);
  }
}
```

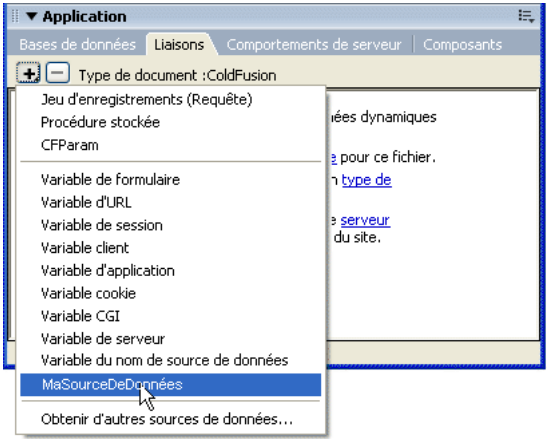
3. Enregistrez le fichier dans le dossier Configuration/Commands sous le nom MaSourceDeDonnées_Variable.js.

Test de la nouvelle source de données

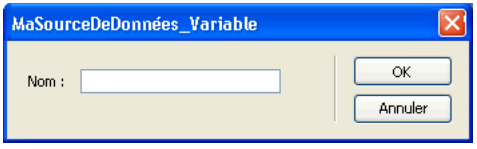
Vous pouvez maintenant ouvrir Dreamweaver (ou le redémarrer s'il est déjà utilisé) et ouvrir un fichier ColdFusion ou en créer un nouveau.

Procédez comme suit pour tester votre nouvelle source de données :

1. Placez le pointeur sur le document, cliquez sur le menu Plus (+) du panneau Liaisons pour afficher l'ensemble des sources de données disponibles. MaSourceDeDonnées doit s'afficher en fin de liste :

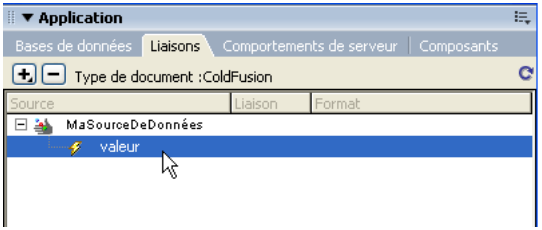


2. Cliquez sur l'option de source de données MaSourceDeDonnées pour afficher la boîte de dialogue MaSourceDeDonnées_Variable créée :

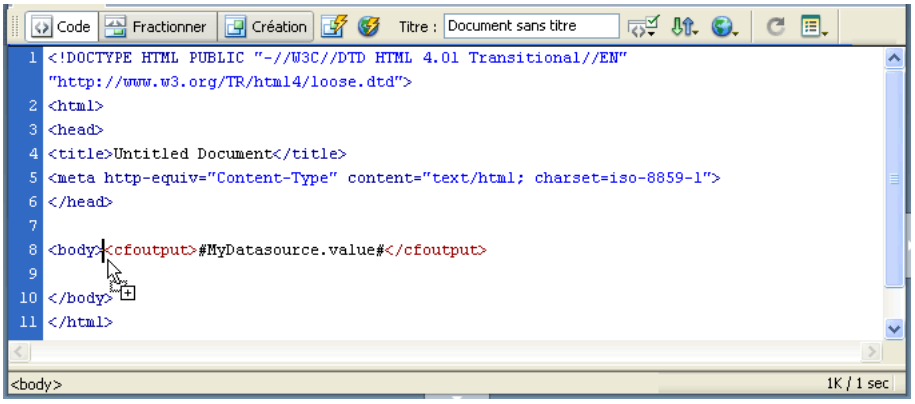


3. Saisissez une valeur dans la boîte de dialogue, puis appuyez sur OK.

Le panneau Liaisons affiche la source de données dans une arborescence comprenant la variable de la boîte de dialogue sous le nom de la source de données :



4. Faites glisser la variable dans votre document ; Dreamweaver se charge d'insérer le code approprié à partir du fichier EDML :



API des sources de données

Les fonctions de l'API de sources de données vous permettent de trouver, ajouter, modifier et supprimer des sources de données ainsi que de générer et inspecter des objets de données dynamiques.

addDynamicSource()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction ajoute une source de données dynamiques. Comme cette fonction est implémentée dans chaque fichier de source de données, Dreamweaver appelle l'implémentation appropriée de la fonction `addDynamicSource()` dès qu'une source de données est sélectionnée dans le menu plus (+).

Par exemple, pour les jeux d'enregistrements ou les commandes, Dreamweaver appelle la fonction `dw.serverBehaviorInspector.popupServerBehavior()`, qui insère un nouveau comportement de serveur dans le document. Pour les variables de demande, de session et d'application, Dreamweaver affiche une boîte de dialogue HTML/JavaScript permettant de recueillir le nom de la variable ; le comportement stocke ce nom en vue d'une utilisation ultérieure.

Une fois la fonction `addDynamicSource()` renvoyée, Dreamweaver efface le contenu de l'arborescence de la source de données et appelle les fonctions `findDynamicSources()` et `generateDynamicSourceBindings()` pour remplir de nouveau l'arborescence.

Valeurs renvoyées

Dreamweaver n'attend rien.

deleteDynamicSource()

Disponibilité

Dreamweaver UltraDev 1.

Description

Dreamweaver appelle cette fonction lorsqu'un utilisateur sélectionne une source de données dans l'arborescence avant de cliquer sur le bouton Moins (-).

Par exemple, dans Dreamweaver, si un jeu d'enregistrements ou une commande sont sélectionnés, la fonction `deleteDynamicSource()` appelle la fonction `dw.serverBehaviorInspector.deleteServerBehavior()`. Si la sélection est une variable de requête, de session ou d'application, la fonction se souvient que la variable a été supprimée et ne l'affiche plus. Une fois la fonction `deleteDynamicSource()` renvoyée, Dreamweaver efface le contenu de l'arborescence des sources de données et appelle les fonctions `findDynamicSources()` et `generateDynamicSourceBindings()` pour obtenir une nouvelle liste de toutes les sources de données associées au document de l'utilisateur.

Arguments

`sourceName`, *`bindingName`*

- L'argument *sourceName* est le nom du nœud de niveau supérieur auquel le nœud enfant est associé.
- L'argument *bindingName* est le nom du nœud enfant.

Valeurs renvoyées

Dreamweaver n'attend rien.

displayHelp()

Description

Si cette fonction est définie, un bouton Aide s'affiche sous les boutons OK et Annuler dans la boîte de dialogue. Cette fonction est appelée lorsque l'utilisateur clique sur le bouton Aide.

Arguments

Néant

Valeurs renvoyées

Dreamweaver n'attend rien.

Exemple

```
// L'instance suivante de la fonction displayHelp() ouvre
// un fichier (dans un navigateur) qui explique comment utiliser
// l'extension.
function displayHelp(){
  var myHelpFile = dw.getConfigurationPath() +
    '/ExtensionsHelp/superDuperHelp.htm';
  dw.browseDocument(myHelpFile);
}
```

editDynamicSource()

Disponibilité

Dreamweaver MX.

Description

Cette fonction est appelée lorsque l'utilisateur clique deux fois sur le nom d'une source de données dans le panneau Liaisons pour la modifier. Implémenter cette fonction vous permet de traiter les modifications utilisateur au sein de l'arborescence. Sinon, le comportement de serveur correspondant à la source de données est automatiquement appelé. Le développeur d'extensions peut utiliser cette fonction pour remplacer l'implémentation par défaut des comportements de serveur et créer un gestionnaire personnalisé.

Arguments

sourceName, *bindingName*

- L'argument *sourceName* est le nom du nœud de niveau supérieur auquel le nœud enfant est associé.
- L'argument *bindingName* est le nom du nœud enfant.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne : `true` si la fonction a traité la modification ; `false` dans le cas contraire.

findDynamicSources()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction renvoie les nœuds de niveau supérieur de l'arborescence des sources de données qui s'affichent dans la boîte de dialogue Données dynamiques ou Texte dynamique, ou encore dans le panneau Liaisons. La fonction `findDynamicSources()` est implémentée dans chaque fichier de source de données. En actualisant l'arborescence, Dreamweaver parcourt tous les fichiers du dossier DataSources et appelle la fonction `findDynamicSources()` dans chacun d'eux.

Valeurs renvoyées

Dreamweaver attend un tableau d'objets JavaScript dans lequel chaque objet peut comporter jusque cinq propriétés, décrites dans la liste suivante :

- La propriété `title` correspond au libellé qui apparaît à droite de l'icône de chaque nœud parent. La propriété `title` est toujours obligatoire.
- La propriété `imageFile` est le chemin d'un fichier contenant une icône (image GIF) représentant le nœud parent dans la commande d'arborescence affichée dans la boîte de dialogue Données dynamiques ou Texte dynamique, ou encore dans le panneau Liaisons. Cette propriété est obligatoire
- La propriété `allowDelete` est facultative. Si cette propriété est réglée sur `false`, lorsque l'utilisateur clique sur ce nœud dans le panneau Liaisons, le bouton moins (-) est désactivé. Si elle est réglée sur `true`, le bouton moins (-) est activé. Si la propriété n'est pas définie, elle prend par défaut la valeur `true`.

- La propriété `dataSource` est le nom du fichier dans lequel la fonction `findDynamicSources()` est définie. Par exemple, la fonction `findDynamicSources()` du fichier Session.htm, situé dans le dossier Configuration/DataSources/ASP_Js, règle la propriété `dataSource` sur session.htm. Cette propriété est obligatoire.
- La propriété `name` est le nom du comportement de serveur associé à la source de données, s'il existe. Certaines sources de données, telles que des jeux d'enregistrements, sont associées à des comportements de serveur. Lorsque vous créez un jeu d'enregistrements et que vous lui attribuez le nom `rsAuthors`, l'attribut `name` doit avoir pour valeur `rsAuthors`. La propriété `name` est toujours définie, mais il peut s'agir d'une chaîne vide (`""`) si aucun comportement de serveur n'est associé à la source de données (tel qu'une variable de session).

REMARQUE	Une classe JavaScript définissant ces propriétés existe dans le fichier DataSourceClass.js. Ce dernier se trouve dans le dossier Configuration/Shared/Common/Scripts.

generateDynamicDataRef()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction génère l'objet de données dynamiques pour un nœud enfant.

Arguments

`sourceName`, *`bindingName`*

- L'argument *`sourceName`* est le nom du nœud de niveau supérieur associé au nœud enfant.
- L'argument *`bindingName`* est le nom du nœud enfant à partir duquel vous souhaitez générer un objet de données dynamiques.

Valeurs renvoyées

Dreamweaver attend une chaîne qui peut être transmise pour formatage à la fonction `formatDynamicDataRef()` avant d'être insérée dans le document d'un utilisateur.

generateDynamicSourceBindings()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction renvoie le nœud enfant d'un nœud de niveau supérieur.

Arguments

sourceName

- L'argument *sourceName* est le nom du nœud de niveau supérieur dont vous voulez renvoyer les nœuds enfants.

Valeurs renvoyées

Dreamweaver attend un tableau d'objets JavaScript dans lequel chaque objet peut comporter jusque quatre propriétés, décrites dans la liste suivante :

- La propriété `title` correspond à l'étiquette qui apparaît à droite de l'icône de chaque nœud parent. Cette propriété est obligatoire
- La propriété `allowDelete` est facultative. Si cette propriété est réglée sur la valeur `false`, lorsque l'utilisateur clique sur ce nœud dans le panneau Liaisons, le bouton moins (-) apparaît désactivé. Si elle est réglée sur la valeur `true`, le bouton moins (-) est activé. Si la propriété n'est pas définie, elle prend par défaut la valeur `true`.
- La propriété `dataSource` est le nom du fichier dans lequel la fonction `findDynamicSources()` est définie. Par exemple, la fonction `findDynamicSources()` dans le fichier Session.htm, situé dans le dossier Configuration/DataSources/ASP_Js, définit la propriété `dataSource` sur `session.htm`. Cette propriété est obligatoire
- La propriété `name` est le nom du comportement de serveur associé à la source de données, s'il existe. Cette propriété est obligatoire. Certaines sources de données, telles que des jeux d'enregistrements, sont associées à des comportements de serveur. Lorsque vous créez un jeu d'enregistrements et que vous lui attribuez le nom `rsAuthors`, la propriété `name` doit avoir pour valeur `rsAuthors`. D'autres sources de données, telles que les variables de session, ne sont pas associées à un comportement de serveur. Leur propriété name doit correspondre à une chaîne vide (`""`).

REMARQUE	Une classe JavaScript définissant ces propriétés existe dans le fichier DataSourceClass.js. Ce dernier se trouve dans le dossier Configuration/Shared/Common/Scripts.

inspectDynamicDataRef()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction détermine le nœud correspondant dans l'arborescence de sources de données depuis un objet de données dynamiques. La fonction `inspectDynamicDataRef()` extrait la chaîne transmise par Dreamweaver et la compare à celle renvoyée par `generateDynamicDataRef()` pour chaque nœud de l'arborescence. En cas de correspondance, la fonction `inspectDynamicDataRef()` indique quel nœud de l'arborescence correspond à la chaîne transmise. La fonction identifie le nœud au moyen d'un tableau contenant deux éléments. Le premier élément est le nom du nœud parent ; le second est le nom du nœud enfant. Si aucune correspondance n'est trouvée, la fonction `inspectDynamicDataRef()` renvoie un tableau vide.

Chaque implémentation de la fonction `inspectDynamicDataRef()` ne recherche que les correspondances de son propre type d'objet. Par exemple, l'implémentation de la fonction `inspectDynamicDataRef()` pour les jeux d'enregistrements ne trouve une correspondance que si la chaîne transmise correspond à un nœud de jeu d'enregistrements dans l'arborescence.

Arguments

`string`

- L'argument *`string`* est l'objet de données dynamiques.

Valeurs renvoyées

Dreamweaver attend un tableau de deux éléments (nom du parent et nom de l'enfant) pour le nœud correspondant ; il renvoie la valeur `null` en l'absence de correspondance.

CHAPITRE 17

Formats de serveur

17

Le Chapitre 16, “Sources de données,” on page 397 explique comment Macromedia Dreamweaver 8 insère des données dynamiques dans le document de l'utilisateur en ajoutant une expression de serveur à l'endroit approprié. Lorsqu'un visiteur demande le document de l'utilisateur à partir du serveur Web, l'expression est convertie en une valeur de base de données, en contenu d'une variable de requête ou en toute autre valeur dynamique. Les formats de serveur de Dreamweaver permettent de formater la valeur dynamique pour la présenter au visiteur.

Ce chapitre traite de l'API de formatage des données dynamiques renvoyées par les fonctions décrites dans Chapitre 16, “Sources de données,” on page 397. Les données dynamiques sont formatées à l'aide d'une combinaison des fonctions décrites dans les deux chapitres. Si l'utilisateur sélectionne un format pour les données dynamiques, Dreamweaver appelle la fonction de source de données `generateDynamicDataRef()`, voir *generateDynamicDataRef()*, page 413 pour obtenir la chaîne à insérer dans le document utilisateur. Avant de l'insérer dans le document de l'utilisateur, Dreamweaver transmet cette chaîne à la fonction `formatDynamicDataRef()`, décrite dans ce chapitre. La chaîne renvoyée par la fonction `formatDynamicDataRef()` représente les données dynamiques formatées finalement insérées dans le document de l'utilisateur.

Les utilisateurs de Dreamweaver peuvent formater les données à l'aide de formats prédéfinis ou créer des formats à partir de types prédéfinis ou de types qu'ils ont créés eux-mêmes.

L'utilisateur peut formater les données dynamiques de plusieurs façons. A partir des boîtes de dialogue Données dynamiques ou Texte dynamique ou du panneau Liaisons, l'utilisateur peut utiliser le menu Format pour formater les données avant de les insérer dans un document HTML. S'il veut créer un format, l'utilisateur peut sélectionner la commande Modifier la liste de formats depuis le menu Format, puis sélectionner ensuite un type de format dans le menu Plus (+). Le menu Plus (+) contient une liste de types de format. Les types de format sont des catégories de format de base (Devise, Date/Heure, Casse, etc.). Ils regroupent tous les paramètres communs à une catégorie et facilitent la création de nouveaux formats.

Il serait par exemple possible de créer un format de devise. Le formatage des devises consiste en fait à convertir un nombre en une chaîne et à insérer des virgules et un symbole de devise (le signe du dollar, $, par exemple). Le type de données de format Devise regroupe tous les paramètres communs et vous demande une valeur pour chacun d'eux.

Fonctionnement du formatage de données

Tous les fichiers de format résident dans le dossier Configuration/ServerFormats/*Modèle_serveur_actuel* Chaque sous-dossier contient un fichier XML et plusieurs fichiers HTML.

Le fichier Formats.xml décrit tous les choix du menu Format. Dreamweaver ajoute automatiquement les options Modifier la liste de formats et Aucun.

Le dossier contient également un fichier HTML pour chaque type de format installé, dont `Casse d'alphabet`, `Devise`, `Date/Heure`, `Math`, `Nombre`, `Pourcentage`, `Simple` et `Rogner`.

Fichier Formats.xml

Le fichier Formats.xml contient une balise `format` pour chaque élément du menu Format. Chaque balise `format` contient les attributs obligatoires suivants :

- L'attribut `file=`*`fileName`* représente le fichier HTML du type de format concerné, tel que « `Devise` ».
- L'attribut `title=`*`string`* représente la chaîne qui apparaît dans le menu Format, telle que « `Devise - valeur par défaut` ».
- L'attribut `expression=`*`regexp`* représente une expression régulière correspondant aux objets de données dynamiques qui utilisent ce format. L'expression détermine le format actuellement appliqué à l'objet de données dynamiques. Par exemple, l'expression correspondant au format « `Devise - valeur par défaut` » serait `"<%\s*=\s*FormatCurrency\(.*, -1, -2, -2, -2\)\s*%>|<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>"`. La valeur de l'attribut d'expression doit être unique parmi toutes les balises `format` du fichier. Elle doit être suffisamment précise pour garantir que seules les instances de ce format correspondent à l'expression.
- L'attribut `visibility=`*`[hidden | visible]`* indique si la valeur est affichée dans le menu Format. Si la valeur de l'attribut `visibility` est `hidden` (masqué), le format ne s'affiche pas dans le menu Format.

La balise `format` peut éventuellement contenir des attributs supplémentaires ayant un nom arbitraire.

Certaines fonctions de formatage de données nécessitent un argument, *`format`*, qui est un objet JavaScript. Cet objet est le nœud correspondant à la balise `format` dans le fichier Formats.xml. L'objet a une propriété JavaScript pour chaque attribut de la balise `format` correspondante.

L'exemple suivant illustre la balise `format` associée à la chaîne « `Devise - valeur par défaut` » :

```
<format file="Currency" title="Currency - default" ¬
expression="<%\s*=\s*FormatCurrency\(.*, -1, -2, -2, -2\)\s*%>|¬
<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>"
NumDigitsAfterDecimal=-1 IncludeLeadingDigit=-2 ¬
UseParensForNegativeNumbers=-2 GroupDigits=-2/>
```

Le type de ce format correspond à `Devise`. La chaîne « `Devise - valeur par défaut` » s'affiche dans le menu Format. L'expression `<%\s*=\s*FormatCurrency\(.*, -1, -2, -2,¬ -2\)\s*%>|<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>` trouve des occurrences de ce format dans le document de l'utilisateur.

Les paramètres `NumDigitsAfterDecimal`, `IncludeLeadingDigit`, `UseParensForNegativeNumbers` et `GroupDigits` concernent le format `Devise` et ne sont pas obligatoires. Ces paramètres s'affichent dans la boîte de dialogue Paramètres du type de format `Devise`. Cette boîte de dialogue apparaît chaque fois qu'un utilisateur choisit le type de format `Devise` dans le menu Plus (+) de la boîte de dialogue Modifier la liste de formats. Les valeurs indiquées pour ces paramètres sont utilisées pour définir le nouveau format.

Le menu Plus (+) de la boîte de dialogue Modifier la liste de formats

Si vous ne voulez pas qu'un fichier du dossier ServerFormats apparaisse dans le menu Plus (+) de la boîte de dialogue Modifier la liste de formats, ajoutez l'instruction suivante de manière à ce qu'elle figure sur la première ligne du fichier HTML :

