22

Interfaçage Flash-PHP-XML et autres interfaçages XML

Dans les chapitres 20 et 21, nous avons présenté différentes techniques de traitement de données XML à partir d'une application Flash ou PHP. Nous allons maintenant appliquer ces techniques d'interfaçage grâce à l'étude de plusieurs cas pratiques.

Vous découvrirez une application PHP permettant la mise à jour d'un fichier XML (interface PHP 4-XML), une application Flash intégrant un menu déroulant dont les options sont définies par un fichier XML externe (interface Flash-XML) et enfin une application Flash-PHP permettant de visionner et d'annoter des images dont l'URL et le commentaire sont stockés dans un fichier XML externe (interface Flash-PHP-XML).

Pour compléter ce tour d'horizon des applications Flash ou PHP utilisant le format XML, deux sections sont dédiées aux interfaçages Flash-XML-PHP-MySQL et Flash-XML-serveur socket PHP. Une application de signets dynamiques illustre les interfaces Flash-XML-PHP-MySQL et une application de dialogue en ligne (chat) illustre les interfaces Flash-XML-serveur socket PHP.

Interface PHP-XML

Voici une application PHP qui permet de mettre à jour un fichier XML.

> **À noter**
>
> Les scripts de lecture et l'analyse des données du fichier XML utilisent les fonctions de l'extension XML de PHP présentées dans le chapitre 21.

Formulaire PHP de mise à jour d'un fichier XML

Cette application permet de mettre à jour des informations stockées dans un fichier XML. Elle est composée du fichier XML dans lequel seront stockés les informations et d'un fichier PHP qui assure l'actualisation des données du fichier XML. Le fichier PHP contient un formulaire dans lequel seront affichés les informations issues du fichier XML avant modification et un programme PHP qui lit les données du fichier XML et les écrit dans ce même fichier lors de la validation du formulaire. Le script PHP est constitué de deux programmes déjà étudiés dans le chapitre 21 : un premier script assure la lecture en direct du fichier XML (revoir `ecritureXml3.php` dans le chapitre 21) et le second écrit les données modifiées par le formulaire dans le fichier XML (revoir `lectureXml2.php` dans le chapitre 21).

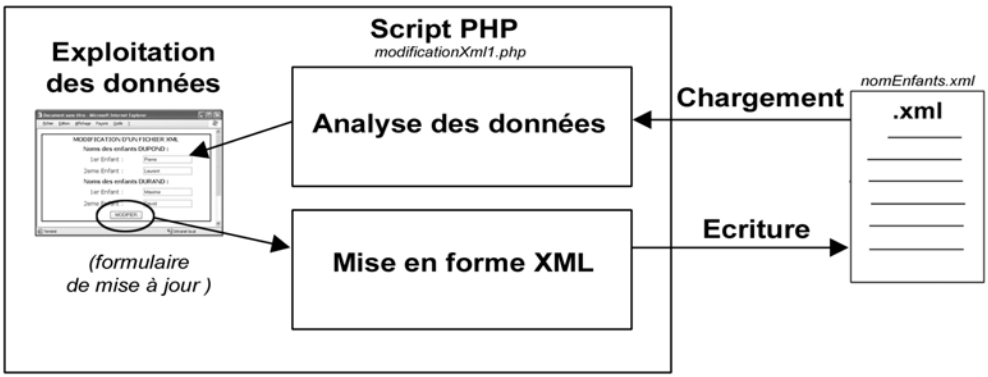

Figure 22-1

Principe de l'interface PHP-XML utilisée pour l'application du formulaire de mise à jour d'un fichier XML

Le fichier XML

Le fichier XML est identique au fichier `nomEnfants.xml` utilisé dans les scripts de lecture XML présentés dans le chapitre 21. Il est structuré autour d'un élément racine `<noms>` dans lequel on retrouve plusieurs éléments `<pere>` comprenant à leur tour des éléments `<enfant>` et leur valeur (valeurs correspondant aux noms des enfants).

> **À noter**
>
> Les éléments `<pere>` sont personnalisés à l'aide d'un attribut `nom` indiquant le nom de la famille concernée.

Fichier nomEnfants.xml :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<noms>
  <pere nom="Dupond">
        <enfant>Pierre</enfant>
        <enfant>Laurent</enfant>
  </pere>
```

```
  <pere nom="Durand">
        <enfant>Maxime</enfant>
        <enfant>David</enfant>
  </pere>
</noms>
```

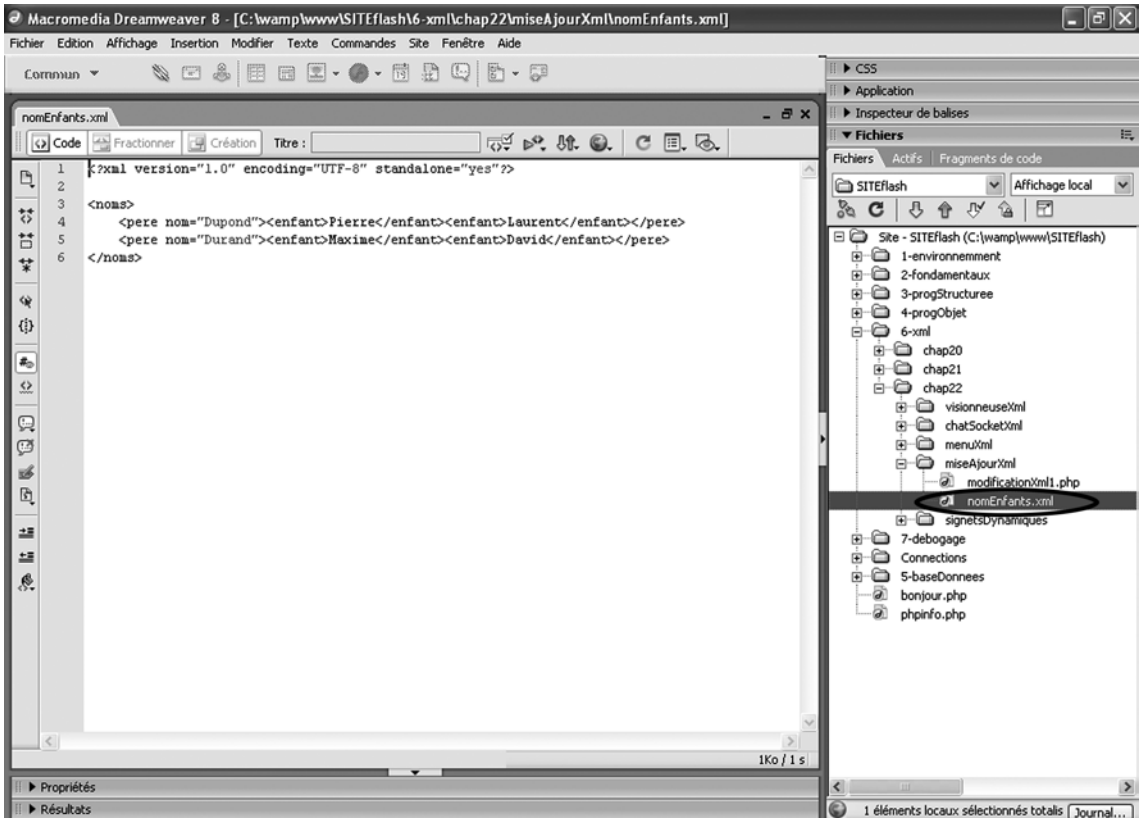

Figure 22-2

Création du fichier XML nomEnfants.xml avec Dreamweaver

1. Créez un nouveau document XML avec Dreamweaver et sauvegardez-le sous le nom `nomEnfants.xml` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local. Nous avons enregistré le fichier XML dans le répertoire `SITEflash/6-xml/chap22/miseAjourXml/` mais vous pouvez utiliser tout autre répertoire de votre choix dans la mesure où il se trouve dans le dossier `www/SITEflash/`.

2. Vérifiez que l'en-tête XML créé automatiquement est conforme à celui du code ci-dessous et modifiez-le si nécessaire (voir figure 22-2) :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

3. À la suite de l'en-tête, ajoutez une balise ouvrante `<noms>`.
4. Sous ce premier élément, créez deux séries de balises `<pere></pere>`. Dans chaque balise `pere`, ajoutez un attribut `nom` suivi du nom de famille approprié.
5. À l'intérieur de chaque élément `pere`, ajoutez deux éléments `<enfant></enfant>` et renseignez leur valeur avec les noms des enfants concernés (n'ajoutez pas d'espace entre les éléments ; voir le code de la figure 22-2).
6. Clôturez le fichier en ajoutant une balise fermante `</noms>` puis enregistrez-le.

Le fichier PHP

Le fichier PHP est constitué de code et d'un formulaire qui s'affiche à l'écran lors de l'appel du fichier. La première partie du code lit les données issues du fichier XML afin d'initialiser les valeurs par défaut de chaque champ du formulaire. La seconde partie met en forme un document XML en y incluant les données modifiées (les données sont envoyées lors de la validation du formulaire intégré dans le fichier) et gère l'écriture de ce document dans le fichier XML.

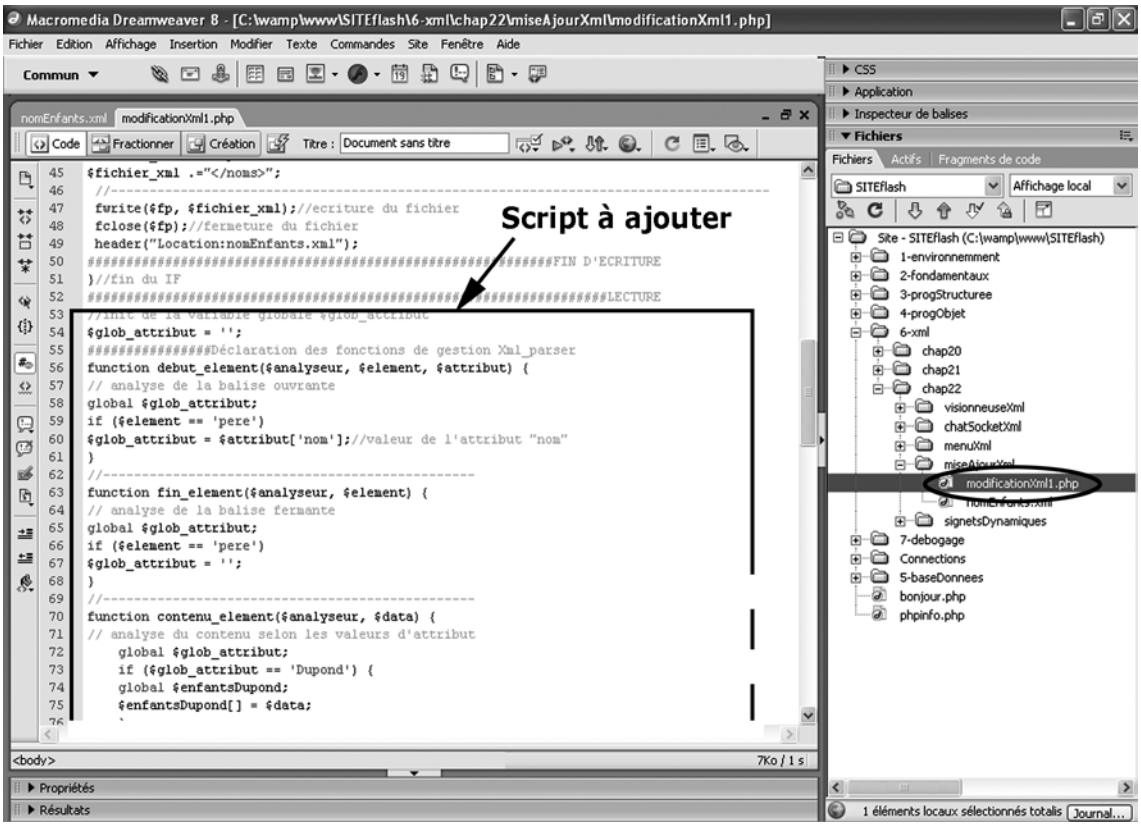

Figure 22-3

Création du formulaire intégré dans le fichier modificationXml1.php avec Dreamweaver

Étapes de création du script PHP :

1. Pour vous éviter de créer de nouveau le formulaire de mise à jour, vous utiliserez le fichier `ecritureXml3.php` (revoir chapitre 21) après l'avoir enregistré sous le nom `modificationXml1.php` dans le répertoire `/6-xml/chap22/interfacePhpXml/` (sous-répertoire de `/wamp/www/SITEflash/`).
2. Copiez le script ci-dessous à partir du fichier `lectureXml2.php` (voir chapitre 21). Ce script sera intégré après le bloc de script d'écriture du fichier `modificationXml1.php` (voir figure 22-3) :