```
<!-- MENU-LOCATION=NONE -->
```

Pour déterminer le contenu du menu, Dreamweaver commence par rechercher un fichier ServerFormats.xml dans le même dossier que les formats de données (par exemple, \Configuration\ServerFormats\ASP\ServerFormats.xml). Le fichier ServerFormats.xml décrit le contenu du menu Plus (+) de la boîte de dialogue Modifier la liste de formats. Il contient des références aux fichiers HTML répertoriés dans le menu.

Dreamweaver recherche une balise de titre dans chaque fichier HTML référencé. Si le fichier contient une balise de titre, le contenu de celle-ci est affiché dans le menu. Si le fichier ne contient pas de balise de titre, c'est le nom du fichier qui apparaît dans le menu.

Une fois qu'il a fini de rechercher le fichier ou si ce fichier n'existe pas, Dreamweaver analyse le reste du dossier pour vérifier s'il contient d'autres éléments devant apparaître dans le menu. S'il s'avère que le dossier principal contient des fichiers qui ne figurent pas dans le menu, Dreamweaver les y ajoute. Si des sous-dossiers contiennent des fichiers qui ne figurent pas déjà dans le menu, Dreamweaver crée un sous-menu et les y ajoute.

Mise en service des fonctions de formatage de données

Les fonctions de formatage de données sont appelées dans les cas suivants :

- Dans la boîte de dialogue Données dynamiques ou Texte dynamique, l'utilisateur choisit un nœud dans l'arborescence des sources de données et un format dans le menu Format. Une fois le format choisi, Dreamweaver appelle la fonction `generateDynamicDataRef()` et transmet la valeur de renvoi de la fonction `generateDynamicDataRef()` à la fonction `formatDynamicDataRef()`. La valeur renvoyée par la fonction `formatDynamicDataRef()` s'affiche dans le paramètre Code de la boîte de dialogue. Lorsque l'utilisateur clique sur OK, la chaîne de code est insérée dans le document de l'utilisateur. Dreamweaver appelle ensuite la fonction `applyFormat()` pour insérer une déclaration de fonction. Pour plus d'informations, consultez la section *generateDynamicDataRef()*, page 413. Un processus similaire a lieu lorsque l'utilisateur utilise le panneau Liaisons.
- L'utilisateur change le format ou efface l'élément de données dynamiques. La fonction `deleteFormat()` est alors appelée. Cette fonction `deleteFormat()` supprime les scripts correspondants du document.
- Lorsque l'utilisateur clique sur le bouton plus (+) de la boîte de dialogue Modifier la liste de formats, Dreamweaver affiche un menu contenant tous les types de format pour le modèle de serveur utilisé. Chaque type de format correspond à un fichier du dossier Configuration\ServerFormats*Modèle_serveur_actuel*.

  Si l'utilisateur choisit, dans le menu Plus (+), un format qui nécessite un paramètre spécifié par l'utilisateur, Dreamweaver exécute le gestionnaire `onLoad` sur la balise `body`, puis ouvre la boîte de dialogue Paramètres, qui affiche les paramètres de ce type de format. Dans cette boîte de dialogue, l'utilisateur choisit les paramètres du format et clique sur OK. Dreamweaver appelle alors la fonction `applyFormatDefinition()`.

Si le format sélectionné n'a pas besoin d'afficher de boîte de dialogue Paramètres, Dreamweaver appelle la fonction `applyFormatDefinition()` lorsque l'utilisateur choisit le type de format dans le menu Plus (+).

- Par la suite, si l'utilisateur modifie le format (en le sélectionnant dans la boîte de dialogue Modifier la liste de formats et en cliquant sur le bouton Modifier), Dreamweaver appelle la fonction `inspectFormatDefinition()` avant d'afficher la boîte de dialogue Paramètres, de façon à ce que les commandes de formulaire puissent être initialisées sur les valeurs correctes.

API de formats de serveur

L'API de formats de serveur est composé des fonctions de formatage de données suivantes.

applyFormat()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction permet de modifier un document en lui ajoutant une déclaration de fonction de format. Lorsque l'utilisateur choisit un format dans le champ de texte Format des boîtes de dialogue Données dynamiques ou Texte dynamique ou du panneau Liaisons, Dreamweaver apporte deux modifications au document : il ajoute la fonction de format appropriée avant la balise (si elle n'est pas déjà présente) et il change l'objet de données dynamiques de façon à ce qu'il appelle la fonction de format appropriée.

Dreamweaver ajoute la déclaration de fonction en appelant la fonction JavaScript `applyFormat()` dans le fichier de format des données. Il modifie l'objet de données dynamique en appelant la fonction `formatDynamicDataRef()`.

La fonction `applyFormat()` doit utiliser le DOM (Document Object Model, Modèle d'objet de document) pour ajouter des déclarations de fonction au début du document de l'utilisateur. Par exemple, si l'utilisateur sélectionne Devise - Valeur par défaut, la fonction ajoute la déclaration de fonction `Currency`.

Arguments

`format`

- L'argument *format* est un objet JavaScript qui décrit le format à appliquer. Cet objet JavaScript constitue le nœud correspondant à la balise `format` dans le fichier Formats.xml. L'objet a une propriété JavaScript pour chaque attribut de la balise `format` correspondante.

Valeurs renvoyées

Dreamweaver n'attend rien.

applyFormatDefinition()

Disponibilité

Dreamweaver UltraDev 1.

Description

Applique les changements à un format créé à partir de la boîte de dialogue Modifier le format.

Les utilisateurs peuvent créer, modifier ou supprimer des formats à partir de la boîte de dialogue Modifier la liste de formats. Cette fonction est appelée pour appliquer toutes les modifications apportées à un format. Elle peut également définir d'autres propriétés, nommées arbitrairement, pour l'objet. Chaque propriété est stockée en tant qu'attribut de la balise `format` dans le fichier Formats.xml.

Arguments

format

- L'argument *format* correspond à l'objet `format` JavaScript. Cette fonction doit faire de la propriété `expression` de l'objet JavaScript l'expression régulière du format. Elle peut également définir d'autres propriétés, nommées arbitrairement, pour l'objet. Chaque propriété est stockée en tant qu'attribut de la balise `format`.

Valeurs renvoyées

Dreamweaver attend l'objet de format, si la fonction est correctement appliquée. En cas d'erreur, une chaîne d'erreur est renvoyée. Si une chaîne vide est renvoyée, le formulaire est fermé mais le nouveau format n'est pas créé, ce qui revient à annuler l'opération.

deleteFormat()

Disponibilité

Dreamweaver UltraDev 1.

Description

Supprime la déclaration de fonction du format au début du document de l'utilisateur.

Lorsque l'utilisateur modifie le format d'un objet de données dynamiques (dans les boîtes de dialogue Données dynamiques ou Texte dynamique ou dans le panneau Liaisons) ou supprime un objet de données dynamiques formaté, Dreamweaver supprime la déclaration de fonction au début du document, ainsi que l'appel de la fonction de l'objet de données dynamiques en appelant la fonction `deleteFormat()`.

Utilisez le DOM avec la fonction `deleteFormat()` pour supprimer la déclaration de fonction au début du document ouvert.

Arguments

`format`

- L'argument *`format`* est un objet JavaScript qui décrit le format à supprimer. Cet objet JavaScript constitue le nœud correspondant à la balise `format` dans le fichier Formats.xml.

Valeurs renvoyées

Dreamweaver n'attend rien.

formatDynamicDataRef()

Disponibilité

Dreamweaver UltraDev 1.

Description

Ajoute à l'objet de données dynamiques l'appel de la fonction de format. Lorsqu'un utilisateur choisit un format dans le champ de texte Format de la boîte de dialogue Données dynamiques ou Texte dynamique ou le panneau Liaisons, Dreamweaver apporte deux modifications au document : il ajoute la fonction de format appropriée avant la balise (si elle n'est pas déjà présente) et il change l'objet de données dynamiques de façon à ce qu'il appelle la fonction de format appropriée.

Dreamweaver ajoute la déclaration de fonction en appelant la fonction JavaScript `applyFormat()` dans le fichier de format des données. Il modifie l'objet de données dynamique en appelant la fonction `formatDynamicDataRef()`.

La fonction `formatDynamicDataRef()` est appelée lorsque l'utilisateur sélectionne un format du champ de texte Format de la boîte de dialogue Données dynamiques ou Texte dynamique ou du panneau Liaisons. Elle ne modifie pas le document de l'utilisateur.

Arguments

dynamicDataObject, *format*

- L'argument *dynamicDataObject* est une chaîne contenant l'objet de données dynamiques.
- L'argument *format* est un objet JavaScript qui décrit le format à appliquer. Cet objet JavaScript constitue le nœud correspondant à la balise `format` dans le fichier Formats.xml. L'objet a une propriété JavaScript pour chaque attribut de la balise `format` correspondante.

Valeurs renvoyées

Dreamweaverattend la nouvelle valeur de l'objet de données dynamiques.

Si une erreur a lieu, la fonction affiche un message d'avertissement dans certaines conditions. Si la fonction renvoie une chaîne vide, le champ Format prend la valeur `Aucun`.

inspectFormatDefinition()

Disponibilité

Dreamweaver UltraDev 1.

Description

Initialise les commandes de formulaire lorsqu'un utilisateur modifie un format existant dans la boîte de dialogue Modifier la liste de formats.

Arguments

format

- L'argument *format* est un objet JavaScript qui décrit le format à appliquer. Cet objet JavaScript constitue le nœud correspondant à la balise `format` dans le fichier Formats.xml. L'objet a une propriété JavaScript pour chaque attribut de la balise `format` correspondante.

Valeurs renvoyées

Dreamweaver n'attend rien.

CHAPITRE 18

Composants

18

Macromedia Dreamweaver gère la création de la plupart des types de composants les plus populaires. Dreamweaver vous permet en outre de développer les types de composants intégrés au panneau Composants.

Composants de base

Les programmeurs ont recours à différentes stratégies pour encapsuler leurs travaux. L'*encapsulation* correspond à la création d'une entité dans une boîte noire virtuelle. Il n'est pas nécessaire de comprendre son fonctionnement pour l'utiliser. Vous devez uniquement savoir quelles informations sont requises pour fonctionner et quelles informations sont générées, une fois la tâche terminée. Un programmeur crée par exemple un programme destiné à extraire des informations d'une base de données d'employés. Toute personne, y compris d'autres programmes, peut alors utiliser ce programme pour interroger la base de données. Le programme est ainsi réutilisable.

L'expérience prouve que les programmes bien organisés qui font appel à l'encapsulation sont plus faciles à gérer, à améliorer et à réutiliser. Les différentes technologies offrent aux programmeurs différentes façons d'effectuer l'encapsulation, et ces stratégies portent différents noms : *fonctions*, *modules* et autres. Macromedia Dreamweaver 8 utilise le terme de *composant* pour décrire certaines des stratégies d'encapsulation les plus répandues et les plus modernes, comme les services Web, JavaBeans et composants ColdFusion (CFC). Lorsque les utilisateurs créent des applications Web dans Dreamweaver, le panneau Composants les aide à utiliser les services Web, JavaBeans et CFC disponibles.

Les composants provenant de technologies récentes (comme les services Web, JavaBeans ou les CFC) sont capables de s'autodécrire. En règle générale, des informations relatives au composant sont imbriquées dans les fichiers qui constituent le composant. La capacité d'un composant à publier ou partager ces informations porte le nom d'*introspection*. En d'autres termes, un programme comme Dreamweaver peut demander à un composant de lui fournir la liste des fonctions qu'il expose (c'est-à-dire les fonctions qu'il peut appeler à partir d'un autre programme). En fonction de la technologie utilisée, un composant peut révéler d'autres informations à son sujet. Par exemple, un service Web pourrait décrire de nouveaux types de données.

Extension du panneau Composants

Si vous avez inventé (ou si vous utilisez) une stratégie qui n'est pas représentée dans le panneau Composants actuel de Dreamweaver, vous pouvez étendre la logique du panneau Composants de façon à ce qu'il prenne en charge de nouveaux types de composants.

Pour ajouter un nouveau type de composant au panneau Composants de Dreamweaver, vous devez localiser les composants disponibles (dans l'environnement utilisateur) et demander des descriptions de chaque composant (ou les analyser s'ils sont écrits à l'aide de fichiers ASCII).

La façon précise dont l'emplacement et les détails des composants sont extraits dépend des technologies. Elle peut également varier selon le modèle de serveur (ASP.NET, JSP/J2EE, ColdFusion ou autre). Le code JavaScript que vous écrivez pour étendre le panneau Composants dépend donc de la technologie de composant à ajouter. Les fonctions décrites ici ont pour but de vous aider à faire apparaître des informations dans le panneau Composants, mais vous devez écrire une bonne partie de la logique de localisation et d'introspection des composants (en interrogeant la structure interne du composant et en rendant ses champs, méthodes et propriétés disponibles via Dreamweaver).

Enfin, les modèles de serveur comme ASP.NET, JSP/J2EE ou ColdFusion tendent à prendre en charge certains types seulement de composants. Par exemple, ASP.NET prend en charge les services Web mais pas les JavaBeans. Macromedia ColdFusion prend également en charge les services Web et les CFC. Lorsque vous ajoutez un type de composant au panneau Composants, il doit être spécifique au modèle de serveur. Ainsi, si un utilisateur Dreamweaver travaille sur un site ColdFusion, seuls Services Web et CF Components doivent être recensés dans la liste déroulante du panneau Composants.

Les fichiers à modifier sont présentés dans ce chapitre. Dans certains cas, vous devrez écrire un code JavaScript appelant certaines fonctions liées aux composants.

Personnalisation du panneau Composants

Le panneau Composants de Dreamweaver permet aux utilisateurs de charger et d'utiliser des composants. Il répertorie tous les types de composants disponibles et compatibles avec chaque modèle de serveur activé. Ainsi, comme la technologie JavaBeans ne peut fonctionner que dans une page JSP (JavaServer Page), les composants JavaBeans ne sont affichés dans le panneau Composants que pour le modèle de serveur JSP. De même, dans la mesure où les composants ColdFusion ne fonctionnent que dans une page ColdFusion, ils ne sont affichés dans le panneau Composants que pour le modèle de serveur ColdFusion.

Vous pouvez ajouter des types de composants au panneau Composants en créant une extension. Vous devez exécuter la procédure ci-dessous pour ajouter un nouveau type de composant au panneau Composants :

1. Ajoutez le composant à la liste des types de composants disponibles associée au(x) modèle(s) de serveur approprié(s).
2. Ajoutez des instructions, appelées étapes de configuration, qui s'affichent sous la forme d'étapes numérotées interactives de configuration des composants dans le panneau Composants ou dans une boîte de dialogue, selon l'extension pour laquelle vous les implémentez. Assurez-vous que chaque étape exécutée par l'utilisateur est cochée.
3. Recensez les composants associés au type défini sur l'ordinateur de l'utilisateur ou sur le site actuel uniquement.
4. Créez un composant lorsque l'utilisateur clique sur le bouton Plus (+) dans le panneau Composants.

Il est également judicieux de permettre à l'utilisateur de modifier un composant existant ou de le supprimer.

Fichiers du panneau Composants

Le dossier Configuration/Components contient un sous-dossier pour chaque modèle de serveur implémenté. Les fichiers de composant sont stockés dans le dossier Configuration/Components/*modèle-serveur*/*TypeComposant*. Vous pouvez ajouter d'autres modèles de serveur et extensions de serveur compatibles (pour plus d'informations, voir Chapitre 19, “Modèles de serveur,” on page 443 et Chapitre 15, “Comportements de serveur,” on page 335).

Pour créer un composant personnalisé fonctionnant dans le panneau Composants :

- Créez un fichier HTML identifiant les emplacements des fichiers JavaScript et d'images de prise en charge.
- Ecrivez le code JavaScript permettant d'activer le composant.
- Créez des fichiers d'image GIF représentant le composant dans le panneau Composants (ou identifiez-les s'il en existe déjà).

Si vous souhaitez qu'un type de composant s'affiche dans la commande d'arborescence, vous devez aussi créer les fichiers facultatifs associés et compléter la commande d'arborescence.

Vous pouvez définir le type composant de sorte qu'il soit utilisable au niveau d'une page Web précise, d'un ensemble de pages Web ou d'un site Web entier. Le code JavaScript doit inclure la logique de persistance du composant pour permettre son auto-enregistrement entre les sessions et son chargement à chaque début de session.

L'exemple suivant illustre une entrée de données dans le fichier JavaBeansList.xml (à enregistrer dans le dossier de configuration multiutilisateur) définissant la classe de composant et son emplacement :

```
<javabeans>
<javabean classname="TestCollection.MusicCollection"
classlocation="d:\music\music.jar"></javabean>
</javabeans>
```

Les JavaBeans doivent contenir la logique leur permettant de s'enregistrer automatiquement dans le dossier de configuration multiutilisateur. De cette façon, la prochaine fois que l'utilisateur lancera une application, le composant se chargera automatiquement à partir du fichier de données enregistré.

Ajout d'un composant de service

Pour ajouter un nouveau service LDAP (Lightweight Directory Access Protocol) à l'aide de Dreamweaver MX :

1. En prenant pour modèle les fichiers de types de composants existants (par exemple, les fichiers du dossier d'application Configuration/Components/Common/WebServices), créez tous les fichiers nécessaires ainsi que les fichiers facultatifs, pour afficher le nouveau type de composant dans le panneau Composants de Dreamweaver, comme indiqué dans le tableau suivant :

Nom de fichier	Description	Obligatoire/ facultatif
.htm	Fichier d'extension qui identifie les autres fichiers de prise en charge JavaScript et GIF.	Obligatoire
.js	Fichier d'extension qui implémente le rappel de l'API des composants.	Obligatoire
.gif	Image qui apparaît dans la liste déroulante Composants.	Obligatoire
*Menus.xml	Lieu de stockage des métadonnées qui définissent la structure du panneau Composants. Bien que le composant WebServices commun n'utilise pas ce fichier, vous pouvez prendre comme exemple le fichier WebServicesMenus.xml du dossier d'application Components/ColdFusion/ WebServices.	Facultatif

*.gif	Images de la barre d'outils, qui peuvent être activées ou désactivées, comme indiqué dans l'exemple suivant :
	`ToolBarImageUp.gif`
	`ToolBarImageDown.gif`
	`ToolBarImageDisabled.gif.`
	
	Ou trois images de nœuds.
		Facultatif

REMARQUE Utilisez le même préfixe pour tous les fichiers se rapportant au même composant, afin que chaque fichier et son composant soient facilement identifiables.

2. Rédigez le code JavaScript destiné à implémenter le nouveau composant de serveur.

   Le fichier d'extension (HTM) définit les emplacements du code JavaScript dans la balise `SCRIPT`. Ces fichiers JavaScript peuvent résider dans le dossier Shared, dans le même dossier que le fichier d'extension ou dans le dossier Common du code s'appliquant aux modèles de serveurs multiples.

   Par exemple, le fichier Configuration/Components/Common/WebServices/WebServices.htm contient la ligne :

   `<SCRIPT SRC="../../Common/WebServices/WebServicesCommon.js"></SCRIPT>.`

Pour plus d'informations sur les fonctions disponibles de l'API des composants, voir *Fonctions de l'API du panneau Composants*, page 431.

CONSEIL	Lorsque vous ajoutez un service, vous pouvez utiliser le panneau Composants pour consulter les méta-informations existantes et vous en servir directement lorsque vous créez l'extension. Dreamweaver vous permet de consulter les composants ajoutés et d'afficher les nœuds dans l'arborescence des composants. Le panneau Composants prend en charge les fonctionnalités de glisser-déplacer et les raccourcis clavier en mode Code.

Remplissage de la commande d'arborescence

Utilisez la propriété `ComponentRec` pour compléter une commande d'arborescence du panneau Composants et faire en sorte qu'elle apparaisse à l'emplacement approprié dans le panneau. Chaque nœud d'une commande d'arborescence doit comporter les propriétés suivantes :

Nom de la propriété	Description	Obligatoire/ facultatif
`name`	Nom de l'élément de nœud dans l'arborescence.	Obligatoire
`image`	Icône de l'élément de nœud dans l'arborescence. Si elle n'est pas spécifiée, une icône par défaut est utilisée.	Facultatif
`hasChildren`	Réagit à un clic sur le bouton Plus (+) ou Moins () dans la commande d'arborescence en chargeant les enfants. Cela vous permet d'utiliser une arborescence qui n'est pas déjà complétée.	Obligatoire
`toolTipText`	Texte de l'info-bulle associée à l'élément de nœud dans l'arborescence.	Facultatif

Nom de la propriété	Description	Obligatoire/ facultatif
`isCodeViewDraggable`	Détermine s'il est possible de glisser-déplacer l'élément vers le mode Code.	Facultatif
`isDesignViewDraggable`	Détermine s'il est possible de glisser-déplacer l'élément vers le mode Création.	Facultatif

Par exemple, les enfants du nœud WebServicesClass suivant sont des méthodes Web :

```
this.name = "TrafficLocatorWebService";
this.image  = "Components/Common/WebServices/WebServices.gif";
this.hasChildren = true;
this.toolTipText = "TrafficLocatorWebService";
this.isCodeViewDraggable = true;
// the following allows of enabling/disabling of the button that appears
// above the Component Tree
this.allowDelete = true;
this.isDesignViewDraggable = false;
```

Fonctions de l'API du panneau Composants

Cette section décrit les fonctions API permettant de remplir le panneau Composants.

getComponentChildren()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie une liste d'objets `ComponentRec` enfant pour l'objet `ComponentRec` parent actif. Pour charger les éléments du niveau racine de l'arborescence, cette fonction doit lire ses métadonnées dans son stock existant.

Arguments

`{parentComponentRec}`

- L'argument *`parentComponentRec`* est l'objet `componentRec` du parent. Si cet argument n'est pas défini, Dreamweaver attend une liste d'objets `ComponentRec` pour le nœud racine.

Valeurs renvoyées

Tableau d'objets `ComponentRec`.

Exemple

Voir la `fonction getComponentChildren(componentRec)` dans le fichier WebServices.js du dossier Configuration/Components/Common/WebServices.

getContextMenuId()

Disponibilité

Dreamweaver MX.

Description

Renvoie l'ID du menu contextuel correspondant au type de composant. Chaque type de composant peut être associé à un menu contextuel. Les menus déroulants du menu contextuel sont définis dans le fichier *ComponentName*Menus.xml. Ils fonctionnent de la même façon que le fichier menu.xml. Une chaîne de menu peut être statique ou dynamique. Les raccourcis clavier (ou touches de raccourci) sont pris en charge.

Arguments

Néant

Valeurs renvoyées

Chaîne définissant l'ID du menu contextuel.

Exemple

L'exemple suivant définit les options de menu du panneau Composants pour les services Web associés au modèle de serveur ASP.NET/C#, et définit les raccourcis clavier correspondant à ce menu :

```
function getContextMenuId()
{
  return "DWWebServicesContext";
}
```

où WWebServicesContext est défini dans le fichier Configuration/Components/ASP.NET_CSharp/WebServices/WebServicesMenus.xml comme suit :

```
<shortcutlist id="DWWebServicesContext">
  <shortcut key="Del" domRequired="false"
  enabled="(dw.serverComponentsPalette.getSelectedNode() != null &&
  (dw.serverComponentsPalette.getSelectedNode().objectType=='Root'))"
  command="clickedDelete();" id="DWShortcuts_ServerComponent_Delete" />
</shortcutlist>
```

```
<menubar name="" id="DWWebServicesContext">
    <menu name="Server Component Popup" id="DWContext_WebServices">
        <menuitem name="Edit Web Service" domRequired="false"
  enabled="dw.serverComponentsPalette.getSelectedNode() != null &&
  (dw.serverComponentsPalette.getSelectedNode().objectType=='Root') &&
  dw.serverComponentsPalette.getSelectedNode().wsRec != null &&
  dw.serverComponentsPalette.getSelectedNode().wsRec.ProxyGeneratorName !=
  null" command="editWebService()"
  id="DWContext_WebServices_EditWebService" />
...
</menubar>
```

getCodeViewDropCode()

Disponibilité

Dreamweaver MX.

Description

Cette fonction extrait le code faisant l'objet d'un glisser-déplacer en mode Code à partir du panneau Composants ou le code coupé, copié ou collé de ce dernier.

Arguments

componentRec

- L'argument *componentRec* est un objet.

Valeurs renvoyées

Chaîne contenant le code de ce composant.

Exemple

L'exemple suivant identifie le code associé à un service Web commun :

```
function getCodeViewDropCode(componentRec)
{
  var codeToDrop="";
  if (componentRec)
  {
    if (componentRec.objectType=="Class")
    {
      codeToDrop =¬
  dw.getExtDataValue("webservices_constructor","insertText");
      codeToDrop =¬
  codeToDrop.replace(RegExp("@@Id@@","g"),componentRec.name);
      codeToDrop =¬
  codeToDrop.replace(RegExp("@@Class@@","g"),componentRec.name);
    }
    else if (componentRec.objectType=="Method")
```

```
    {
      codeToDrop = componentRec.dropCode;
    }
    if(componentRec.dropCode)
    {
      codeToDrop = componentRec.dropCode;
    }
    else
    {
      codeToDrop = componentRec.name;
    }
  }
  return codeToDrop;
}
```

getSetupSteps()

Disponibilité

Dreamweaver MX.

Description

Dreamweaver appelle cette fonction si `setupStepsCompleted()` renvoie zéro ou un nombre entier positif. Cette fonction contrôle les instructions de configuration côté serveur, qui peuvent être implémentées à l'aide d'extensions basées sur une boîte de dialogue modale et sur des composants de serveur.

Cette fonction renvoie un tableau des chaînes que Dreamweaver affiche soit dans la boîte de dialogue des étapes de configuration, soit dans le panneau Composants, selon le type d'extension.

Arguments

Néant

Valeurs renvoyées

Tableau de *n*+1 chaînes, *n* correspondant au nombre d'étapes, comme décrit dans la liste ci-dessous :

- Le titre qui apparaît au-dessus de la liste des étapes de configuration.
- Pour chaque étape, le texte des instructions, qui peut comprendre un balisage HTML, dès lors qu'il est valide au sein d'une balise `li`.

Vous pouvez inclure des liens hypertexte (balises `a`) dans la liste des étapes en utilisant la syntaxe ci-dessous :

```
<a href="#" onMouseDown="handler">Blue Underlined Text</a>
```

La valeur *"handler"* peut être remplacée par l'une des chaînes suivantes ou toute expression JavaScript telle que "`dw.browseDocument('http://www.macromedia.com')`" :

- "`Event:SetCurSite`" ouvre une boîte de dialogue pour définir le site courant.
- "`Event:CreateSite`" ouvre une boîte de dialogue pour créer un nouveau site.
- "`Event:SetDocType`" ouvre une boîte de dialogue pour modifier le type de document de l'utilisateur.
- "`Event:CreateConnection`" ouvre une boîte de dialogue pour créer une nouvelle connexion à une base de données.
- "`Event:SetRDSPassword`" ouvre une boîte de dialogue pour définir le nom d'utilisateur et le mot de passe du service RDS (Remote Development Service) (ColdFusion uniquement).
- "`Event:CreateCFDataSource`" ouvre l'administrateur ColdFusion dans un navigateur.

Exemple

L'exemple suivant définit quatre étapes pour les composants ColdFusion et fournit un lien hypertexte à la quatrième étape pour que l'utilisateur puisse entrer son nom d'utilisateur et son mot de passe RDS :

```
function getSetupSteps()
{
  var doSDK = false;
  dom = dw.getDocumentDOM();
  if (dom && dom.serverModel)
  {
    var aServerModelName = dom.serverModel.getDisplayName();
  }
  else
  {
    var aServerModelName = site.getServerDisplayNameForSite();
  }
  if (aServerModelName.length)
  {
    if(aServerModelName != "ColdFusion")
    {
      if(needsSDKInstalled != null)
      {
        doSDK = needsSDKInstalled();
      }
    }
  }

  var someSteps = new Array();
  someSteps.push(MM.MSG_WebService_InstructionsTitle);
  someSteps.push(MM.MSG_Dynamic_InstructionsStep1);
  someSteps.push(MM.MSG_Dynamic_InstructionsStep2);
```

```
  if(doSDK == true)
  {
    someSteps.push(MM.MSG_WebService_InstructionsStep3);
  }
  someSteps.push(MM.MSG_WebService_InstructionsStep4);

  return someSteps;
}
```

setupStepsCompleted()

Disponibilité

Dreamweaver MX.

Description

Dreamweaver appelle cette fonction avant que l'onglet Composants ne devienne visible. Dreamweaver appelle ensuite la fonction `getSetupSteps()` si la fonction `setupStepsCompleted()` renvoie zéro ou un entier positif.

Arguments

Néant

Valeurs renvoyées

Entier représentant le nombre d'étapes de configuration déjà effectuées par l'utilisateur, comme décrit dans la liste suivante :

- La valeur zéro ou un nombre entier positif indique le nombre d'étapes déjà effectuées.
- La valeur `-1` indique que toutes les étapes de configuration nécessaires ont été effectuées ; la liste d'instructions n'est donc pas affichée.

handleDesignViewDrop()

Disponibilité

Dreamweaver MX.

Description

Gère l'opération de déplacement lorsque l'utilisateur fait glisser un tableau ou un mode du panneau Base de données ou un composant du panneau Composants vers le mode Création.

Arguments

`componentRec`

- L'argument *`componentRec`* est un objet qui comporte les propriétés suivantes :

- La propriété `name` correspond au nom de l'élément de nœud dans l'arborescence.
- La propriété `image` est une icône facultative pour l'élément de nœud dans l'arborescence. Si cette icône est omise, Dreamweaver MX utilise une icône par défaut.
- La propriété `hasChildren` est une valeur booléenne qui indique si l'élément de nœud dans l'arborescence peut être développé : si la réponse est `true`, Dreamweaver MX affiche les boutons Plus (+) et Moins (-) pour l'élément de nœud dans l'arborescence ; si la réponse est `false`, l'élément ne peut pas être développé.
- La propriété `toolTipText` est le texte facultatif de l'info-bulle associée à l'élément de nœud dans l'arborescence.
- La propriété `isCodeViewDraggable` est une valeur booléenne qui indique s'il est possible de glisser-déplacer l'élément de nœud dans l'arborescence vers le mode Code.
- La propriété `isDesignViewDraggable` est une valeur booléenne qui indique s'il est possible de glisser-déplacer l'élément de nœud dans l'arborescence vers le mode Création.

Valeurs renvoyées

Valeur booléenne indiquant si l'opération de déplacement a réussi : `true` en cas de succès et `false` dans le cas contraire.

Exemple

L'exemple suivant détermine si le composant est un tableau ou un mode et renvoie la valeur `bHandled` appropriée :

```
function handleDesignViewDrop(componentRec)
{
  var bHandled = false;
  if (componentRec)
  {
    if ((componentRec.objectType == "Table")||
        (componentRec.objectType == "View"))
    {
      alert("popup Recordset Server Behavior");
      bHandled = true;
    }
  }
  return bHandled;
}
```

handleDoubleClick()

Disponibilité

Dreamweaver MX.

Description

Lorsque l'utilisateur double-clique sur un nœud de l'arborescence, le gestionnaire d'événements est appelé pour que l'utilisateur puisse apporter des modifications. Cette fonction est facultative. Elle peut renvoyer la valeur `false`, ce qui indique que le gestionnaire d'événements n'est pas défini. Dans ce cas, le fait de double-cliquer déclenche le comportement par défaut, à savoir le développement ou la réduction des nœuds de l'arborescence.

Arguments

`componentRec`

- L'argument *`componentRec`* est un objet qui comporte les propriétés suivantes :
  - La propriété `name` est le nom de l'élément de nœud dans l'arborescence.
  - La propriété `image` est une icône facultative pour l'élément de nœud dans l'arborescence. Si cette icône est omise, Dreamweaver utilise une icône par défaut.
  - La propriété `hasChildren` est une valeur booléenne qui indique si l'élément de nœud dans l'arborescence peut être développé : si la réponse est `true`, Dreamweaver affiche les boutons Plus (+) et Moins (-) pour l'élément de nœud dans l'arborescence ; si la réponse est `false`, l'élément ne peut pas être développé.
  - La propriété `toolTipText` est un texte d'info-bulle facultatif pour l'élément de nœud dans l'arborescence.
  - La propriété `isCodeViewDraggable` est une valeur booléenne qui indique s'il est possible de glisser-déplacer l'élément de nœud dans l'arborescence vers le mode Code.
  - La propriété `isDesignViewDraggable` est une valeur booléenne qui indique s'il est possible de glisser-déplacer l'élément de nœud dans l'arborescence vers le mode Création.

Valeurs renvoyées

Aucune.

Exemple

Dans l'exemple suivant, l'extension est à même de gérer un double-clic sur l'élément de nœud dans l'arborescence ; si la valeur `false` est renvoyée, le comportement par défaut consiste à développer/réduire les nœuds.

```
function handleDoubleClick(componentRec)
{
  var selectedObj = dw.serverComponentsPalette.getSelectedNode();
  if(dwscripts.IS_WIN)
  {
    if (selectedObj && selectedObj.wsRec &&
  selectedObj.wsRec[ProxyGeneratorNamePropName])
```

```
    {
      if (selectedObj.objectType == "Root")
      {
        editWebService();
        return true;
      }
      else if (selectedObj.objectType == "MissingProxyGen")
      {
        displayMissingProxyGenMessage(componentRec);
        editWebService();
        return true;
      }
    }
  }
  return false;
}
```

toolbarControls()

Disponibilité

Dreamweaver MX.

Description

Chaque type de composant renvoie une liste d'objets `toolBarButtonRec` représentant les icônes de la barre d'outils, de gauche à droite. Chaque objet `toolBarButtonRec` contient les propriétés suivantes :

Nom de propriété	Description
`image`	Chemin d'accès au fichier d'image.
`disabledImage`	(Facultatif) Recherche de chemin d'accès à l'image désactivée correspondant au bouton de la barre d'outils.
`pressedImage`	(Facultatif) Recherche de chemin d'accès à l'image correspondant au bouton enfoncé dans la barre d'outils.
`toolTipText`	Info-bulle du bouton dans la barre d'outils.
`toolStyle`	Gauche ou droite.
`enabled`	Code JavaScript qui renvoie une valeur booléenne (`true` ou `false`). Les activateurs sont appelés si les conditions suivantes sont remplies : • La fonction `dreamweaver.serverComponents.refresh()` est appelée. • La sélection dans l'arborescence change. • Le modèle de serveur change.

Nom de propriété	Description
`command`	Code JavaScript à exécuter. Le gestionnaire de commande peut utiliser la fonction `dreamweaver.serverComponents.refresh()` pour actualiser les composants.
`menuId`	ID de menu unique du bouton du menu déroulant lorsque l'utilisateur clique sur le bouton. Lorsque cet ID est défini, il remplace le gestionnaire de commande. En d'autres termes, le bouton peut être soit associé à une commande, soit à un bouton ayant un menu déroulant associé, mais pas aux deux à la fois.

Arguments

Néant

Valeurs renvoyées

Tableau des boutons de barre d'outils, de gauche à droite.

Exemple

Dans l'exemple suivant, des propriétés sont attribuées aux boutons de la barre d'outils :

```
function toolbarControls()
{
  var toolBarBtnArray = new Array();

  dom = dw.getDocumentDOM();
  var plusButton = new ToolbarControlRec();
  var aServerModelName = null;
  if (dom && dom.serverModel)
  {
    aServerModelName = dom.serverModel.getDisplayName();
  }
  else
  {
    //look in the site for potential server model
    aServerModelName = site.getServerDisplayNameForSite();
  }

  if (aServerModelName.length)
  {
    if(aServerModelName == "ColdFusion")
    {
      plusButton.image= PLUS_BUTTON_UP;
      plusButton.pressedImage= PLUS_BUTTON_DOWN;
      plusButton.disabledImage= PLUS_BUTTON_UP;
      plusButton.toolStyle= "left";
      plusButton.toolTipText= MM.MSG_WebServicesAddToolTipText;
```

```
    plusButton.enabled= "dwscripts.IS_WIN";
    plusButton.command= "invokeWebService()";
  }
  else
  {
    plusButton.image= PLUSDROPBUTTONUP;
    plusButton.pressedImage= PLUSDROPBUTTONDOWN;
    plusButton.disabledImage= PLUSDROPBUTTONUP;
    plusButton.toolStyle= "left";
    plusButton.toolTipText= MM.MSG_WebServicesAddToolTipText;
    plusButton.enabled= "dwscripts.IS_WIN";
    plusButton.menuId = "DWWebServicesChoosersContext";
  }
  toolBarBtnArray.push(plusButton);

  var minusButton = new ToolbarControlRec();
  minusButton.image= MINUSBUTTONUP;
  minusButton.pressedImage= MINUSBUTTONDOWN;
  minusButton.disabledImage= MINUSBUTTONDISABLED;
  minusButton.toolStyle= "left";
  minusButton.toolTipText= MM.MSG_WebServicesDeleteToolTipText;
  minusButton.command = "clickedDelete()";
  minusButton.enabled = "(dw.serverComponentsPalette.getSelectedNode()
!= null && dw.serverComponentsPalette.getSelectedNode() &&
((dw.serverComponentsPalette.getSelectedNode().objectType=='Root') ||
(dw.serverComponentsPalette.getSelectedNode().objectType == 'Error') ||
(dw.serverComponentsPalette.getSelectedNode().objectType ==
'MissingProxyGen')))";
  toolBarBtnArray.push(minusButton);

  if(aServerModelName != null && aServerModelName.indexOf(".NET") >= 0)
  {
    var deployWServiceButton = new ToolbarControlRec();
    deployWServiceButton.image= DEPLOYSUPPORTBUTTONUP;
    deployWServiceButton.pressedImage= DEPLOYSUPPORTBUTTONDOWN;
    deployWServiceButton.disabledImage= DEPLOYSUPPORTBUTTONUP;
    deployWServiceButton.toolStyle= "right";
    deployWServiceButton.toolTipText=
MM.MSG_WebServicesDeployToolTipText;
    deployWServiceButton.command =
"site.showTestingServerBinDeployDialog()";
    deployWServiceButton.enabled = true;
    toolBarBtnArray.push(deployWServiceButton);
  }
  //add the rebuild proxy button for windows only.
  //bug 45552:
  if(navigator.platform.charAt(0) !="M")
  {
    var proxyButton = new ToolbarControlRec();
    proxyButton.image= PROXYBUTTONUP;
```

```
        proxyButton.pressedImage= PROXYBUTTONDOWN;
        proxyButton.disabledImage= PROXYBUTTONDISABLED;
        proxyButton.toolStyle= "right";
        proxyButton.toolTipText= MM.MSG_WebServicesRegenToolTipText;
        proxyButton.command = "reGenerateProxy()";
        proxyButton.enabled = "enableRegenerateProxyButton()";
        toolBarBtnArray.push(proxyButton);
      }
    }

    return toolBarBtnArray;

}
```

CHAPITRE 19

Modèles de serveur

19

Les modèles de serveur sont les technologies qui permettent d'exécuter des scripts sur un serveur. Lorsqu'ils définissent un nouveau site, les utilisateurs peuvent indiquer le modèle de serveur qu'ils souhaitent utiliser au niveau du site et au niveau du document. Ce modèle de serveur gère tous les éléments dynamiques que l'utilisateur ajoute au document.

Les fichiers de configuration des modèles de serveur sont stockés dans le dossier Configuration/ServerModels. Chaque modèle de serveur dispose, au sein de ce dossier, d'un fichier HTML particulier qui implémente l'ensemble des fonctions requises.

Fonctionnement de la personnalisation des modèles de serveur

Vous pouvez personnaliser certaines des fonctionnalités d'un modèle de serveur en utilisant les fonctions disponibles dans l'API des modèles de serveur.

Lorsqu'ils démarrent Macromedia Dreamweaver 8 pour la première fois, les nouveaux utilisateurs sont invités à indiquer les modèles de serveur qu'ils souhaitent utiliser. Pour les cas où l'utilisateur n'indique aucun modèle de serveur, vous pouvez créer une boîte de dialogue dynamique l'invitant à suivre les étapes nécessaires. Cette boîte de dialogue apparaîtra lorsque l'utilisateur essaiera d'insérer un objet de serveur. Pour plus d'informations sur la création de cette boîte de dialogue, voir les fonctions getSetupSteps() et setupStepsCompleted().

Vous pouvez décider de créer un modèle de serveur spécialisé. Macromedia vous conseille de créer un modèle de serveur entièrement nouveau plutôt que de modifier l'un de ceux fournis avec Dreamweaver. Pour plus d'informations sur la création de types de documents pris en charge par un modèle de serveur, voir *Types de documents extensibles dans Dreamweaver*, page 37.

Lorsque vous créez un modèle de serveur, vous devez implémenter la fonction `canRecognizeDocument()` dans votre fichier de modèle de serveur. Cette fonction indique à Dreamweaver le niveau de priorité qu'il doit attribuer au modèle de serveur en matière de gestion des extensions de fichier lorsque plusieurs modèles de serveur revendiquent une extension précise.

Fonctions de l'API des modèles de serveur

Cette section décrit les fonctions qui permettent de configurer les modèles de serveur pour Dreamweaver.

canRecognizeDocument()

Disponibilité

Dreamweaver MX.

Description

Lors de l'ouverture d'un document (et lorsque plusieurs modèles de serveur revendiquent une extension de fichier), Dreamweaver appelle cette fonction pour chacun des modèles de serveur associés à l'extension afin de voir si l'une des fonctions peut établir que le document lui appartient. Si plusieurs modèles de serveur revendiquent l'extension de fichier, Dreamweaver accorde la priorité à celui renvoyant le nombre entier le plus élevé.

REMARQUE	Dans la mesure où tous les modèles de serveur de Dreamweaver renvoient la valeur `1`, les modèles de serveur tiers peuvent avoir la priorité au niveau d'une extension de fichier.

Arguments

`dom`

- L'argument *`dom`* correspond à l'objet de document Macromedia, renvoyé par la fonction `dreamweaver.getDocumentDOM()`.

Valeurs renvoyées

Dreamweaver attend un nombre entier indiquant la priorité que vous accordez au modèle de serveur pour l'extension de fichier. Cette fonction doit renvoyer la valeur `-1` si le modèle de serveur ne revendique pas l'extension de fichier ; dans les autres cas de figure, elle doit renvoyer une valeur supérieure à zéro.

Exemple

Dans l'exemple suivant, si l'utilisateur ouvre un document JavaScript pour le modèle de serveur en cours, le code renvoie la valeur `2`. Cette valeur permet au modèle de serveur courant d'être prioritaire sur le modèle de serveur par défaut de Dreamweaver.

```
var retVal = -1;
var langRE = /@\s*language\s*=\s*(\"|\')?javascript(\"|\')?/i;
// Search for the string language="javascript"
var oHTML = dom.documentElement.outerHTML;
if (oHTML.search(langRE) > -1)
   retVal = 2;
return retVal;
```

getFileExtensions()

Disponibilité

Dreamweaver UltraDev 1, déconseillée dans Dreamweaver MX.

Description

Renvoie les extensions de fichier compatibles avec un modèle de serveur. Par exemple, le modèle de serveur ASP prend en charge les extensions de fichier .asp et .htm. Cette fonction renvoie un tableau de chaînes ; Dreamweaver utilise ces chaînes pour compléter la liste Extension de page par défaut figurant dans la catégorie Serveur d'application de la boîte de dialogue Définition du site.

REMARQUE La liste Extension de page par défaut n'existe que dans Dreamweaver version 4 ou antérieure. Dans Dreamweaver MX et les versions ultérieures, la boîte de dialogue Définition du site ne répertorie pas les paramètres relatifs aux extensions de fichier. Au lieu de cela, Dreamweaver lit le fichier Extensions.txt et analyse l'élément `documenttype` dans le fichier mmDocumentTypes.xml. Pour plus d'informations sur ces deux fichiers et l'élément `documenttype`, voir *Types de documents extensibles dans Dreamweaver*, page 37.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau de chaînes représentant les extensions de fichier autorisées.

getLanguageSignatures()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie un objet qui décrit la méthode et les signatures de tableau utilisées par le langage de script. La fonction `getLanguageSignatures()` permet d'associer des mappages de signatures génériques à des mappages spécifiques au langage pour les éléments suivants :

- La fonction
- Les constructeurs
- Le code de dépôt (valeurs renvoyées)
- Les tableaux
- Les exceptions
- Les mappages des types de données pour les types de données primitifs

La fonction `getLanguageSignatures()` renvoie une table de mappage des déclarations de signature. Les développeurs d'extensions peuvent se servir de cette table de mappage pour produire des blocs de code spécifiques au langage que Dreamweaver place dans la page (en se basant sur le modèle de serveur adapté à la page) lorsqu'un utilisateur effectue un glisser-déplacer sur une méthode relative à des services Web, par exemple.

Pour obtenir des exemples de rédaction de cette fonction, voir les fichiers d'implémentation HTML des modèles de serveur JSP et ASP.Net. Les fichiers d'implémentation des modèles de serveur se trouvent dans le dossier Configuration/ServerModels.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un objet définissant les signatures du langage de script. Cet objet doit mapper les signatures génériques à celles spécifiques au langage.

getServerExtension()

Disponibilité

Dreamweaver UltraDev 4, déconseillée dans Dreamweaver MX.

Description

Cette fonction renvoie l'extension par défaut des fichiers exploitant le modèle de serveur en cours. L'objet `serverModel` est défini sur le modèle de serveur du site sélectionné si aucun document utilisateur n'est sélectionné.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne représentant les extensions de fichier prises en charge.

getServerInfo()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie un objet JavaScript accessible depuis le code JavaScript, par appel de la fonction `dom.serverModel.getServerInfo()`. Par ailleurs, `serverName`, `serverLanguage` et `serverVersion` sont des propriétés spécifiques, également accessibles au moyen des fonctions JavaScript suivantes :

```
dom.serverModel.getServerName()
dom.serverModel.getServerLanguage()
dom.serverModel.getServerVersion()
```

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un objet contenant les propriétés du modèle de serveur.

Exemple

```
var obj = new Object();
obj.serverName = "ASP";
obj.serverLanguage = "JavaScript";
obj.serverVersion = "2.0";
```