```
##############################################################LECTURE
// Initialisation de la variable globale $glob_attribut
$glob_attribut = '';
################Déclaration des fonctions de gestion xml_parser
function debut_element($analyseur, $element, $attribut) {
// Analyse de la balise ouvrante
global $glob_attribut;
if ($element == 'pere')
$glob_attribut = $attribut['nom']; // Valeur de l'attribut nom
}
//-----------------------------------------------
function fin_element($analyseur, $element) {
// Analyse de la balise fermante
global $glob_attribut;
if ($element == 'pere')
$glob_attribut = '';
}
//-----------------------------------------------
function contenu_element($analyseur, $data) {
// Analyse du contenu selon les valeurs d'attribut
      global $glob_attribut;
      if ($glob_attribut == 'Dupond') {
      global $enfantsDupond;
      $enfantsDupond[] = $data;
      }
      if ($glob_attribut == 'Durand') {
      global $enfantsDurand;
      $enfantsDurand[] = $data;
      }
}
################Configuration et appel du parser xml
$analyseur = xml_parser_create(); // création du parser
xml_parser_set_option($analyseur, XML_OPTION_CASE_FOLDING, false);
// Permet de différencier les MAJUSCULES et les minuscules dans les noms de balise
xml_parser_set_option($analyseur, XML_OPTION_TARGET_ENCODING,"UTF-8");
// Permet d'indiquer le type de codage du fichier XML
xml_set_element_handler($analyseur, 'debut_element','fin_element');
xml_set_character_data_handler($analyseur, 'contenu_element');
```