```
...
return obj;
```

getServerLanguages()

Disponibilité

Dreamweaver UltraDev 1, déconseillée dans Dreamweaver MX.

Description

Cette fonction renvoie les langages de script pris en charge d'un modèle de serveur avec un tableau de chaînes. Dreamweaver utilise ces chaînes pour compléter la liste Langage de script par défaut figurant dans la catégorie Serveur d'application de la boîte de dialogue Définition du site.

REMARQUE

La liste Langage de script par défaut n'existe que dans Dreamweaver version 4 ou antérieure. Dans Dreamweaver MX et les versions ultérieures, la boîte de dialogue Définition du site ne répertorie pas les langages de script pris en charge et la fonction `getServerLanguages()` n'est pas utilisée. La raison pour laquelle la version MX de Dreamweaver n'utilise pas cette fonction est que chaque modèle de serveur ne dispose dans cette version du programme que d'un seul langage de serveur.

Dans les versions antérieures de Dreamweaver, un modèle de serveur pouvait prendre en charge plusieurs langages de script ; le modèle de serveur ASP, par exemple, prend en charge JavaScript et VBScript.

Si vous souhaitez qu'un fichier du dossier ServerFormats s'applique uniquement à un langage de script spécifique, ajoutez l'instruction suivante de manière à ce qu'elle figure sur la première ligne du fichier HTML :

```
<!-- SCRIPTING-LANGUAGE=XXX -->
```

Dans cet exemple, `XXX` représente le langage de script. Cette instruction génère l'apparition du comportement de serveur dans le menu Plus (+) du panneau Comportements de serveur uniquement lorsque le langage de script sélectionné est `XXX`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau de chaînes représentant les langages de script pris en charge.

getServerModelExtDataNameUD4()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie le nom de l'implémentation du modèle de serveur que Dreamweaver doit utiliser lorsqu'il accède à des fichiers de données d'extension UltraDev 4, qui résident dans le dossier Configurations/ExtensionData.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne, telle que `"ASP/JavaScript"`.

getServerModelDelimiters()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie les délimiteurs de scripts que le serveur d'application utilise et indique si chaque délimiteur peut participer à la fusion des blocs de code. Vous pouvez accéder à la valeur renvoyée depuis le code JavaScript, en appelant la fonction `dom.serverModel.getDelimiters()`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau d'objets dans lequel chaque objet comporte les trois propriétés suivantes :

- La propriété *`startPattern`* est une expression régulière qui correspond au délimiteur d'ouverture du script (tel que `"<%"`).
- La propriété *`endPattern`* est une expression régulière qui correspond au délimiteur de fermeture du script (tel que `"%>"`).

- La propriété *participateInMerge* est une valeur booléenne qui indique si le contenu entouré des délimiteurs utilisés doit (`true`) ou non (`false`) prendre part à la fusion de blocs.

getServerModelDisplayName()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie le nom qui doit s'afficher dans l'interface utilisateur pour le modèle de serveur. Vous pouvez accéder à cette valeur depuis le code JavaScript, en appelant la fonction `dom.serverModel.getDisplayName()`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne, telle que `"ASP JavaScript"`.

getServerModelFolderName()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie le nom du dossier à utiliser pour le modèle de serveur dans le dossier Configuration. Vous pouvez accéder à cette valeur depuis le code JavaScript, en appelant la fonction `dom.serverModel.getFolderName()`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend une chaîne, telle que `"ASP_JS"`.

getServerSupportsCharset()

Disponibilité

Dreamweaver MX.

Description

Cette fonction renvoie une valeur `true` si le serveur en cours prend en charge le jeu de caractères spécifié. Dans JavaScript, vous pouvez déterminer si le modèle de serveur prend en charge un jeu de caractères donné en appelant la fonction `dom.serverModel.getServerSupportsCharset()`.

Arguments

`metaCharSetString`

- L'argument *`metaCharSetString`* est une chaîne qui stocke la valeur de l'attribut « `charset=` » des documents.

Valeurs renvoyées

Dreamweaver attend une valeur booléenne.

getVersionArray()

Disponibilité

Dreamweaver UltraDev 1, déconseillée dans Dreamweaver MX.

Description

Cette fonction extrait le mappage de technologies de serveur et de numéros de version. Cette fonction est appelée par `dom.serverModel.getServerVersion()`.

Arguments

Néant

Valeurs renvoyées

Dreamweaver attend un tableau d'objets de version, chacun étant doté d'un nom et d'une valeur, comme indiqué dans les exemples suivants :

- ASP version 2.0
- ADODB version 2.1

CHAPITRE 20

Traducteurs de données

20

Les traducteurs de données permettent de convertir un balisage spécialisé, basé sur des SSI (Server-Side Includes, inclusions à partir du serveur), des instructions JavaScript conditionnelles ou d'autres codes (PHP3, JSP, CFML ou ASP, par exemple), sous forme de code pouvant être lu et affiché par Macromedia Dreamweaver 8. Dans Dreamweaver, vous pouvez traduire des attributs au sein de balises ainsi que des balises entières ou des blocs de code. Tous les traducteurs de données (blocs/balises ou attributs) sont des fichiers HTML.

Les balises ou les blocs de code convertis doivent être délimités dans des régions verrouillées pour que les marqueurs d'origine soient préservés. Les attributs traduits ne nécessitent pas de verrou, ce qui facilite l'inspection des balises qui les contiennent.

La traduction de données (en particulier pour les balises entières ou les blocs de code) peut impliquer des opérations complexes qui ne peuvent pas être réalisées avec JavaScript ou qui peuvent être réalisées plus efficacement en utilisant le langage C. Si vous connaissez bien le langage C ou C++, lisez également Chapitre 21, “Extensions C,” on page 477.

Le tableau ci-dessous recense les fichiers utilisés pour créer un traducteur de données.

Chemin	Fichier	Description
Configuration/ThirdPartyTags/	*langage*.xml	Contient des informations sur les balises dans le langage de marquage.
Configuration/thirdPartyTags	*langage*.gif	Icônes des balises dans le langage.
Configuration/Translators/	*langage*.htm	Contient des fonctions JavaScript associées au traducteur de données.

Fonctionnement des traducteurs de données

Dreamweaver traite tous les fichiers de traducteur de la même façon, qu'ils convertissent des balises entières ou uniquement des attributs. Au démarrage, Dreamweaver lit tous les fichiers du dossier Configuration/Translators et appelle la fonction `getTranslatorInfo()` pour obtenir des informations sur le traducteur. Dreamweaver ignore tous les fichiers dans lesquels la fonction `getTranslatorInfo()` est absente ou qui contiennent une erreur empêchant de la définir.

REMARQUE

Pour éviter que des erreurs JavaScript n'affectent le démarrage, les erreurs détectées dans un fichier de traducteur ne sont signalées qu'après le chargement de tous les traducteurs. Pour plus d'informations sur les traducteurs de débogage, voir *Recherche de bogues dans le traducteur*, page 462.

Dreamweaver appelle également la fonction `translateMarkup()` dans tous les fichiers de traducteur appropriés (comme indiqué dans les préférences de traduction) chaque fois que l'utilisateur ajoute un nouveau contenu ou modifie un contenu existant qui requiert une traduction. Dreamweaver appelle la fonction `translateMarkup()` lorsque l'utilisateur effectue l'une des opérations suivantes :

- Ouverture d'un fichier dans Dreamweaver
- Retour au mode Création après avoir effectué des modifications dans le panneau HTML ou le mode Code
- Modification des propriétés d'un objet dans le document actif
- Insertion d'un objet (à l'aide du panneau Objets ou du menu Insertion)
- Actualisation du document actif après sa modification dans une autre application
- Application d'un modèle au document
- Collage ou déplacement d'un contenu vers ou à l'intérieur de la fenêtre de document
- Enregistrement des modifications dans un fichier dépendant
- Appel d'une commande, d'un comportement, d'un comportement de serveur, d'un inspecteur de propriétés ou de toute autre extension qui définit la propriété `innerHTML` ou `outerHTML` d'un objet de balise ou la propriété `data` d'un objet de commentaire
- Sélection de Fichier > Convertir > Format compatible avec les navigateurs 3.0
- Sélection de Modifier > Convertir > Tableaux en calques
- Sélection de Modifier > Convertir > Calques en tableaux

- Modification d'une balise ou d'un attribut dans Quick Tag Editor, suivie de l'activation de la touche de tabulation ou Entrée

Choix du type de traducteur

Tous les traducteurs doivent contenir les fonctions `getTranslatorInfo()` et `translateMarkup()` et doivent résider dans le dossier Configuration/Translators. Ils se distinguent toutefois par le type de code qu'ils insèrent dans le document de l'utilisateur et par la façon dont ce code doit être contrôlé, comme indiqué dans la liste suivante :

- Pour traduire de petites parties de balises serveur qui déterminent des valeurs d'attribut ou ajoutent conditionnellement des attributs à une balise HTML standard, vous devez écrire un traducteur d'attributs. Les balises HTML standard contenant des attributs traduits peuvent être contrôlées à l'aide des inspecteurs de propriétés intégrés à Dreamweaver. Il n'est pas nécessaire de rédiger un inspecteur de propriétés personnalisé (voir *Ajout d'un attribut traduit à une balise*, page 455).
- Pour traduire une balise entière (une SSI, par exemple) ou un bloc de code (JavaScript, ColdFusion, PHP ou tout autre script), vous devez écrire un traducteur de blocs/balises. Le code généré par un traducteur de blocs/balises ne peut pas être contrôlé à l'aide des inspecteurs de propriétés intégrés à Dreamweaver. Vous devez rédiger un inspecteur personnalisé pour le contenu traduit si vous souhaitez que les utilisateurs puissent modifier les propriétés du code d'origine (voir *Verrouillage des balises ou des blocs de code traduits*, page 458).

Ajout d'un attribut traduit à une balise

La traduction des attributs est liée à la capacité de l'analyseur Dreamweaver à ignorer le balisage de serveur. Par défaut, Dreamweaver ignore déjà les types de balisage de serveur les plus courants (tels que ASP, CFML et PHP) ; si vous utilisez un balisage de serveur dont les marqueurs de début et de fin sont différents, vous devez modifier la base de données de balises tierces pour assurer le bon fonctionnement de votre traducteur. Pour plus d'informations sur la façon de modifier la base de données de balises propriétaires, voir Personnalisation de Dreamweaver dans *Utilisation de Dreamweaver*.

Lorsque Dreamweaver gère la conservation du balisage de serveur d'origine, le traducteur génère une valeur d'attribut valide qui s'affiche dans la fenêtre de document. Si vous utilisez un balisage de serveur uniquement pour des attributs qui n'ont pas d'effet visible pour l'utilisateur, vous n'avez pas besoin de traducteur.

Le traducteur crée une valeur d'attribut ayant un effet visible dans la fenêtre de document en ajoutant un attribut spécial, `mmTranslatedValue`, à la balise qui contient le balisage de serveur. L'attribut `mmTranslatedValue` et sa valeur ne sont ni visibles dans le panneau HTML ou en mode Code, ni enregistrés avec le document.

L'attribut `mmTranslatedValue` doit être unique à l'intérieur de la balise. S'il est probable que le traducteur devra traduire plusieurs attributs dans une même balise, ajoutez-lui une routine qui ajoute des numéros à l'attribut `mmTranslatedValue` (par exemple, `mmTranslatedValue1`, `mmTranslatedValue2`, etc.).

La valeur de l'attribut `mmTranslatedValue` doit être une chaîne codée en URL contenant au moins une paire attribut/valeur valide. Cela signifie que `mmTranslatedValue="src=%22open.jpg%22"` est une traduction valide pour `src="<? if (dayType == weekday) then open.jpg else closed.jpg" ?>` et `<? if (dayType == weekday) then src="open.jpg" else src="closed.jpg" ?>`. La chaîne `mmTranslatedValue="%22open.jpg%22"` n'est valide pour aucun de ces deux exemples parce qu'elle contient uniquement la valeur, et non l'attribut.

Traduction de plusieurs attributs à la fois

L'attribut `mmTranslatedValue` peut contenir plusieurs paires attribut/valeur valides. Examinez le code non traduit suivant :

```
<img <? if (dayType==weekday) then src="open.jpg" width="320" ¬
  height="100" else
src="closed.jpg" width="100" height="320" ?> alt="We're open 24 ¬
hours a day from
12:01am Monday until 11:59pm Friday">
```

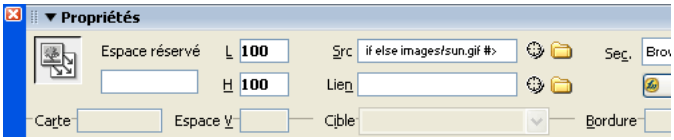
L'exemple suivant indique comment se présente le balisage après traduction :

```
<img <? if (dayType==weekday) then src="open.jpg" width="320" ¬
height="100" else
src="closed.jpg" width="100" height="320" ?>
mmTranslatedValue="src=%22open.jpg%22 width=%22320%22 ¬
  height=%22100%22"
alt="We're open 24 hours a day from 12:01am Monday until 11:59pm ¬
Friday">
```

Les espaces entre les paires attribut/valeur dans l'attribut `mmTranslatedValue` ne sont pas codés. Etant donné que l'application Dreamweaver recherche ces espaces lorsqu'elle essaie de restituer la valeur traduite, chaque paire attribut/valeur de l'attribut `mmTranslatedValue` doit être codée séparément, puis toutes les paires doivent être de nouveau regroupées pour former l'ensemble de l'attribut `mmTranslatedValue`. Pour accéder à un exemple de ce processus, voir *Exemple de traducteur d'attributs simple*, page 463.

Contrôle des attributs traduits

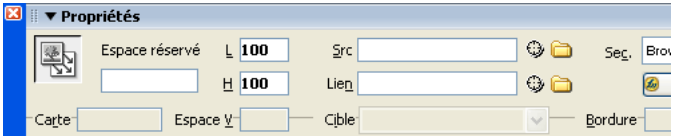
Lorsque le balisage de serveur spécifie un seul attribut et que cet attribut est représenté dans un inspecteur de propriétés, Dreamweaver affiche le balisage dans l'inspecteur de propriétés, comme illustré dans la figure suivante :



Le balisage s'affiche, qu'il soit associé ou non à un traducteur. Le traducteur s'exécute chaque fois que l'utilisateur modifie le balisage de serveur indiqué dans l'inspecteur de propriétés.

Lorsque le balisage de serveur contrôle plusieurs attributs dans une balise, il n'apparaît pas dans l'inspecteur de propriétés. Toutefois, l'éclair indique qu'il existe un balisage traduit pour l'élément sélectionné.

REMARQUE L'icône en forme d'éclair n'apparaît pas lorsque du texte ou des cellules, des lignes ou des colonnes de tableau sont sélectionnés. La traduction continue si l'utilisateur modifie le balisage de serveur dans le panneau et s'il existe un traducteur qui traite ce type de balisage.



Les zones de texte dans l'inspecteur de propriétés sont modifiables. Les utilisateurs peuvent donc saisir des valeurs pour les attributs susceptibles d'être contrôlés par un balisage de serveur, ce qui entraîne l'existence d'attributs en double. Si une valeur traduite et une valeur régulière sont définies pour un attribut donné, Dreamweaver affiche la valeur traduite dans la fenêtre de document. Vous devez décider si le traducteur doit rechercher les attributs en double pour les supprimer.

Verrouillage des balises ou des blocs de code traduits

En règle générale, vous attendez d'un traducteur qu'il modifie le balisage de façon à ce que Dreamweaver puisse l'afficher, mais vous souhaitez sauvegarder celui d'origine, et non les modifications. Dans un tel cas, Dreamweaver fournit des balises XML spéciales pour envelopper le contenu traduit et se reporter au code d'origine.

Lorsque vous utilisez ces balises XML, le contenu des attributs d'origine est reproduit en mode Code. Si vous enregistrez le fichier, le balisage d'origine non traduit est écrit dans le fichier. Le contenu non traduit est ce qui apparaît en mode Code.

La syntaxe des balises XML est indiquée dans l'exemple suivant :

```
<MM:BeginLock translatorClass="translatorClass" ¬
type="tagNameOrType" depFiles="dependentFilesList" ¬
orig="encodedOrignalMarkup">
Translated content
<MM:EndLock>
```

Dans cet exemple, la signification des valeurs en italique est la suivante :

- La valeur *translatorClass* correspond à l'identificateur unique pour le traducteur. C'est la première chaîne du tableau que renvoie la fonction `getTranslatorInfo()`.
- La valeur *tagNameOrType* est une chaîne qui identifie le type de balisage (ou le nom de balise associé) contenu dans le verrou. Cette chaîne ne peut contenir que des caractères alphanumériques, des traits d'union (-) ou des caractères de soulignement (_). Vous pouvez vérifier cette valeur dans la fonction `canInspectSelection()` d'un inspecteur de propriétés personnalisé pour déterminer si cet inspecteur est adapté au contenu. Pour plus d'informations, consultez la section *Création d'inspecteurs de propriétés pour contenu verrouillé*, page 459. Un contenu verrouillé ne peut pas être contrôlé par les inspecteurs de propriétés intégrés de Dreamweaver. Par exemple, la spécification de `type="IMG"` n'entraîne pas l'affichage de l'inspecteur d'images.
- La valeur *dependentFilesList* est une chaîne qui contient une liste de fichiers séparés par des virgules dont dépend le balisage verrouillé. Les fichiers sont référencés sous la forme d'URL relatives au document de l'utilisateur. Si l'utilisateur met à jour l'un des fichiers de la liste *dependentFilesList*, Dreamweaver retraduit automatiquement le contenu dans le document comportant cette liste.

- La valeur *encodedOriginalMarkup* est une chaîne qui contient le balisage d'origine non traduit et codé à l'aide d'un extrait de code URL (utilisez `%22` pour ", `%3C` pour <, `%3E` pour > et `%25` pour %). La façon la plus rapide de coder une chaîne en URL consiste à utiliser la méthode `escape()`. Par exemple, si `myString` est égal à `'<img src="foo.gif">'`, `escape(myString)` renvoie `%3Cimg%20src=%22foo.gif%22%3E`.

L'exemple suivant montre la partie de code verrouillée pouvant être générée à partir de la traduction de la SSI `<!--#include virtual="/footer.html" -->` :

```
<MM:BeginLock translatorClass="MM_SSI" type="ssi" ¬
depFiles="C:\sites\webdev\footer.html" orig="%3C!--#include ¬
virtual=%22/footer.html%22%20--%3E">
<!-- begin footer -->
<CENTER>
<HR SIZE=1 NOSHADE WIDTH=100%>

<BR>

[<A TARGET="_top" HREF="/">home</A>]
[<A TARGET="_top" HREF="/products/">products</A>]
[<A TARGET="_top" HREF="/services/">services</A>]
[<A TARGET="_top" HREF="/support/">support</A>]
[<A TARGET="_top" HREF="/company/">about us</A>]
[<A TARGET="_top" HREF="/help/">help</A>]
</CENTER>
<!-- end footer -->
<MM:EndLock>
```

Création d'inspecteurs de propriétés pour contenu verrouillé

Une fois que vous avez créé un traducteur, vous devez créer un inspecteur de propriétés pour le contenu afin de permettre à l'utilisateur d'en modifier les propriétés (le fichier à inclure ou l'une des conditions d'une instruction conditionnelle, par exemple). Le contrôle d'un contenu traduit pose un problème particulier pour les raisons suivantes :

- L'utilisateur peut décider de modifier les propriétés du contenu traduit. Dans ce cas, ces modifications doivent être répercutées dans le contenu non traduit.
- Le DOM (modèle d'objet de document) contient le contenu traduit (ce qui signifie que les balises de verrouillage et les balises qu'elles englobent sont des nœuds du DOM), mais la propriété `outerHTML` de l'objet `documentElement` ainsi que les fonctions `dreamweaver.getSelection()` et `dreamweaver.nodeToOffsets()` agissent sur la source non traduite.

- Les balises que vous contrôlez sont différentes avant et après la traduction.

Un commentaire de l'inspecteur de propriétés pour la balise `HAPPY` non traduite peut ressembler, par exemple, au code suivant :

```
<!-- tag:HAPPY,priority:5,selection:exact,hline,vline, attrName:xxx,¬
  attrValue:yyy -->
```

Par contre, un commentaire de ce même inspecteur de propriétés pour la balise `HAPPY` traduite ressemblerait au code suivant :

```
<!-- tag:*LOCKED*,priority:5,selection:within,hline,vline -->
```

La fonction `canInspectSelection()` de l'inspecteur de propriétés `HAPPY` non traduit est simple. Le type `selection` correspondant à `exact`, elle peut renvoyer la valeur `true` sans autre analyse. Pour l'inspecteur de propriétés de la balise `HAPPY` traduite, cette fonction est plus compliquée. Le mot-clé `*LOCKED*` indique que l'inspecteur est approprié lorsque la sélection se trouve à l'intérieur d'une région verrouillée, mais comme un document peut comporter plusieurs régions verrouillées, des vérifications supplémentaires sont nécessaires pour déterminer si l'inspecteur correspond à cette région précise.

Le contrôle d'un contenu traduit présente un problème supplémentaire. Lorsque vous appelez la fonction `dom.getSelection()`, les valeurs renvoyées par défaut sont des décalages dans la source non traduite. Pour étendre la sélection de façon à sélectionner uniquement la région verrouillée, utilisez la méthode suivante :

```
var currentDOM = dw.getDocumentDOM();
var offsets = currentDOM.getSelection();
var theSelection = currentDOM.offsetsToNode(offsets[0],offsets[0]+1);
```

L'utilisation de `offsets[0]+1` comme deuxième argument permet de vous assurer que vous restez dans les limites de la balise de verrouillage de début lorsque vous traduisez les décalages d'octets en nœud. Si vous utilisez `offsets[1]` comme deuxième argument, vous risquez de sélectionner le nœud situé au-dessus du verrou.

Une fois la sélection effectuée et après vous être assuré que `nodeType` est réglé sur `node.ELEMENT_NODE`), vous pouvez contrôler l'attribut `type` pour vérifier que la région verrouillée correspond à cet inspecteur de propriétés, comme indiqué dans l'exemple suivant :

```
if (theSelection.nodeType == node.ELEMENT_NODE && ¬
theSelection.getAttribute('type') == 'happy'){
  return true;
}else{
  return false
}
```

Pour renseigner les zones de texte de l'inspecteur de propriétés pour la balise traduite, vous devez analyser la valeur de l'attribut `orig`. Par exemple, si le code non traduit est `<HAPPY TIME="22">` et que l'inspecteur de propriétés contient une zone de texte Time, vous devez extraire la valeur de l'attribut `TIME` de la chaîne `orig` :

```
function inspectSelection() {
  var currentDOM = dw.getDocumentDOM();
  var currSelection = currentDOM.getSelection();
  var theObj = currentDOM.offsetsToNode¬
  (curSelection[0],curSelection[0]+1);

  if (theObj.nodeType != Node.ELEMENT_NODE) {
    return;
  }

  // To convert the encoded characters back to their
  // original values, use the unescape() method.
  var origAtt = unescape(theObj.getAttribute("ORIG"));

  // Convert the string to lowercase for processing
  var origAttLC = origAtt.toLowerCase();

  var timeStart = origAttLC.indexOf('time="');
  var timeEnd = origAttLC.indexOf('"',timeStart+6);
  var timeValue = origAtt.substring(timeStart+6,timeEnd);

  document.layers['timelayer'].document.timeForm.timefield.¬
  value = timeValue;
}
```

Après avoir analysé l'attribut `orig` en vue de renseigner les zones de texte de l'inspecteur de propriétés pour la balise traduite, vous devrez probablement définir la valeur de l'attribut `orig` si l'utilisateur modifie la valeur de l'une des zones. Il existe certaines restrictions quant aux modifications que vous pouvez effectuer dans une région verrouillée. Vous pouvez contourner ce problème en modifiant le balisage d'origine puis en procédant à une nouvelle traduction.

L'inspecteur de propriétés des SSI traduites (fichier ssi_translated.js du dossier Configuration/Inspectors) démontre cette technique avec sa fonction `setComment()`. Au lieu de réécrire l'attribut `orig`, l'inspecteur assemble un nouveau commentaire de SSI. Il insère ce commentaire dans le document, remplaçant l'ancien en réécrivant le contenu du document, ce qui génère un nouvel attribut `orig`. Le code suivant résume cette technique :

```
// Assemble the new include comment. radioStr and URL are
// variables defined earlier in the code.
newInc = "<!--#include " + radioStr + "=" + '"' + URL + '"' ¬
+" -->";

// Get the contents of the document.
var entireDocObj = dreamweaver.getDocumentDOM();
```

```
var docSrc = entireDocObj.documentElement.outerHTML;

// Store everything up to the SSI comment and everything after
// the SSI comment in the beforeSelStr and afterSelStr variables.
var beforeSelStr = docSrc.substring(0, curSelection[0] );
var afterSelStr  = docSrc.substring(curSelection[1]);

// Assemble the new contents of the document.
docSrc = beforeSelStr + newInc + afterSelStr;

// Set the outerHTML of the HTML tag (represented by
// the documentElement object) to the new contents,
// and then set the selection back to the locked region
// surrounding the SSI comment.
entireDocObj.documentElement.outerHTML = docSrc;
entireDocObj.setSelection(curSelection[0], curSelection[0]+1);
```

Recherche de bogues dans le traducteur

Si la fonction `translateMarkup()` contient certains types d'erreur, le traducteur se charge correctement, mais il échoue sans afficher de message d'erreur si vous essayer de l'appeler. Bien que cette panne discrète empêche Dreamweaver de devenir instable, elle peut ralentir le développement, surtout lorsque vous devez rechercher une erreur de syntaxe mineure dans plusieurs lignes de code.

Si l'exécution du traducteur échoue, une méthode de débogage efficace consiste à convertir le traducteur en commande, en procédant comme suit :

1. Copiez tout le contenu du fichier du traducteur dans un nouveau document, puis enregistrez-le dans le sous-dossier Configuration/Commands du dossier de l'application Dreamweaver.
2. Au début du document, entre les balises `SCRIPT`, ajoutez la fonction suivante :

   ```
   function commandButtons(){
     return new Array( "OK","translateMarkup(dreamweaver.¬
     getDocumentPath('document'), dreamweaver.getSiteRoot(), ¬
     dreamweaver.getDocumentDOM().documentElement.outerHTML); ¬
     window.close()", "Cancel", "window.close()");
   }
   ```

3. A la fin de la fonction `translateMarkup()`, appliquez un commentaire à la ligne `return` *`whateverTheReturnValueIs`* et remplacez-la par `dreamweaver.getDocumentDOM().documentElement.outerHTML =` *`whateverTheReturnValueIs`*, comme indiqué dans l'exemple suivant :

   ```
   // return theCode;
   dreamweaver.getDocumentDOM().documentElement.outerHTML = ¬
   ```

```
  theCode;
}
/* end of translateMarkup() */
```

4. Dans la balise `BODY` du document, ajoutez un formulaire sans zones de texte :

```
<body>
<form>
Hello.
</form>
</body>
```

5. Redémarrez Dreamweaver, puis sélectionnez la commande relative à votre traducteur dans le menu Commandes. Lorsque vous cliquez sur OK, la fonction `translateMarkup()` est appelée et simule une traduction.

   Si aucun message d'erreur ne s'affiche et que la traduction continue d'échouer, il est probable que votre code contienne une erreur de logique.

6. Ajoutez des instructions `alert()` à des points stratégiques de la fonction `translateMarkup()` de façon à vous assurer que vous touchez les ramifications appropriées et à vérifier les valeurs des variables et des propriétés à des points différents :

```
for (var i=0; i< foo.length; i++){
  alert("we're at the top of foo.length array, and the value ¬
  of i is " + i);
  /* rest of loop */
}
```

7. Une fois les instructions `alert()` ajoutées, choisissez votre commande dans le menu Commandes, cliquez sur Annuler, puis choisissez-la de nouveau. Cette opération recharge le fichier de commandes en intégrant vos modifications.

Exemple de traducteur d'attributs simple

Pour mieux comprendre le principe de la traduction d'attributs, il peut être utile d'examiner un exemple. Le traducteur suivant est un balisage « Pound Conditional » (Poco), une syntaxe quelque peu similaire à ASP ou PHP.

Vous créez le traducteur d'attributs en exécutant les étapes suivantes :

- Création de la balise tagspec
- Création de l'icône
- Création du traducteur d'attributs

Création de la balise tagspec

La première étape pour assurer le bon fonctionnement de ce traducteur consiste à créer une balise `tagspec` pour le balisage Poco afin d'éviter que Dreamweaver n'analyse les instructions Poco non traduites.

Pour créer la balise tagspec :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<tagspec tag_name="poco" start_string="<#" end_string="#>" ¬
detect_in_attribute="true" icon="poco.gif" icon_width="17" ¬
icon_height="15"></tagspec>
```

3. Enregistrez le fichier sous le nom poco.xml dans le dossier Configuration/ThirdPartyTags.

Création de l'icône

Vous créez ensuite l'icône associée aux balises Poco.

Pour créer l'icône :

1. Créez un fichier d'image de 18 x 18 pixels pour l'icône des balises Poco.
2. Enregistrez le fichier sous le nom poco.gif dans le dossier Configuration/ThirdPartyTags.

Création du traducteur d'attributs

Vous créez un fichier HTML contenant les fonctions requises pour le traducteur d'attributs.

Pour créer le fichier HTML :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<html>
<head>
<title>Conditional Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
<script language="JavaScript">

/*************************************************************
 * This translator handles the following statement syntaxes: *
 * <* # if (condition) then foo else bar #>                    *
 * <* # if (condition) then att="foo" else att="bar" #>        *
 * <* # if (condition) then att1="foo" att2="jinkies"          *
 * att3="jeepers" else att1="bar" att2="zoinks" #>           *
 *                                                           *
```

```
 * It does not handle statements with no else clause.        *
 ************************************************************/

var count = 1;

function translateMarkup(docNameStr, siteRootStr, inStr){
var count = 1;
// Counter to ensure unique mmTranslatedValues
var outStr = inStr;
// String that will be manipulated
var spacer = "";
// String to manage space between encoded attributes
var start = inStr.indexOf('<# if'); // 1st instance of Pound Conditional
  code

// Declared but not initalized. //
var attAndValue;
// Boolean indicating whether the attribute is part of
// the conditional statement
var trueStart;
// The beginning of the true case
var falseStart;
// The beginning of the false case
var trueValue;
// The HTML that would render in the true case
var attName;
// The name of the attribute that is being'
// set conditionally.
var equalSign;
// The position of the equal sign just to the
// left of the <#, if there is one
var transAtt;
// The entire translated attribute
var transValue;
// The value that must be URL-encoded
var back3FromStart;
// Three characters back from the start position
// (used to find equal sign to the left of <#
var tokens;
// An array of all the attributes set in the true case
var end;
// The end of the current conditional statement.


// As long as there's still a <# conditional that hasn't been
// translated
while (start != -1){
   back3FromStart = start-3;
   end = outStr.indexOf(' #>',start);
   equalSign = outStr.indexOf('="<# if',back3FromStart);
```

```
attAndValue = (equalSign != -1)?false:true;
trueStart = outStr.indexOf('then', start);
falseStart = outStr.indexOf(' else', start);
trueValue = outStr.substring(trueStart+5, falseStart);
tokens = dreamweaver.getTokens(trueValue,' ');


// If attAndValue is false, find out what attribute you're
// translating by backing up from the equal sign to the
// first space. The substring between the space and the
// equal sign is the attribute.
if (!attAndValue){
  for (var i=equalSign; i > 0; i--){
    if (outStr.charAt(i) == " "){
      attName = outStr.substring(i+1,equalSign);
      break;
    }
  }
  transValue = attName + '="' + trueValue + '"';
  transAtt = ' mmTranslatedValue' + count + '="' + ¬
  escape(transValue) + '"';
  outStr = outStr.substring(0,end+4) + transAtt + ¬
  outStr.substring(end+4);

// If attAndValue is true, and tokens is greater than
// 1, then trueValue is a series of attribute/value
// pairs, not just one. In that case, each attribute/value
// pair must be encoded separately and then added back
// together to make the translated value.
}else if (tokens.length > 1){
  transAtt = ' mmTranslatedValue' + count + '="'
  for (var j=0; j < tokens.length; j++){
    tokens[j] = escape(tokens[j]);
    if (j>0){
      spacer=" ";
    }
    transAtt += spacer + tokens[j];
  }
  transAtt += '"';
  outStr = outStr.substring(0,end+3) + transAtt + ¬
  outStr.substring(end+3)

// If attAndValue is true and tokens is not greater
// than 1, then trueValue is a single attribute/value pair.
// This is the simplest case, where all that is necessary is
// to encode trueValue.
}else{
  transValue = trueValue;
  transAtt = ' mmTranslatedValue' + count + '="' + ¬
  escape(transValue) + '"';
```

```
      outStr = outStr.substring(0,end+3) + transAtt + ¬
      outStr.substring(end+3);
    }

    // Increment the counter so that the next instance
    // of mmTranslatedValue will have a unique name, and
    // then find the next <# conditional in the code.
    count++;
    start = outStr.indexOf('<# if',end);
   }

 // Return the translated string.
 return outStr
}

function getTranslatorInfo(){
 returnArray = new Array(7);

 returnArray[0] = "Pound_Conditional";     // The translatorClass
 returnArray[1] = "Pound Conditional Translator"; // The title
 returnArray[2] = "2";                // The number of extensions
 returnArray[3] = "html";             // The first extension
 returnArray[4] = "htm";              // The second extension
 returnArray[5] = "1";                // The number of expressions
 returnArray[6] = "<#";               // The first expression
 returnArray[7] = "byString";               //
 returnArray[8] = "50";               //

 return returnArray
}

</script>
</head>

<body>
</body>
</html>
```

3. Enregistrez le fichier sous le nom poco.htm dans le dossier Configuration/Translators.

Exemple de traducteur de blocs/balises simple

Pour vous aider à comprendre la traduction, regardez un traducteur entièrement écrit en JavaScript, qui n'utilise de bibliothèque C pour aucune fonctionnalité. Le traducteur suivant serait plus efficace s'il était écrit en C, mais la version JavaScript est plus simple et constitue de ce fait un exemple idéal pour illustrer le fonctionnement des traducteurs.

Comme la plupart des traducteurs, celui-ci est conçu pour émuler le comportement d'un serveur. Supposons que votre serveur Web soit configuré pour remplacer la balise `KENT` par la photo d'un technicien différent selon le jour de la semaine, l'heure de la journée ou la plate-forme de l'utilisateur. Le traducteur exécute la même opération, mais uniquement localement.

Pour créer le traducteur de blocs/balises :

1. Créez un document vierge.
2. Entrez le code suivant :

```
<html>
<head>
<title>Kent Tag Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
<script language="JavaScript">
/************************************************************
 * The getTranslatorInfo() function provides information  *
 * about the translator, including its class and name,    *
 * the types of documents that are likely to contain the  *
 * markup to be translated, the regular expressions that  *
 * a document containing the markup to be translated      *
 * would match (whether the translator should run on all  *
 * files, no files, in files with the specified           *
 * extensions, or in files matching the specified         *
 * expressions).                                          *
 ************************************************************/
function getTranslatorInfo(){
  //Create a new array with 6 slots in it
  returnArray = new Array(6);

  returnArray[0] = "DREAMWEAVER_TEAM"// The translatorClass
  returnArray[1] = "Kent Tags"// The title
  returnArray[2] = "0" // The number of extensions
  returnArray[3] = "1"// The number of expressions
  returnArray[4] = "<kent"// Expression
  returnArray[5] = "byExpression"// run if the file contains "<kent"
  return returnArray;
}

/
  ************************************************************************
  *****
* The translateMarkup() function performs the actual translation.
  *
* In this translator, the translateMarkup() function is written
  *
* entirely in JavaScript (that is, it does not rely on a C library) --
  *
* and it's also extremely inefficient. It's a simple example, however,
  *
```

```
* which is good for learning.                                              *
*****************************************************************************
  **/
function translateMarkup(docNameStr, siteRootStr, inStr){
  var outStr = "";                 // The string to be returned after
  translation
  var start = inStr.indexOf('<kent>');     // The first position of the
  KENT tag
                                           // in the document.
  var replCode = replaceKentTag();        // Calls the replaceKentTag()
  function
                                    // to get the code that will replace
  KENT.
  var outStr = "";                 // The string to be returned after
  translation

   //If the document does not contain any content, terminate the
  translation.
  if ( inStr.length <= 0 ){
     return "";
  }

  // As long as start, which is equal to the location in inStr of the
  // KENT tag, is not equal to -1 (that is, as long as there is another
  // KENT tag in the document)
  while (start != -1){
     // Copy everything up to the start of the KENT tag.
     // This is very important, as translators should never change
     // anything other than the markup that is to be translated.
     outStr = inStr.substring(0, start);
    // Replace the KENT tag with the translated HTML, wrapped in special
     // locking tags. For more information on the replacement operation,
  see
     // the comments in the replaceKentTag() function.
     outStr = outStr + replCode;

     // Copy everything after the KENT tag.
     outStr = outStr + inStr.substring(start+6);

     // Use the string you just created for the next trip through
     // the document. This is the most inefficient part of all.
     inStr = outStr;
     start = inStr.indexOf('<kent>');

  }
  // When there are no more KENT tags in the document, return outStr.
  return outStr;
}

/*************************************************************
```

```
* The replaceKentTag() function assembles the HTML that will  *
* replace the KENT tag and the special locking tags that will *
* surround the HTML. It calls the getImage() function to      *
* determine the SRC of the IMG tag.                           *
**************************************************************/
function replaceKentTag(){
  // The image to display.
  var image = getImage();
  // The location of the image on the local disk.
  var depFiles = dreamweaver.getSiteRoot() + image;
  // The IMG tag that will be inserted between the lock tags.
  var imgTag = '<IMG SRC="/' + image + '" WIDTH="320" HEIGHT="240"
  ALT="Kent">\n';
  // 1st part of the opening lock tag. The remainder of the tag is
  assembled below.
  var start = '<MM:BeginLock translatorClass="DREAMWEAVER_TEAM"
  type="kent"';
  // The closing lock tag.
  var end = '<MM:EndLock>';

  //Assemble the lock tags and the replacement HTML.
  var replCode = start + ' depFiles="' + depFiles + '"';
  replCode = replCode + ' orig="%3Ckent%3E">\n';
  replCode = replCode + imgTag;
  replCode = replCode + end;

  return replCode;
}

/*****************************************************************
 * The getImage() function determines which image to display      *
 * based on the day of the week, the time of day and the          *
 * user's platform. The day and time are figured based on UTC     *
 * time (Greenwich Mean Time) minus 8 hours, which gives          *
 * Pacific Standard Time (PST). No allowance is made for Daylight *
 * Savings Time in this routine.                                  *
*****************************************************************/
function getImage(){
  var today = new Date();            // Today's date & time.
  var day = today.getUTCDay();       // Day of the week in the GMT time
  zone.
                                     // 0=Sunday, 1=Monday, and so on.
  var hour = today.getUTCHours();    // The current hour in GMT, based on
  the
                                     // 24-hour clock.
  var SFhour = hour - 8;             // The time in San Francisco, based
  on the
                                     // 24-hour clock.
  var platform = navigator.platform; // User's platform. All Windows
  machines
```

```
                                        // are identified by Dreamweaver as
   "Win32",
                                         // all Macs as "MacPPC".
  var imageRef;                        // The image reference to be returned.
// If SFhour is negative, you have two adjustments to make.
   // First, subtract one from the day count because it is already the
   wee
   // hours of the next day in GMT. Second, add SFhour to 24 to
   // give a valid hour in the 24-hour clock.
   if (SFhour < 0){
     day = day - 1;
       // The day count back one would make it negative, and it's
   Saturday,
       // so set the count to 6.
       if (day < 0){
           day = 6;
       }
     SFhour = SFhour + 24;
   }

  // Now determine which photo to show based on whether it's a workday or
   a
  // weekend; what time it is; and, if it's a time and day when Kent is
  // working, what platform the user is on.

  //If it's not Sunday
  if (day != 0){
     //And it's between 10am and noon, inclusive
     if (SFhour >= 10 && SFhour <= 12){
        imageRef = "images/kent_tiredAndIrritated.jpg";
      //Or else it's between 1pm and 3pm, inclusive
     }else if (SFhour >= 13 && SFhour <= 15){
        imageRef =  "images/kent_hungry.jpg";
     //Or else it's between 4pm and 5pm, inclusive
     }else if (SFhour >= 16 && SFhour <= 17){
        //If user is on Mac, show Kent working on Mac
        if (platform == "MacPPC"){
           imageRef = "images/kent_gettingStartedOnMac.jpg";
        //If user is on Win, show Kent working on Win
        }else{
           imageRef = "images/kent_gettingStartedOnWin.jpg";
        }
     //Or else it's after 6pm but before the stroke of midnight
     }else if (SFhour >= 18){
         //If it's Saturday
         if (day == 6){
           imageRef = "images/kent_dancing.jpg";
        //If it's not Saturday, check the user's platform
        }else if (platform == "MacPPC"){
           imageRef = "images/kent_hardAtWorkOnMac.jpg";
        }else{
```

```
            imageRef = "images/kent_hardAtWorkOnWin.jpg";
         }
      }else{
         imageRef = "images/kent_sleeping.jpg";
      }
   //If it's after midnight and before 10am, or anytime on Sunday
   }else{
      imageRef = "images/kent_sleeping.jpg";
   }

   return imageRef;
}

</script>
</head>

<body>
</body>
</html>
```

3. Enregistrez le fichier sous le nom kent.htm dans le dossier Configuration/Translators.

API du traducteur de données

Cette section décrit les fonctions utilisées pour définir les traducteurs de Dreamweaver.

getTranslatorInfo()

Description

Cette fonction affiche des informations sur le traducteur et sur les fichiers qu'il peut affecter.

Arguments

Néant

Valeurs renvoyées

Tableau de chaînes. Les éléments du tableau doivent apparaître dans l'ordre suivant :

1. La chaîne *translatorClass* identifie le traducteur de façon unique. Cette chaîne doit commencer par une lettre et contenir uniquement des caractères alphanumériques, des traits d'union (-) et des caractères de soulignement (_).
2. La chaîne *title* décrit le traducteur en 40 caractères maximum.
3. La chaîne *nExtensions* indique le nombre d'extensions de fichier qui suivent. Si *nExtensions* est égal à zéro, le traducteur peut s'exécuter sur n'importe quel fichier. Si *nExtensions* est égal à zéro, *nRegExps* est l'élément suivant dans le tableau.

4. La chaîne *extension* indique une extension de fichier ("htm" ou "SHTML", par exemple) compatible avec ce traducteur. Cette chaîne ne tient pas compte des majuscules et des minuscules et ne doit pas commencer par un point. Le tableau doit contenir le même nombre d'éléments *extension* que celui spécifié dans *nExtensions*.
5. La chaîne *nRegExps* indique le nombre d'expressions régulières qui suivent. Si *nRegExps* est égal à zéro, *runDefault* est l'élément suivant dans le tableau.
6. La chaîne *regExps* indique une expression régulière que vous pouvez vérifier. Le tableau doit contenir le même nombre d'éléments *regExps* que celui spécifié dans *nRegExps*, et au moins l'un des éléments *regExps* doit correspondre à une partie du code source du document pour que le traducteur puisse traiter un fichier.
7. La chaîne *runDefault* indique le moment d'exécution du traducteur. La liste suivante indique les valeurs de chaîne possibles :

Chaîne	Définition
"allFiles"	Le traducteur s'exécute systématiquement.
"noFiles"	Le traducteur ne s'exécute jamais.
"byExtension"	Le traducteur s'exécute lorsque les fichiers sont dotés de l'une des extensions spécifiées.
"byExpression"	Le traducteur s'exécute lorsque le document contient une correspondance pour l'une des expressions régulières spécifiées.
"bystring"	Le traducteur s'exécute lorsque le document contient une correspondance pour l'une des chaînes spécifiées.

REMARQUE Si vous définissez *runDefault* sur "byExtension" sans spécifier d'extension (voir l'étape 4.), l'effet produit est le même que celui obtenu avec la valeur "allFiles". Si vous définissez *runDefault* sur "byExpression" sans spécifier d'expressions (voir l'étape 6.), l'effet produit est le même que celui obtenu avec la valeur "noFiles".

8. La chaîne *priority* indique la priorité par défaut pour l'exécution de ce traducteur. Cette priorité est un nombre compris entre 0 et 100. Si vous ne définissez pas de priorité, sa valeur par défaut est 100. La priorité maximale est 0 et la priorité minimale, 100. Lorsque plusieurs traducteurs s'appliquent à un document, ce paramètre détermine l'ordre dans lequel ils sont appliqués. La priorité la plus haute est appliquée en premier. Lorsque plusieurs traducteurs ont la même priorité, ils sont appliqués dans l'ordre alphabétique par translatorClass.

Exemple

Dans l'exemple suivant, la fonction `getTranslatorInfo()` fournit des informations relatives à un traducteur pour SSI (Server-Side Includes) :

```
function getTranslatorInfo(){
  var transArray = new Array(11);

  transArray[0] = "SSI";
  transArray[1] = "Server-Side Includes";
  transArray[2] = "4";
  transArray[3] = "htm";
  transArray[4] = "stm";
  transArray[5] = "html";
  transArray[6] = "shtml";
  transArray[7] = "2";
  transArray[8] = "<!--#include file";
  transArray[9] = "<!--#include virtual";
  transArray[10] = "byExtension";
  transArray[11] = "50";

  return transArray;
}
```

translateMarkup()

Description

Cette fonction effectue la traduction.

Arguments

docName, *siteRoot*, *docContent*

- L'argument *docName* est une chaîne qui contient l'URL de type file:// du document à traduire.
- L'argument *siteRoot* est une chaîne qui contient l'URL de type file:// de la racine du site où se trouve le document à traduire. Si le document se trouve en dehors d'un site, cette chaîne peut être vide.
- L'argument *docContent* est une chaîne qui indique le contenu du document.

Valeurs renvoyées

Soit une chaîne qui contient le document traduit, soit une chaîne vide si rien n'est traduit.

Exemple

L'instance suivante de la fonction `translateMarkup()` appelle la fonction C `translateASP()` contenue dans une DLL (Windows) ou dans une bibliothèque de code (Macintosh) appelée ASPTrans :

```
function translateMarkup(docName, siteRoot, docContent){
  var translatedString = "";
  if (docContent.length > 0){
  translatedString = ASPTrans.translateASP(docName, siteRoot, ¬
  docContent);
  }
  return translatedString;
}
```

Pour obtenir un exemple entièrement JavaScript, voir *Exemple de traducteur d'attributs simple*, page 463 ou *Exemple de traducteur de blocs/balises simple*, page 467.

liveDataTranslateMarkup()

Disponibilité

Dreamweaver UltraDev 1.

Description

Cette fonction traduit les documents lorsque l'utilisateur utilise la fenêtre Live Data. Lorsque l'utilisateur sélectionne la commande Affichage > Live Data ou clique sur le bouton Actualiser, Dreamweaver appelle la fonction `liveDataTranslateMarkup()` au lieu de la fonction `translateMarkup()`.

Arguments

`docName`, *`siteRoot`*, *`docContent`*

- L'argument *`docName`* est une chaîne qui contient l'URL de type file:// du document à traduire.
- L'argument *`siteRoot`* est une chaîne qui contient l'URL de type file:// de la racine du site où se trouve le document à traduire. Si le document se trouve en dehors d'un site, cette chaîne peut être vide.
- L'argument *`docContent`* est une chaîne qui indique le contenu du document.

Valeurs renvoyées

Soit une chaîne qui contient le document traduit, soit une chaîne vide si rien n'est traduit.

Exemple

L'instance suivante de la fonction `liveDataTranslateMarkup()` appelle la fonction C `translateASP()` contenue dans une DLL (Windows) ou dans une bibliothèque de code (Macintosh) appelée ASPTrans :

```
function liveDataTranslateMarkup(docName, siteRoot, docContent){
  var translatedString = "";
  if (docContent.length > 0){
  translatedString = ASPTrans.translateASP(docName, siteRoot, docContent);
  }
  return translatedString;
}
```

CHAPITRE 21

Extensions C

21

Les extensions C vous permettent d'implémenter des fichiers d'extension Macromedia Dreamweaver 8 en combinant du code JavaScript et votre propre code C. Créez des fonctions en langage C, intégrez-les à une DLL ou à une bibliothèque partagée, enregistrez la bibliothèque dans le sous-dossier Configuration/JSExtensions du dossier de l'application Dreamweaver, puis appelez ces fonctions à partir de JavaScript à l'aide de l'interpréteur JavaScript intégré à Dreamweaver.

Supposons, par exemple, que vous souhaitiez définir un objet Dreamweaver permettant d'insérer dans le document actif le contenu d'un fichier spécifié par l'utilisateur. Etant donné que le code JavaScript côté client ne prend pas en charge les E/S de fichier, vous devez créer une fonction en C pour fournir cette fonctionnalité.

Intégration des fonctions C

Vous pouvez utiliser le code HTML et JavaScript ci-dessous pour créer un objet simple d'insertion du texte d'un fichier. La fonction `objectTag()` appelle la fonction C `readContentsOfFile()`, stockée dans une bibliothèque appelée `maBibliothèque`.

```
<HTML>
<HEAD>
<SCRIPT>
function objectTag() {
  fileName = document.forms[0].myFile.value;
  return myLibrary.readContentsOfFile(fileName);
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Enter the name of the file to be inserted:
<INPUT TYPE="file" NAME="myFile">
</FORM>
```

```
</BODY>
</HTML>
```

La fonction `readContentsOfFile()` accepte une liste d'arguments de l'utilisateur, extrait l'argument de nom de fichier, lit le contenu du fichier et le renvoie. Pour plus d'informations sur les structures de données et les fonctions JavaScript apparaissant dans la fonction `readContentsOfFile()`, voir *Extensions C et interpréteur JavaScript*, page 479.

```
JSBool
readContentsOfFile(JSContext *cx, JSObject *obj, unsigned int ¬
argc, jsval *argv, jsval *rval)
{
  char *fileName, *fileContents;
  JSBool success;
  unsigned int length;

  /* Make sure caller passed in exactly one argument. If not,
  * then tell the interpreter to abort script execution.  */
  if (argc != 1){
    JS_ReportError(cx, "Wrong number of arguments", 0);
  return JS_FALSE;
  }

  /* Convert the argument to a string */
  fileName = JS_ValueToString(cx, argv[0], &length);
  if (fileName == NULL){
    JS_ReportError(cx, "The argument must be a string", 0);
    return JS_FALSE;
  }

  /* Use the string (the file name) to open and read a file */
  fileContents = exerciseLeftToTheReader(fileName);

  /* Store file contents in rval, which is the return value ¬
     passed
  * back to the caller */
  success = JS_StringToValue(cx, fileContents, 0, *rval);
  free(fileContents);

  /* Return true to continue or false to abort the script */
  return success;
}
```

Pour vous assurer que la fonction `readContentsOfFile()` s'exécute correctement et ne provoque pas d'erreur JavaScript, vous devez enregistrer la fonction avec l'interpréteur JavaScript en ajoutant une fonction `MM_Init()` à votre bibliothèque. Lorsque Dreamweaver charge la bibliothèque au démarrage, il appelle la fonction `MM_Init()` pour obtenir les trois informations suivantes :

- le nom JavaScript de la fonction ;

- un pointeur vers la fonction ;
- le nombre d'arguments attendus par la fonction.

L'exemple suivant illustre la fonction `MM_Init()` associée à la bibliothèque `maBibliothèque` :

```
void
MM_Init()
{
  JS_DefineFunction("readContentsOfFile", readContentsOfFile, 1);
}
```

Votre bibliothèque doit inclure exactement une instance de la macro suivante :

```
/* MM_STATE is a macro that expands to some definitions that are
 * needed to interact with Dreamweaver. This macro must
 * be defined exactly once in your library. */
MM_STATE
```

REMARQUE

La bibliothèque peut être implémentée en C ou en C++, mais le fichier qui contient la fonction `MM_Init()` et la macro `MM_STATE` doivent être implémentés en C. En effet, le compilateur C++ modifie les noms de fonction, ce qui rend impossible la détection de la fonction `MM_Init()` par Dreamweaver.

Extensions C et interpréteur JavaScript

Dans votre bibliothèque, le code C doit interagir avec l'interpréteur JavaScript de Dreamweaver à trois moments distincts :

- au démarrage, pour enregistrer les fonctions de la bibliothèque ;
- lorsque la fonction est appelée, pour analyser les arguments que JavaScript transmet à C ;
- avant le retour de la fonction, pour compresser la valeur renvoyée.

Pour accomplir ces tâches, l'interpréteur définit plusieurs types de données et affiche une API. Les définitions des types de données et des fonctions répertoriées dans cette section apparaissent dans le fichier mm_jsapi.h. Pour que votre bibliothèque fonctionne correctement, vous devez inclure le fichier mm_jsapi.h et la ligne suivante au début de tous les fichiers de votre bibliothèque :

```
#include "mm_jsapi.h"
```

L'inclusion du fichier mm_jsapi.h a pour effet d'inclure, à son tour, le fichier mm_jsapi_environment.h, qui définit la structure `MM_Environment`.

Types de données

L'interpréteur JavaScript définit les types de données suivants.

typedef struct JSContext JSContext

Un pointeur vers ce type de données opaque est transmis à la fonction C. Certaines des fonctions de l'API utilisent ce pointeur comme argument.

typedef struct JSObject JSObject

Un pointeur vers ce type de données opaque est transmis à la fonction C. Ce type de données représente un objet pouvant correspondre à un objet de tableau ou tout autre type d'objet.

typedef struct jsval jsval

Une structure de données opaque pouvant contenir un nombre entier ou un pointeur vers un nombre à virgule flottante, une chaîne ou un objet. Certaines des fonctions de l'API permettent de lire les valeurs des arguments de fonction à partir du contenu d'une structure `jsval` et d'autres permettent d'écrire la valeur renvoyée par la fonction en rédigeant une structure `jsval`.

typedef enum { JS_FALSE = 0, JS_TRUE = 1 } JSBool

Type de données simple qui stocke une valeur booléenne.

API d'extension C

L'API d'extension C se compose des fonctions suivantes :

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc, jsval *argv, jsval *rval)

Description

Cette signature de fonction décrit les implémentations C des fonctions JavaScript dans les situations suivantes :

- *cx* est un pointeur vers une structure `JSContext` opaque qui doit être transmis à certaines fonctions de l'API JavaScript. Cette variable contient le contexte d'exécution de l'interpréteur.
- *obj* est un pointeur vers l'objet dans le contexte duquel le script s'exécute. Pendant l'exécution du script, le mot clé `this` désigne cet objet.
- *argc* correspond au nombre d'arguments transmis à la fonction.
- *argv* est un pointeur vers un tableau de structures `jsval`. Le tableau comporte `argc` éléments en longueur.
- *rval* est un pointeur vers une structure `jsval` unique. La valeur renvoyée par la fonction doit être enregistrée dans `*rval`.

La fonction renvoie `JS_TRUE` si elle réussit ; `JS_FALSE` dans le cas contraire. Si la fonction renvoie la valeur `JS_FALSE`, l'exécution du script en cours s'arrête et un message d'erreur apparaît.

JSBool JS_DefineFunction()

Description

Cette fonction enregistre une fonction C avec l'interpréteur JavaScript dans Dreamweaver. Une fois que la fonction `JS_DefineFunction()` a enregistré la fonction C spécifiée dans l'argument *call*, vous pouvez l'appeler dans un script JavaScript en utilisant le nom spécifié dans l'argument *name*. L'argument `name` fait la distinction entre majuscules et minuscules.

Normalement, vous appelez cette fonction à partir de la fonction `MM_Init()`, que Dreamweaver appelle au démarrage.

Arguments

`char *name, JSNative call, unsigned int nargs`

- *name* est le nom de la fonction tel qu'il apparaît dans JavaScript.
- *call* est un pointeur vers une fonction C. La fonction doit accepter les mêmes arguments que `readContentsOfFile` et renvoyer une valeur `JSBool` qui indique si son exécution a réussi ou pas.
- *nargs* est le nombre d'arguments que la fonction doit recevoir.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

char *JS_ValueToString()

Description

Cette fonction extrait un argument de fonction à partir d'une structure `jsval`, le convertit en chaîne (si possible) et renvoie la valeur convertie à l'appelant.

REMARQUE

Ne modifiez pas le pointeur tampon renvoyé, car cela pourrait endommager les structures des données de l'interpréteur JavaScript. Pour modifier la chaîne, copiez les caractères dans un autre tampon et créez une chaîne JavaScript.

Arguments

`JSContext *cx, jsval v, unsigned int *pLength`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *v* est la structure `jsval` de laquelle est extraite la chaîne.
- *pLength* est un pointeur vers un nombre entier qui n'est pas signé. Cette fonction définit pour `*plength` une valeur égale à la longueur de la chaîne en octets.

Valeurs renvoyées

Pointeur vers une chaîne terminée par 0 en cas de réussite ou vers une valeur `null` en cas d'échec. La routine d'appel ne doit pas libérer cette chaîne lorsqu'elle a terminé.

JSBool JS_ValueToInteger()

Description

Cette fonction extrait un argument de fonction à partir d'une structure `jsval`, le convertit en nombre entier (si possible) et renvoie la valeur convertie à l'appelant.

Arguments

`JSContext *cx, jsval v, long *lp`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *v* est la structure `jsval` de laquelle l'entier doit être extrait.
- *lp* est un pointeur vers un nombre entier de 4 octets. Cette fonction stocke la valeur convertie dans `*lp`.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_ValueToDouble()

Description

Cette fonction extrait un argument de fonction d'une structure `jsval`, le convertit en réel double (si possible) et renvoie la valeur convertie à l'appelant.

Arguments

`JSContext *cx, jsval v, double *dp`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *v* est la structure `jsval` de laquelle le double est extrait.
- *dp* est un pointeur vers un réel double de 8 octets. Cette fonction stocke la valeur convertie dans `*dp`.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_ValueToBoolean()

Description

Cette fonction extrait un argument de fonction à partir d'une structure `jsval`, le convertit en valeur booléenne (si possible) et renvoie la valeur convertie à l'appelant.

Arguments

`JSContext *cx, jsval v, JSBool *bp`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *v* est la structure `jsval` de laquelle la valeur booléenne est extraite.
- *bp* est un pointeur vers une valeur booléenne `JSBool`. Cette fonction stocke la valeur convertie dans `*bp`.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_ValueToObject()

Description

Cette fonction extrait un argument de fonction à partir d'une structure `jsval`, le convertit en objet (si possible) et renvoie la valeur convertie à l'appelant. Si l'objet correspond à un tableau, utilisez `JS_GetArrayLength()` et `JS_GetElement()` pour lire son contenu.

Arguments

```
JSContext *cx, jsval v, JSObject **op
```

- *`cx`* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *`v`* est la structure `jsval` de laquelle l'objet est extrait.
- *`op`* est un pointeur vers un pointeur `JSObject`. Cette fonction stocke la valeur convertie dans `*op`.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_StringToValue()

Description

Cette fonction stocke la valeur renvoyée d'une chaîne dans une structure `jsval`. Elle alloue un nouvel objet de chaîne JavaScript.

Arguments

```
JSContext *cx, char *bytes, size_t sz, jsval *vp
```

- *`cx`* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- L'argument *`bytes`* est la chaîne à stocker dans la structure `jsval`. La chaîne est copiée de façon à ce que l'appelant puisse la libérer lorsqu'elle ne sera plus nécessaire. Si la taille de la chaîne n'est pas spécifiée (voir l'argument *`sz`*), la chaîne doit se terminer par un 0.
- *`sz`* indique la taille de la chaîne en octets. Si *`sz`* a pour valeur 0, la longueur de la chaîne se terminant par 0 est calculée automatiquement.
- *`vp`* est un pointeur vers la structure `jsval` dans laquelle le contenu de la chaîne doit être copié.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_DoubleToValue()

Description

Cette fonction stocke la valeur renvoyée d'un nombre à virgule flottante dans une structure `jsval`.

Arguments

`JSContext *cx, double dv, jsval *vp`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *dv* est un nombre à virgule flottante de 8 octets.
- *vp* est un pointeur vers la structure `jsval` dans laquelle le contenu du réel double doit être copié.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSVal JS_BooleanToValue()

Description

Cette fonction stocke dans une structure `jsval` la valeur booléenne renvoyée.

Arguments

`JSBool bv`

- *bv* est une valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

Valeurs renvoyées

Structure `JSVal` qui contient la valeur booléenne que vous avez transmise à la fonction en tant qu'argument.

JSVal JS_IntegerToValue()

Description

Cette fonction convertit une valeur d'entier long en structure `JSVal`.

Arguments

lv

- L'argument *lv* est un entier long que vous souhaitez convertir en structure `jsval`.

Valeurs renvoyées

Structure `JSVal` qui contient le nombre entier que vous avez transmis à la fonction en tant qu'argument.

JSVal JS_ObjectToValue()

Description

Cette fonction stocke la valeur renvoyée d'un objet dans une structure `JSVal`. Utilise la fonction `JS_ NewArrayObject()` pour créer un objet de tableau et utilise `JS_SetElement()` pour en définir le contenu.

Arguments

`JSObject *obj`

- *obj* est un pointeur vers l'objet `JSObject` que vous voulez convertir en structure `JSVal`.

Valeurs renvoyées

Structure `JSVal` qui contient l'objet que vous avez transmis à la fonction en tant qu'argument.

char *JS_ObjectType()

Description

A partir d'une référence d'objet, la fonction `JS_ObjectType()` renvoie le nom de la classe de l'objet. Ainsi, s'il s'agit d'un objet DOM, la fonction renvoie "`Document`". S'il s'agit d'un nœud dans le document, elle renvoie "`Element`". Dans le cas d'un objet de type tableau, elle renvoie "`Array`".

REMARQUE	Ne modifiez pas le pointeur tampon renvoyé, car cela pourrait endommager les structures des données de l'interpréteur JavaScript.

Arguments

`JSObject *obj`

- En principe, cet argument est transmis et converti à l'aide de `JS_ValueToObject()`.

Valeurs renvoyées

Pointeur vers une chaîne terminée par 0. L'appelant ne doit pas libérer cette chaîne une fois l'opération effectuée.

JSObject *JS_NewArrayObject()

Description

Cette fonction crée un nouvel objet contenant un tableau d'arguments `JSVal`.

Arguments

```
JSContext *cx, unsigned int length, jsval *v
```

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *length* correspond au nombre d'éléments que le tableau peut contenir.
- *v* est un pointeur facultatif vers les arguments `jsval` devant être stockés dans le tableau. Si la valeur renvoyée est différente de `null`, *v* correspond à un tableau contenant des éléments *length*. Si la valeur renvoyée est `null`, le contenu initial de l'objet de tableau est indéterminé (et peut être défini à l'aide de la fonction `JS_SetElement()`).

Valeurs renvoyées

Un pointeur vers un nouvel objet de tableau (array) ou la valeur `null` en cas d'échec de l'exécution de la fonction.

long JS_GetArrayLength()

Description

A partir d'un pointeur vers un objet de tableau, extrait le nombre d'éléments contenus dans le tableau.

Arguments

```
JSContext *cx, JSObject *obj
```

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *obj* est un pointeur vers un objet de tableau.

Valeurs renvoyées

Soit le nombre d'éléments figurant dans le tableau, soit -1 en cas d'échec d'exécution de la fonction.

JSBool JS_GetElement()

Description

Cette fonction lit un seul élément d'un objet de tableau.

Arguments

`JSContext *cx, JSObject *obj, unsigned int index, jsval *v`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *obj* est un pointeur vers un objet de tableau.
- *index* est un index des nombres entiers du tableau. Le premier élément est l'index 0, le dernier élément est l'index (`length - 1`).
- *v* est un pointeur vers une structure `jsval` dans laquelle sera copié le contenu de la structure `jsval` figurant dans le tableau.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_SetElement()

Description

Cette fonction écrit un seul élément d'un objet de tableau.

Arguments

*JSContext *cx, JSObject *obj, unsigned int index, jsval *v*

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *obj* est un pointeur vers un objet de tableau.
- *index* est un index des nombres entiers du tableau. Le premier élément est l'index `0`, le dernier élément est l'index (`length - 1`).
- *v* est un pointeur vers une structure `jsval` dont le contenu doit être copié dans la structure `jsval` du tableau.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_ExecuteScript()

Description

Cette fonction compile et exécute une chaîne JavaScript. Si le script renvoie une valeur, celle-ci apparaît dans `*rval`.

Arguments

`JSContext *cx`, `JSObject *obj`, `char *script`, `unsigned int sz`, `jsval *rval`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *obj* est un pointeur vers l'objet dans le contexte duquel le script s'exécute. Pendant l'exécution du script, le mot clé `this` désigne cet objet. En général, il s'agit du pointeur `JSObject` qui a été transmis à la fonction JavaScript.
- *script* est une chaîne qui contient le code JavaScript. Si la taille de la chaîne n'est pas spécifiée (voir l'argument *sz*), la chaîne doit se terminer par un 0.
- *sz* indique la taille de la chaîne en octets. Si sz a pour valeur 0, la longueur de la chaîne se terminant par 0 est calculée automatiquement.
- *rval* est un pointeur vers une seule structure `jsval`. La valeur renvoyée pour la fonction est stockée dans `*rval`.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

JSBool JS_ReportError()

Description

Cette fonction décrit la cause d'une erreur de script. Appelez cette fonction avant de renvoyer la valeur `JS_FALSE` pour une erreur de script, pour donner à utilisateur la raison de l'échec du script (par exemple « wrong number of arguments »).

Arguments

`JSContext *cx`, `char *error`, `size_t sz`

- *cx* est le pointeur `JSContext` opaque transmis à la fonction JavaScript.
- *error* est une chaîne qui contient le message d'erreur. La chaîne est copiée de façon à ce que l'appelant puisse la libérer lorsqu'elle ne sera plus nécessaire. Si la taille de la chaîne n'est pas spécifiée (voir l'argument *sz*), la chaîne doit se terminer par un 0.
- *sz* indique la taille de la chaîne en octets. Si sz a pour valeur 0, la longueur de la chaîne se terminant par 0 est calculée automatiquement.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec.

API de configuration multiutilisateur et d'accès aux fichiers

Macromedia vous recommande de toujours utiliser l'API de configuration multiutilisateur et d'accès aux fichiers pour accéder au système de fichiers par l'intermédiaire d'extensions C. Pour les fichiers autres que les fichiers de configuration, les fonctions accèdent directement au fichier ou dossier spécifié.

Dreamweaver prend en charge les configurations multiutilisateur pour les systèmes Windows XP, Windows 2000 et Mac OS X.

Dreamweaver est généralement installé dans un dossier à accès restreint, tel que C:\Program Folders sous Windows. De ce fait, seuls les utilisateurs disposant de droits d'accès de type administrateur peuvent modifier le dossier Configuration de Dreamweaver. Pour permettre aux utilisateurs travaillant dans des environnements multiutilisateur de créer et de gérer des configurations distinctes, Dreamweaver crée un dossier de configuration séparé pour chacun d'entre eux. Chaque fois que Dreamweaver ou une extension JavaScript écrit dans le dossier Configuration de Dreamweaver, Dreamweaver écrit automatiquement dans le dossier de configuration de l'utilisateur à la place. Ainsi, Dreamweaver permet à chaque utilisateur de personnaliser les paramètres de configuration respectifs, sans aucune incidence sur les configurations personnalisées des autres utilisateurs.

Dreamweaver crée le dossier de configuration de l'utilisateur à un emplacement dans lequel l'utilisateur dispose de tous les droits d'accès en lecture/écriture. L'emplacement du dossier Configuration varie selon la plate-forme utilisateur.

Sous Windows 2000 et Windows XP :

```
<drive>:\Documents and Settings\<username>\ ¬
  Application Data\Macromedia\Dreamweaver 8\Configuration
```

REMARQUE Sous Windows XP, ce dossier peut se trouver dans un dossier masqué.

Pour Mac OS X :

```
<drive>:Users:<username>:Library:Application Support: ¬
  Macromedia:Dreamweaver 8:Configuration
```

Il est fréquent que les extensions JavaScript soient amenées à ouvrir des fichiers et à écrire dans le dossier Configuration.. Elles peuvent accéder au système de fichiers en utilisant DWFile et MMNotes ou en transmettant une URL à la fonction `dreamweaver.getDocumentDOM()`. En général, lorsqu'une extension accède au système de fichiers dans un dossier de configuration, elle utilise la fonction `dw.getConfigurationPath()` et ajoute le nom du fichier, ou elle obtient le chemin d'accès en se reportant à l'élément `dom.URL` d'un document ouvert et en ajoutant le nom du fichier. Une extension peut également obtenir le chemin d'accès en se reportant à l'élément `dom.URL` et en extrayant le nom du fichier. La fonction `dw.getConfigurationPath()` et l'élément `dom.URL` renvoient toujours une URL du dossier Configuration de Dreamweaver, même si le document se trouve dans le dossier de configuration de l'utilisateur.

Chaque fois qu'une extension JavaScript ouvre un fichier dans le dossier Configuration de Dreamweaver, ce dernier interrompt l'accès et vérifie d'abord le dossier de configuration de l'utilisateur. Si une extension JavaScript enregistre des données sur le disque dans le dossier Configuration de Dreamweaver via DWFile ou MMNotes, Dreamweaver intercepte l'appel et le redirige vers le dossier de configuration de l'utilisateur.

Sous Windows 2000 ou Windows XP par exemple, si l'utilisateur demande "`file:///C|/Program Files/Macromedia/Dreamweaver/Configuration/Objects/Common/Table.htm`", Dreamweaver recherche d'abord un fichier Table.htm dans le dossier C:/Documents et Settings/*nom d'utilisateur*/Macromedia/Dreamweaver/Configuration/Objects/Common, et s'il existe, l'utilise.

Les extensions C, ou les bibliothèques partagées, doivent utiliser l'API de configuration multiutilisateur et d'accès aux fichiers pour lire et écrire dans le dossier Configuration de Dreamweaver. Le recours à cette API permet à Dreamweaver de lire et d'écrire dans le dossier de configuration de l'utilisateur et garantit que les opérations liées aux fichiers n'échouent pas en raison de droits d'accès insuffisants. Si votre extension C accède à des fichiers du dossier Configuration de Dreamweaver créés par modification de DWFile, MMNotes ou DOM au moyen de JavaScript, vous devez impérativement utiliser cette API, car les fichiers risquent de résider dans le dossier de configuration de l'utilisateur.

REMARQUE La plupart des extensions JavaScript ne requièrent aucune modification pour pouvoir écrire dans le dossier de configuration de l'utilisateur. Seules les bibliothèques C partagées qui écrivent dans le dossier de configuration doivent être mises à jour afin d'utiliser les fonctions de l'API de configuration multiutilisateur et d'accès aux fichiers.

Lorsque vous supprimez un fichier du dossier Configuration de Dreamweaver, ce dernier ajoute une entrée à un fichier masque pour indiquer les fichiers du dossier qui ne doivent pas apparaître dans l'interface utilisateur. Un fichier ou dossier masqué n'existera pas pour Dreamweaver, même s'il peut exister physiquement dans le dossier.

Supposons, par exemple, que vous ayez utilisé la corbeille du panneau Fragments de code pour supprimer un dossier appelé javascript et un fichier appelé onepixelborder.csn. Dans ce cas, Dreamweaver écrit un fichier nommé mm_deleted_files.xml dans le dossier de configuration de l'utilisateur, ressemblant à ceci :

```
<?xml version = "1.0" encoding="utf-8" ?>
  <deleteditems>
  <item name="snippets/javascript/" />
  <item name="snippets/html/onepixelborder.csn" />
  </deleteditems>
```

Pour afficher le panneau Fragments de code, Dreamweaver lit tous les fichiers du dossier Configuration/Snippets de l'utilisateur ainsi que ceux du dossier Configuration/Snippets de Dreamweaver, à l'exception du dossier Configuration/Snippets/javascript et du fichier Configuration/Snippets/html/onepixelborder.csn, puis ajoute la liste de fichiers ainsi obtenue au panneau Fragments de code.

Si une extension C appelle la fonction `MM_ConfigFileExists()` de l'URL "file:///c|Program Files/Macromedia/Dreamweaver/Configuration/Snippets/javascript/onepixelborder.csn", cette dernière renvoie la valeur `false`. De même, si une extension JavaScript essaie d'appeler la fonction `dw.getDocumentDom("file:///c|Program Files/Macromedia/Dreamweaver/Configuration/Snippets/javascript/onepixelborder.csn")`, la valeur renvoyée est `null`.

Vous pouvez modifier le fichier mm_deleted_files.xml pour empêcher Dreamweaver d'afficher des fichiers dans l'interface utilisateur, tels que des objets ou un contenu prêt à l'emploi dans une nouvelle boîte de dialogue. Vous pouvez appeler la fonction `MM_DeleteConfigfile()` pour ajouter des chemins de fichier au fichier mm_deleted_files.xml.

JS_Object MM_GetConfigFolderList()

Disponibilité

Dreamweaver MX.

Description

Cette fonction extrait une liste répertoriant les dossiers ou les fichiers (ou les deux à la fois) du dossier spécifié. Si vous spécifiez un dossier de configuration, la fonction obtient la liste des dossiers qui existent dans le dossier de configuration de l'utilisateur ainsi que dans celui de Dreamweaver, après filtrage par mm_deleted_files.xml.

Arguments

`char *fileURL, char *constraints`

- *char *fileUrl* est un pointeur vers une chaîne qui nomme le dossier dont vous souhaitez obtenir le contenu. La chaîne doit se présenter sous la forme d'une URL de fichier (file:///). Cette chaîne peut comporter les caractères génériques suivants : astérisques (*) et points d'interrogation (?). Utilisez les astérisques (*) pour représenter un ou plusieurs caractères indéterminés et les points d'interrogation (?) pour représenter un seul caractère indéterminé.
- L'argument *char *contstraints* peut être "files" ou "directories" ou encore une valeur null. Si vous précisez null, la fonction MM_GetConfigFolderList() renvoie des fichiers et des dossiers.

Valeurs renvoyées

JSObject est un tableau qui contient la liste des fichiers ou des dossiers figurant soit dans le dossier de configuration de l'utilisateur, soit dans le dossier Configuration de Dreamweaver, après filtrage par le fichier mm_deleted_files.xml.

Exemples

```
JSObject *jsobj_array;
jsobj_array = MM_GetConfigFolderList("file:///¬
  c|/Program Files/Macromedia/Dreamweaver/Configuration", "directories" );
```

JSBool MM_ConfigFileExists()

Disponibilité

Dreamweaver MX.

Description

Cette fonction vérifie si le fichier spécifié existe. S'il s'agit d'un fichier dans un dossier de configuration, la fonction recherche le fichier dans le dossier Configuration de l'utilisateur ou celui de Dreamweaver. Elle vérifie également si le nom du fichier est répertorié dans le fichier mm_deleted_files.xml. Si tel est le cas, la fonction renvoie la valeur false.

Arguments

*char *fileUrl*

- *char *fileUrl* est un pointeur vers une chaîne qui nomme le fichier souhaité, sous la forme d'une URL de type file://.

Valeurs renvoyées

Valeur booléenne : JS_TRUE en cas de réussite ; JS_FALSE en cas d'échec.

Exemple

```
char *dwConfig = “file:///c|/Program Files/Macromedia/Dreamweaver/
  Configuration/Extensions.txt”;
int fileno = 0;
if(MM_ConfigFileExists(dwConfig))
{
  fileno = MM_OpenConfigFile(dwConfig, “read”);
}
```

int MM_OpenConfigFile()

Disponibilité

Dreamweaver MX.

Description

Cette fonction ouvre le fichier et renvoie son identificateur dans le système d'exploitation. Vous pouvez utiliser cet identificateur dans des appels à des fonctions de fichiers système. Vous devez fermer l'identificateur de fichier en appelant la fonction `_close`.

S'il s'agit d'un fichier de configuration, la fonction le recherche soit dans le dossier de configuration de l'utilisateur, soit dans le dossier Configuration de Dreamweaver. Si vous ouvrez le fichier de configuration à des fins d'écriture, la fonction crée le fichier dans le dossier de configuration de l'utilisateur, même s'il existe dans le dossier Configuration de Dreamweaver.

REMARQUE

Si vous souhaitez commencer par lire le fichier, ouvrez-le en mode `"read"`. Lorsque vous souhaitez écrire dans le fichier, fermez l'identificateur de lecture et rouvrez le fichier en mode `"write"` ou `"append"`.

Arguments

`char *fileURL, char *mode`

- *`char *fileURL`* est un pointeur vers une chaîne qui nomme le fichier en cours d'ouverture, exprimé sous la forme d'une URL de fichier (file:///). S'il indique un chemin d'accès dans le dossier Configuration de Dreamweaver, la fonction `MM_OpenConfigFile()` commence par déterminer le chemin d'accès avant d'ouvrir le fichier.

- *char *mode* pointe vers une chaîne qui indique la façon dont vous souhaitez ouvrir le fichier. Vous avez le choix entre null, "read", "write" et "append". Si vous spécifiez "write" et que le fichier n'existe pas, la fonction MM_OpenconfigFile() le crée. Si vous spécifiez "write", la fonction MM_OpenConfigFile() ouvre le fichier en mode de partage exclusif. Si vous spécifiez "read", MM_OpenConfigFile() ouvre le fichier en mode de partage non exclusif.

  Si vous ouvrez le fichier en mode "write", toutes les données existant dans le fichier sont tronquées avant que les nouvelles données ne soient écrites. Si vous ouvrez le fichier en mode "append", toutes les nouvelles données sont ajoutées à la fin du fichier.

Valeurs renvoyées

Nombre entier représentant l'identificateur du fichier dans le système d'exploitation. Renvoie la valeur -1 si le fichier est introuvable ou n'existe pas.

Exemple

```
char *dwConfig = "file:///c|/Program Files/Macromedia/Dreamweaver/
  Configuration/Extensions.txt";
int = fileno;
if(MM_ConfigFileExists(dwConfig))
{
  fileno = MM_OpenConfigFile(dwConfig, "read");
}
```

JSBool MM_GetConfigFileAttributes()

Disponibilité

Dreamweaver MX.

Description

Cette fonction trouve le fichier et en renvoie les attributs. Vous pouvez définir tous les arguments sur *null*, à l'exception de fileURL, si vous n'avez pas besoin de leur valeur.

Arguments

```
char *fileURL, unsigned long *attrs, unsigned long *filesize,
  unsigned long *modtime, unsigned long *createtime
```

- *char *fileURL* est un pointeur vers une chaîne qui nomme le fichier dont vous voulez les arguments. Vous devez fournir cet argument sous forme de fichier (file:// URL). Si *fileURL* indique un chemin d'accès dans le dossier Configuration de Dreamweaver, la fonction MM_GetConfigFileAttributes() commence par déterminer le chemin d'accès à utiliser avant d'ouvrir le fichier.

- *unsigned long *attrs* représente l'adresse d'un nombre entier contenant les bits d'attribut renvoyés (voir JSBool MM_SetConfigFileAttributes() pour la liste des attributs disponibles).
- *unsigned long *filesize* est l'adresse d'un entier dans lequel la fonction renvoie la taille du fichier en octets.
- *unsigned long *modtime* est l'adresse d'un entier dans lequel la fonction renvoie l'heure à laquelle le fichier a été modifié pour la dernière fois. Cette heure correspond à la valeur horaire du système d'exploitation. Pour plus d'informations sur la valeur horaire du système d'exploitation, voir `DWfile.getModificationDate()` dans le *Guide des API de Dreamweaver*.
- *unsigned long *createtime* est l'adresse d'un entier dans lequel la fonction renvoie l'heure à laquelle le fichier a été créé. Cette heure correspond à la valeur horaire du système d'exploitation. Pour plus d'informations sur la valeur horaire du système d'exploitation, voir `DWfile.getCreationDate()` dans le *Guide des API de Dreamweaver*.

Valeurs renvoyées

Valeur booléenne : `JS_TRUE` en cas de réussite ; `JS_FALSE` en cas d'échec. Renvoie la valeur `JS_FALSE` si le fichier n'existe pas ou si une erreur se produit pendant l'obtention des attributs.

Exemple

```
char dwConfig = "file:///c|/Program Files/Macromedia/Dreamweaver/
  Configuration/Extensions.txt";
unsigned long attrs;
unsigned long filesize;
unsigned long modtime;
unsigned long createtime;
MM_GetConfigAttributes(dwConfig, &attrs, &filesize, &modtime, &createtime);
```

JSBool MM_SetConfigFileAttributes()

Disponibilité

Dreamweaver MX.

Description

Cette fonction applique les attributs que vous spécifiez pour le fichier, s'ils diffèrent des attributs actuels.

Si l'URL de fichier spécifiée décrit un chemin d'accès au dossier Configuration de Dreamweaver, cette fonction commence par copier le fichier dans le dossier de configuration de l'utilisateur avant d'appliquer les attributs. Si les attributs sont identiques aux attributs actuels du fichier, ce dernier n'est pas copié.

Arguments

*char *fileURL, unsigned long attrs*

- *char *fileURL* est un pointeur vers une chaîne qui nomme le fichier dont vous souhaitez définir les attributs, exprimé sous la forme d'une URL de fichier (file://).
- *unsigned long attrs* spécifie les bits d'attribut à appliquer au fichier. Vous pouvez utiliser l'opérateur logique OR dans les constantes suivantes pour définir les attributs :

  ```
  MM_FILEATTR_NORMAL
  MM_FILEATTR_RDONLY
  MM_FILEATTR_HIDDEN
  MM_FILEATTR_SYSTEM
  MM_FILEATTR_SUBDIR
  ```

Valeurs renvoyées

Valeur booléenne : JS_TRUE en cas de réussite ; JS_FALSE en cas d'échec. Renvoie JS_FALSE si le fichier n'existe pas ou s'il va être supprimé.

Exemple

```
char *dwConfig = "file:///c|/Program Files/Macromedia/Dreamweaver/
  Configuration/Extensions.txt";
unsigned long attrs;
attrs = (MM_FILEATTR_NORMAL | MM_FILEATTR_RDONLY);
int fileno = 0;
if(MM_SetConfigFileAttrs(dwConfig, attrs))
{
  fileno = MM_OpenConfigFile(dwConfig);
}
```

JSBool MM_CreateConfigFolder()

Disponibilité

Dreamweaver MX.

Description

Cette fonction crée un dossier à l'emplacement spécifié.

Si l'argument *fileURL* indique un dossier dans le dossier Configuration de Dreamweaver, la fonction crée le dossier dans le dossier de configuration de l'utilisateur. Si *fileURL* n'indique pas un sous-dossier du dossier Configuration de Dreamweaver, la fonction crée le dossier ainsi que tous ses dossiers parents dans le chemin d'accès, si ceux-ci n'existent pas déjà.

Arguments

*char *fileURL*

- *char *fileURL* est un pointeur vers une chaîne qui nomme le dossier de configuration à créer, exprimé sous la forme d'une URL de fichier (file://).

Valeurs renvoyées

Valeur booléenne : JS_TRUE en cas de réussite ; JS_FALSE en cas d'échec.

Exemple

```
char *dwConfig = "file:///c|/Program Files\Macromedia\Dreamweaver
  \Configuration\Extensions.txt";
MM_CreateConfigFolder(dwConfig);
```

JSBool MM_RemoveConfigFolder()

Disponibilité

Dreamweaver MX.

Description

Cette fonction supprime le dossier, ainsi que ses fichiers et sous-dossiers. Si le sous-dossier réside dans le dossier Configuration de Dreamweaver, elle marque le dossier pour suppression dans le fichier mm_deleted_files.xml.

Arguments

*char *fileURL*

- *char *fileURL* est un pointeur vers une chaîne qui nomme le dossier à supprimer, exprimé sous la forme d'une URL de fichier (file://).

Valeurs renvoyées

Valeur booléenne : JS_TRUE en cas de réussite ; JS_FALSE en cas d'échec.

Exemple

```
char *dwConfig = "file:///c|/Program Files\Macromedia\Dreamweaver
  \Configuration\Objects";
MM_RemoveConfigFolder(dwConfig);
```

JSBool MM_DeleteConfigFile()

Disponibilité

Dreamweaver MX.

Description

Cette fonction supprime le fichier le cas échéant. Si le fichier réside à un niveau inférieur au dossier Configuration de Dreamweaver, la fonction marque le fichier pour suppression dans le fichier mm_deleted_files.xml.

Si l'argument *fileURL* n'indique pas de dossier dans le dossier Configuration de Dreamweaver, la fonction supprime le fichier spécifié.

Arguments

*char *fileURL*

- *char *fileURL* est un pointeur vers une chaîne qui nomme le dossier de configuration à supprimer, exprimé sous la forme d'une URL de fichier (file://).

Valeurs renvoyées

Valeur booléenne : JS_TRUE en cas de réussite ; JS_FALSE en cas d'échec.

Exemple

```
char dwConfig = "file:///c:|Program Files\Macromedia\Dreamweaver
  \Configuration\Objects\insertbar.xml";
MM_DeleteConfigFile(dwConfig);
```

Appel d'une fonction C à partir de JavaScript

A présent que vous savez comment fonctionnent les extensions C dans Dreamweaver et que vous connaissez les fonctions et les types de données sur lesquels elles reposent, vous allez apprendre à construire une bibliothèque et à appeler une fonction.

L'exemple qui suit nécessite les cinq fichiers suivants situés dans le dossier de l'application Dreamweaver Samples/Extending sous forme d'archives, pour Macintosh et Windows :

- mm_jsapi.h est un fichier d'en-tête contenant les définitions des fonctions et des types de données décrits dans *Extensions C et interpréteur JavaScript*, page 479.
- Le fichier mm_jsapi_environment.h définit la structure MM_Environment.h.
- Le fichier MMInfo.h fournit un accès à l'API de Design Notes.
- Le fichier Sample.c example définit la fonction computeSum().

- Le fichier Make Sample.mak permet de créer le fichier Sample.c source dans une DLL avec Microsoft Visual C++. Sample.mcp est un fichier équivalent destiné à créer un bundle Mach-O par le biais de Metrowerks CodeWarrior et Sample.xcode est un fichier équivalent pour Apple Xcode. Si vous utilisez un autre outil, vous pouvez créer votre propre fichier Make.

Pour créer la DLL sous Windows avec VS.Net 2003 :

1. Sélectionnez Fichier > Ouvrir > Sample.mak en réglant le type de fichier sur Tous les fichiers (*.*). (VS.Net 2003 n'ouvre pas directement les fichiers .mak.) Vous êtes invité à confirmer que vous souhaitez convertir le projet au nouveau format.
2. Choisissez Générer > Générer la solution.

   Une fois l'opération de génération terminée, le fichier Sample.dll apparaît dans le dossier qui contient Sample.mak (ou dans l'un de ses sous-dossiers).

Pour créer la DLL sous Windows avec Microsoft Visual C++ :

1. Dans Microsoft Visual C++, choisissez Fichier >Ouvrir un espace de travail, puis sélectionnez Sample.mak.
2. Choisissez Générer > Tout reconstruire.

   Une fois l'opération de génération terminée, le fichier Sample.dll apparaît dans le dossier qui contient Sample.mak (ou dans l'un de ses sous-dossiers).

Pour créer la bibliothèque partagée sur le Macintosh par le biais de Metrowerks CodeWarrior 9 ou ultérieur :

1. Ouvrez Sample.mcp.
2. Compilez le projet (Projet > Make) pour générer un bundle Mach-O.

   Une fois la compilation terminée, le fichier Sample.bundle apparaît dans le dossier qui contient Sample.mcp.

REMARQUE	Le bundle Mach-O Bundle généré ne peut être utilisé que par Macromedia Dreamweaver 8. Il n'est en effet pas reconnu par les versions antérieures de Dreamweaver.

Pour créer la bibliothèque partagée sur le Macintosh par le biais de Apple Xcode 1.5 ou ultérieur :

1. Ouvrez Sample.xcode.
2. Compilez le projet (Build > Build) pour générer un bundle Mach-O.

Une fois la compilation terminée, le fichier Sample.bundle apparaît dans le dossier de compilation en regard du fichier Sample.xcode.

REMARQUE	Le bundle Mach-O Bundle généré ne peut être utilisé que par Macromedia Dreamweaver 8. Il n'est en effet pas reconnu par les versions antérieures de Dreamweaver.

Pour appeler la fonction computeSum() à partir de l'objet Insérer barre horizontale :

1. Dans le dossier Configuration du dossier de l'application Dreamweaver, créez un dossier appelé JSExtensions.
2. Copiez le fichier Sample.dll (Windows) ou Sample.bundle (Macintosh) dans le dossier JSExtensions.
3. Dans un éditeur de texte, ouvrez le fichier HR.htm qui se trouve dans le dossier Configuration/Objects/Common.
4. Ajoutez la ligne `alert(Sample.computeSum(2,2));` à la fonction `objectTag()` pour qu'elle apparaisse, comme indiqué dans l'exemple ci-dessous :

```
function objectTag() {
  // Return the html tag that should be inserted
  alert(Sample.computeSum(2,2));
  return "<HR>";
}
```

5. Enregistrez le fichier et redémarrez Dreamweaver.

Pour exécuter la fonction `computeSum()`, sélectionnez Insertion > HTML>Barre horizontale.

Une boîte de dialogue contenant le chiffre 4 (résultat de la somme de 2 plus 2) s'affiche.

PARTIE 4

Appendix

4

Find information about supporting files and reference resources that can aid in developing Macromedia Dreamweaver 8 extensions.

ANNEXE

Dossier Shared

Le dossier Shared est le lieu de stockage central des fonctions utilitaires, des classes et des images utilisées communément par toutes les extensions. Toutes les extensions peuvent référencer les fichiers dans les sous-dossiers du dossier Shared. Vous pouvez également ajouter des utilitaires communs à ceux déjà fournis par Macromedia Dreamweaver 8. Les dossiers Configuration multiutilisateur installés pour les utilisateurs sous Windows XP, Windows 2000 et Macintosh OS X contiennent également un dossier Shared autorisant les personnalisations. Par exemple, lorsque vous installez une extension à partir de Macromedia Exchange, vous remarquerez peut-être que la nouvelle extension ajoute des données à votre dossier Configuration/Shared utilisateur plutôt qu'au dossier Configuration/Shared de l'application Dreamweaver. Pour plus d'informations sur les dossiers Configuration de Dreamweaver sur un ordinateur multiutilisateur, voir *Dossiers de configuration multiutilisateur*, page 111.

Contenu du dossier Shared

Le dossier Shared est divisé en sous-dossiers qui contiennent des fichiers partagés par plusieurs extensions, notamment des fonctions permettant entre autres de naviguer dans le système de dossiers des utilisateurs, d'insérer une commande d'arborescence et de créer des grilles modifiables.

REMARQUE

Les fichiers JavaScript du dossier Shared contiennent des commentaires dans le code qui fournissent des détails sur les fonctions qu'ils contiennent.

En plus de consulter les fichiers JavaScript dans le dossier Shared, consultez également les fichiers HTML dans le dossier Configuration contenant ces fichiers JavaScript. Ils vous fourniront des explications sur l'utilisation de ces fichiers.

En règle générale, vous serez amené à utiliser les fonctions et ressources des dossiers Common et Macromedia (MM) ou à ajouter des ressources dans le dossier Common pour les utiliser dans de nouvelles extensions. Commencez toujours par rechercher les utilitaires et les fonctions dans le dossier Shared/Common/Scripts. Ces fonctions et utilitaires sont les plus récents et forment l'interface formelle avec le dossier Shared. Les fichiers des autres dossiers risquent quant à eux de ne pas être à jour.

Le dossier Shared contient plus spécifiquement les dossiers présentés ci-dessous.

Dossier Common

Le dossier Common contient des scripts et des classes partagés à utiliser dans les extensions de tiers.

CodeBehindMgr.js	Contient les fonctions permettant de créer un document code-behind. Un document code-behind permet de créer des pages distinctes qui séparent le code relatif à la logique de l'interface utilisateur (UI) du code relatif à une conception d'interface utilisateur. Les méthodes de `JSCodeBehindMgr` définies dans ce fichier permettent de créer de nouveaux documents code-behind et de gérer le lien vers les documents de conception.
ColumnValueNodeClass.js	Contient des fonctions permettant d'associer les colonnes des bases de données à des valeurs. Les méthodes de `ColumnValueNode` définies dans ce fichier permettent d'obtenir et de définir plusieurs valeurs et propriétés d'une colonne de base de données. Dreamweaver utilise cette classe de stockage lorsqu'il applique et inspecte des objets d'opérations de modification (insertion et mise à jour d'enregistrements) et lorsqu'il utilise la classe SQLStatement.
CompilerClass.js	Contient des fonctions pour une classe de base utilisée par `CompilerASPNetCSharp` et `CompilerASPNVBNet` mais peut être étendu de façon à prendre en charge d'autres compilateurs.
DataSourceClass.js	Contient des fonctions qui définissent la structure de renvoi pour findDynamicSources().

DBTreeControlClass.js	Contient des fonctions qui créent une commande d'arborescence de base de données. Cette classe permet de créer et d'interagir avec une commande d'arborescence de base de données. Pour créer une commande d'arborescence de base de données comme celle des comportements de serveurs de jeux d'enregistrements avancés, créez une liste `<select>` avec `type="mmdatabasetree"` dans votre fichier HTML. Joignez une classe `CBTreeControl` à la commande HTML en transmettant le nom de la liste `<select>` à l'élaborateur de la classe. Utilisez ensuite les fonctions `DBTreeControl` pour manipuler la commande.
dotNetUtils.js	Contient des fonctions permettant de faciliter l'utilisation des inspecteurs de propriétés des objets et des comportements de serveur pour les contrôles de formulaires ASP.NET qui sont traduits.
dwscripts.js	Vous trouverez dans le fichier principal des fonctions utiles à toutes les extensions Dreamweaver. Il comprend des fonctions permettant d'utiliser des chaînes, des fichiers, des Design Notes, etc.
dwscriptsExtData.js	Ce fichier est une extension du fichier dwscripts.js. Il facilite l'utilisation des comportements de serveur, en particulier avec les fichiers EDML de comportements de serveurs. Utilisé très largement dans l'implémentation des comportements de serveur dans Dreamweaver.
dwscriptsServer.js	Ce fichier est une extension du fichier dwscripts.js. Il contient des fonctions spécifiques aux modèles de serveur. Un grand nombre de ces fonctions sont utilisées pour travailler avec les comportements de serveur.
GridControlClass.js	Utilisez cette classe pour créer et manipuler une grille modifiable. Vous devez ajouter une liste de sélection spéciale à votre page HTML et y joindre cette classe dans JavaScript pour manipuler la grille.
ImageButtonClass.js	Cette classe facilite le contrôle de l'aspect du bouton : pression, survol de souris pendant pression, survol de souris et désactivé pendant pression.
ListControlClass.js	Contient les fonctions permettant de gérer une balise `<select>` ›. Egalement appelé contrôle de liste. Les méthodes de l'objet `ListControl` dans ce fichier permettent d'obtenir, de définir et de changer la valeur de la commande `SELECT`.
PageSettingsASPNet.js	Contient des fonctions qui définissent les propriétés d'un document ASP .NET.

RadioGroupClass.js	Contient les fonctions permettant de définir et de gérer un groupe de boutons radio. Les méthodes de l'objet `RadioGroup` de ce fichier permettent de définir et d'obtenir les valeurs et le comportement d'un groupe de boutons radio. Vous devez joindre cette classe aux boutons radio dans votre document HTML pour contrôler leur comportement.
SBDatabaseCallClass.js	Sous-classe de la classe `ServerBehavior`. Cette classe comprend des fonctionnalités spécifiques à la création d'appels de bases de données, par exemple l'appel d'une procédure stockée, l'utilisation de SQL pour renvoyer un jeu d'enregistrements, etc. Il s'agit d'une classe de base abstraite, ce qui signifie qu'elle ne peut pas être créée et utilisée seule. Pour l'utiliser, vous devez définir `SBDatabaseCall()` comme sous-classe et implémenter les fonctions d'espace réservé. Dreamweaver utilise cette classe pour implémenter ses comportements de serveur de jeux d'enregistrements et de procédures stockées.
ServerBehaviorClass.js	Contient des fonctions permettant de communiquer des informations concernant les comportements de serveur à Dreamweaver. Vous pouvez définir cette classe en sous-classe lorsque vous implémentez vos comportements de serveur.
ServerSettingsASPNet.js	Contient des fonctions permettant de stocker les propriétés d'un serveur ASP .NET.
SQLStatementClass.js	Contient des fonctions permettant de créer et de modifier des instructions SQL telles que `SELECT`, `INSERT`, `UPDATE`, `DELETE` et des instructions de procédures stockées.
tagDialogsCmn.js	Contient des fonctions visant à vous aider à développer des boîtes de dialogue de balise personnalisées. Les méthodes de l'objet `tagDialog` définies dans ce fichier modifient les attributs et les valeurs d'une balise particulière.
TagEditClass.js	Contient des fonctions permettant de modifier des balises sans changer le DOM de la page en cours. Les méthodes de l'objet `TagEdit` définies dans ce fichier permettent d'obtenir et de définir la valeur, les attributs et les enfants d'une balise. Cette classe est utile pour créer des modifications complexes, le DOM ne devenant pas obsolète.

TreeControlClass.js	Contient des fonctions permettant de gérer une commande d'arborescence dans Dreamweaver. Les méthodes de l'objet `TreeControl` définies dans ce fichier permettent d'obtenir, de définir et d'organiser les valeurs dans une arborescence. Vous devez joindre cette classe à une balise `MM:TREECONTROL` spéciale dans votre page HTML pour gérer la fonctionnalité de la commande d'arborescence.
XMLPropSheetClass.js	Contient des fonctions permettant de gérer l'emplacement et les valeurs d'une fiche de propriétés XML.

Dossier MM

Le dossier MM contient les scripts, les images et les classes partagés utilisés par les extensions fournies avec Dreamweaver, comme les scripts de création d'une barre de navigation, spécifiant les appels pré chargés, et les définitions des raccourcis clavier.

Dossier Scripts

Le sous-dossier Scripts contient les fonctions utilitaires suivantes :

CFCutilities.js	Contient les fonctions utilitaires liées aux composants Macromedia ColdFusion. Les fonctions permettent d'analyser les attributs depuis la balise d'ouverture d'un nœud donné, d'analyser une arborescence CFC, d'obtenir le DOM d'URL en cours, d'obtenir le DOM de CFC, etc.
event.js	Contient des fonctions permettant d'enregistrer les événements, de notifier les parties des événements à partir du fichier menus.xml et d'ajouter des notificateurs d'événement au fichier menus.xml.
FlashObjects.js	Contient des fonctions permettant de mettre à jour un sélecteur de couleur, de vérifier la couleur hex, de vérifier un lien absolu, d'ajouter une extension à un nom de fichier, de générer des messages d'erreur, de définir des attributs Flash, de vérifier un lien vers un objet Flash, etc.
insertFireworksHTML.js	Contient des fonctions permettant d'insérer un code HTML Fireworks dans les documents Dreamweaver. Les fonctions permettent de vérifier si le document en cours est un document Fireworks, d'insérer du code HTML Fireworks au point d'insertion, de mettre à jour le bloc de style Macromedia Fireworks par rapport à Dreamweaver, etc. Contient également des fonctions utilitaires liées.

jumpMenuUI.js	Contient des fonctions à utiliser avec l'objet Menu de reroutage et le comportement Menu de reroutage. Les fonctions permettent de compléter les options de menus, de créer une étiquette d'option, d'ajouter une option, de supprimer une option, etc.
keyCodes.js	Contient un tableau de codes de raccourcis clavier.
navBar.js	Contient des classes et des fonctions permettant de travailler avec une barre de navigation et des éléments de la barre de navigation. Comprend des fonctions permettant d'ajouter, de supprimer et de manipuler des éléments de la barre de navigation.
NBInit.js	Contient des fonctions liées aux comportements des images de barre de navigation.
pageEncodings.js	Définit différents codes de langue.
preload.js	Contient des fonctions permettant d'ajouter et de supprimer des appels d'images pré chargées dans le gestionnaire `BODY/onLoad MM_preloadImages`.
RecordsetDialogClass.js	Contient la classe statique et les fonctions permettant d'afficher l'interface utilisateur des comportements de serveur de jeux d'enregistrements. Les fonctions déterminent l'interface à afficher, simple ou avancée. Héberge également des fonctionnalités partagées entre les implémentations de l'interface utilisateur et permet de passer d'une interface utilisateur à une autre.
sbUtils.js	Contient des fonctions partagées à utiliser dans les comportements de serveur Macromedia. La classe `dwscripts` du dossier Configuration/Shared/Common/Scripts contient des utilitaires plus généraux.
setText.js	Contient des fonctions permettant de quitter une chaîne d'expression, de revenir à une chaîne d'expression et d'extraire une chaîne d'expression.
sortTable.js	Contient des fonctions permettant d'initialiser et de trier un tableau, ainsi que des fonctions permettant de trier un tableau, de définir le pointeur de la souris en icône de main ou en pointeur et de vérifier le type et la version du navigateur.

Le dossier Scripts contient également deux sous-dossiers : Class et CMN.

Dossier Class

Le dossier Class contient les fonctions utilitaires suivantes :

classCheckbox.js	Aide à manipuler un contrôle de case à cocher dans votre extension HTML.
FileClass.js	Contient une classe représentant un fichier dans le système de fichiers. Les chemins sont représentés par des URL pour garantir la compatibilité entre plates-formes. Les méthodes sont `toString()`, `getName()`, `getSimpleName()`, `getExtension()`, `getPath()`, `setPath()`, `isAbsolute()`, `getAbsolutePath()`, `getParent()`, `getAbsoluteParent()`, `exists()`, `getAttributes()`, `canRead()`, `canWrite()`, `isFile()`, `isFolder()`, `listFolder()`, `createFolder()`, `getContents()`, `setContents()`, `copyTo()` et `remove()`.
GridClass.js	Contient une classe permettant de gérer `MM:TREECONTROL`.
GridControlClass.js	Version plus ancienne de `GridControlClass` dans le dossier Common. Voir le fichier GridControlClass.js du dossier Shared/Common/Scripts.
ImageButtonClass.js	Version plus ancienne de `ImageButtonClass` dans le dossier Common. Voir le fichier ImageButtonClass.js du dossier Shared/Common/Scripts.
ListControlClass.js	Version plus ancienne de `ListControlClass` dans le dossier Common. Voir le fichier Shared/Common/Scripts/ ListControlClass.js.
NameValuePairClass.js	Crée et gère une liste de paires nom/valeur. Les noms peuvent comporter n'importe quels caractères. Les valeurs peuvent être vides, mais elles ne peuvent pas être définies sur `null`, qui reviendrait à les supprimer.
PageControlClass.js	Exemple d'une classe de page à utiliser avec la classe `TabControl`. Voir la classe `TabControl`.
PreferencesClass.js	Contient un objet et des méthodes contenant toutes les informations de préférences d'une commande.
RadioGroupClass.js	Version plus ancienne de `RadioGroupClass` dans le dossier Common. Voir le fichier RadioGroupClass.js du dossier Shared/Common/Scripts.
TabControlClass.js	Aide à créer une extension ayant plusieurs onglets, `page.lastUnload()`

Dossier CMN

Le dossier CMN contient les fonctions utilitaires suivantes :

dateID.js	Contient deux fonctions, `createDateID()` et `decipherDateID()`. Pour trois chaînes données, dayFormat, dateFormat et timeFormat, `createDateID()` crée un ID pour chacune d'elles. Pour un tableau de dates donné, `decipherDateID()` renvoie un tableau contenant trois éléments : dayFormat, dateFormat et timeFormat.
displayHelp.js	Contient une fonction permettant d'afficher le document d'aide spécifié.
docInfo.js	Contient des fonctions qui fournissent des informations sur le document de l'utilisateur. Ces fonctions permettent d'effectuer des opérations telles que le renvoi d'un tableau de références d'objets pour un type de navigateur et une balise spécifiés, le renvoi de toutes les instances d'un nom de balise spécifié, la recherche d'une balise qui entoure la sélection en cours, etc.
DOM.js	Contient des fonctions d'aide générales pour utiliser le DOM Dreamweaver. Comprend des fonctions qui permettent d'obtenir le nœud racine du document actif, de trouver la balise d'un nom donné, de créer une liste de nœuds à partir du nœud de départ spécifié, de vérifier si une balise donnée est contenue dans une autre balise, d'effectuer diverses opérations sur les fonctions de comportement, etc.
enableControl.js	Contient une fonction, `SetEnabled()`, qui permet d'activer ou de désactiver une commande en fonction des arguments qu'elle reçoit. Il est possible d'activer une commande déjà activée ou de désactiver une commande déjà désactivée.
errmsg.js	Contient des fonctions de connexion pour accumuler la sortie de tracés dans un tableau de pages de connexion qui apparaissent dans une boîte de dialogue.
file.js	Contient des fonctions en rapport avec des opérations sur les fichiers. Ces fonctions permettent à l'utilisateur de rechercher un nom de fichier local, de convertir le chemin relatif en chemin URL du fichier, de renvoyer le nom de fichier du document en cours, de déterminer si un document spécifié a été enregistré sur le site en cours et de renvoyer son chemin relatif, ou de déterminer si un fichier spécifié est ouvert.

form.js	Contient des fonctions permettant d'ajouter un formulaire autour d'une chaîne de texte donnée si un formulaire n'existe pas déjà dans le document ou le calque en cours. Comprend des fonctions permettant de déterminer si un objet est un calque et si le curseur se trouve à l'intérieur d'un formulaire.
handler.js	Contient des fonctions permettant d'obtenir une fonction pour un gestionnaire d'événement, d'ajouter une fonction à un gestionnaire d'événement et de supprimer une fonction d'un gestionnaire d'événement.
helper.js	Contient diverses fonctions utiles permettant de remplacer le codage, de rétablir les guillemets ("), de vérifier si un nœud se trouve dans une plage de sélection et de vérifier s'il existe des noms d'objets en double.
insertion.js	Contient la fonction `insertIntoDocument()`, qui insère une chaîne de texte dans un document au point d'insertion. Contient également les fonctions de prise en charge `getHigherBlockTag()` et `arrContains()`. La fonction `getHigherBlockTag()` permet d'obtenir le prochain blockTag le plus élevé, comme défini dans le tableau blockTags. La fonction `arrCon()` permet de localiser un élément spécifié dans un tableau.
localText.js	Variables réservées, non destinées à un usage général. Utilisez Startup/mminit.htm à la place ou utilisez les chaînes provenant des fichiers Dreamweaver Configuration/Strings/*.xml.
menuItem.js	Contient des fonctions permettant d'ajouter des étoiles ou des valeurs à un élément de menu et de les supprimer.
niceName.js	Contient des fonctions permettant de convertir un tableau de références Object en un tableau de noms plus simples.
quickString.js	Contient des fonctions permettant de regrouper des chaînes de petite taille sans avoir à allouer de la mémoire à chaque fois.
string.js	Contient un ensemble générique de fonctions permettant de manipuler et d'analyser des chaînes de texte. Ces fonctions sont : `extractArgs()`, `escQuotes()`, `unescQuotes()`, `quoteMeta()`, `errMsg()`, `badChars()`, `getParam()`, `quote()`, `stripSpaces()`, `StripChars()`, `AllInRange()`, `reformat()`, `trim()`, `createDisplayString()`, `entityNameEncode()`, `entityNameDecode()`, `stripAccelerator()` et `SprintF()`,

TemplateUtils.js	Contient des fonctions utilitaires pour les modèles Dreamweaver. Ces fonctions permettent d'insérer une région modifiable dans un document, d'insérer une région répétée dans un document, de rechercher une région modifiable spécifiée dans un document, etc.
UI.js	Contient des fonctions génériques qui contrôlent l'interface utilisateur. Ces fonctions permettent de localiser un objet désigné dans le document en cours, de charger les options de liste de sélection avec des chaînes localisées, de renvoyer la valeur de l'attribut d'une option sélectionnée et d'envelopper le message d'une alerte.

Autres dossiers

La liste suivante décrit les autres dossiers intéressants du dossier Shared :

- **Controls**

  Le dossier Controls contient les éléments utilisés pour créer un comportement de serveur. Ces contrôles comprennent des interfaces pour les menus de texte et de jeux d'enregistrements.

  REMARQUE

  Ces contrôles sont utilisés par le Créateur de comportements de serveur de Dreamweaver et par de nombreux comportements de serveur de Dreamweaver, mais certains sont utiles pour gérer un contrôle dans votre extension.

- **Fireworks**

  Le dossier Fireworks contient les fichiers de prise en charge pour l'intégration de Fireworks.

- **UltraDev**

  Dreamweaver conserve ce dossier principalement pour des raisons de compatibilité. Il ne doit pas être utilisé pour les nouvelles extensions. Utilisez le dossier Dreamweaver Configuration/Shared/Common, qui contient également la majeure partie de cette fonctionnalité. Voir *Dossier Common*, page 506.

Utilisation du dossier Shared

Commencez par rechercher les codes d'extension dans le dossier Dreamweaver Configuration/Shared/Common. Ce dossier contient en effet les fonctionnalités les plus courantes et les plus récentes.

Les extensions peuvent utiliser les ressources du dossier Shared pour leur propre fonctionnalité. Un objet, une commande ou une autre extension peuvent spécifier l'un des fichiers JavaScript dans le dossier Shared comme fichier source dans une balise `script`, puis utiliser cette fonction dans le corps du fichier ou dans un autre fichier JavaScript inclus. Les objets et les commandes peuvent même lier plusieurs fichiers JavaScript ensemble et ces fichiers JavaScript peuvent utiliser les ressources du dossier Shared.

Par exemple, ouvrez le fichier d'objet Hypertext (Hyperlink.htm) dans le dossier d'application Configuration/Objects/Common. Remarquez que la balise d'en-tête contient les lignes suivantes :

```
<script language="javascript" src="../../Shared/Common/Scripts/
  ListControlClass.js"></script>
<script language="javascript" src="Hyperlink.js"></script>
```

Si vous ouvrez le fichier Hyperlink.js lié, les lignes suivantes apparaissent :

```
LIST_LINKS = new ListControl('linkPath');
```

et

```
LIST_TARGETS = new ListControl('linkTarget');
```

Avec les nouvelles déclarations `listControl`, Hyperlink.js définit deux nouveaux objets ListControl. Le code du fichier Hyperlink.htm les joint ensuite aux commandes `SELECT` du formulaire, comme suit :

```
<td  align="left"> <input name="linkText" type="text"
class="basicTextField" value="">
```

et

```
<td align="left" nowrap><select name="linkPath" class="basicTextField"
  editable="true">
```

Le script Hyperlink.js peut maintenant appeler les méthodes ou obtenir les propriétés des objets `LIST_LINKS` ou `LIST_TARGETS` pour qu'ils interagissent avec les commandes `SELECT` du formulaire.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X