```
//----------------------------------------
$document = file('nomEnfants.xml');//chargement du fichier XML
// Analyse chaque ligne du document XML
foreach ($document as $line) {
xml_parse($analyseur, $line);
}
// Détruit l'analyseur XML
xml_parser_free($analyseur);
##########################################################FIN DE LECTURE
?>
```

3. Le formulaire utilisé dans le script `ecritureXml3.php` était uniquement destiné à l'ajout de données ; les valeurs initiales des quatre champs n'étaient donc pas configurées. Ajoutez quatre petits scripts dans le formulaire afin de combler ce manque et de le transformer en formulaire de mise à jour (les valeurs initiales utilisées sont celles des tableaux `$enfantsDupond` et `$enfants-Durand`). Les scripts à ajouter dans le formulaire sont signalés par des caractères gras dans le code ci-dessous :

```
<table width="496" border="0" cellspacing="0" cellpadding="5">
            <tr>
              <td colspan="2"><div align="center" class="Style1">MODIFICATION
                  D'UN FICHIER XML </div></td>
            </tr>
            <tr>
              <td colspan="2"><div align="center"><span class="Style1">Noms des
                    enfants DUPOND : </span></div></td>
            </tr>
            <tr>
              <td width="248"><div align="right" class="Style2">1er Enfant :
                  </div></td>
              <td width="248"><input name="enfant11" type="text" id="enfant11"
              ➥value="<?php echo $enfantsDupond[0]; ?>"></td>
            </tr>
            <tr>
              <td><div align="right" class="Style2">2eme Enfant : </div></td>
              <td><input name="enfant12" type="text" id="enfant12"
              ➥value="<?php echo $enfantsDupond[1]; ?>"></td>
            </tr>
            <tr>
              <td colspan="2"><div align="center"><span class="Style1">Noms des
                    enfants DURAND : </span></div></td>
            </tr>
            <tr>
              <td width="248"><div align="right" class="Style2">1er Enfant :
                  </div></td>
```

```
        <td><input name="enfant21" type="text" id="enfant21"
        ➥value="<?php echo $enfantsDurand[0]; ?>"></td>
      </tr>
      <tr>
        <td><div align="right" class="Style2">2eme Enfant : </div></td>
        <td><input name="enfant22" type="text" id="enfant22"
        ➥value="<?php echo $enfantsDurand[1]; ?>"></td>
      </tr>
      <tr>
        <td colspan="2"><div align="center">
            <input type="submit" name="Submit" value=" MODIFIER">
            <input name="action" type="hidden" id="action" value="ecriture">
        </div></td>
      </tr>
    </table>
```

4. Une fois ces modifications effectuées, enregistrez le nouveau fichier et testez son fonctionnement depuis le Web local. Dès l'appel du fichier `modificationXml1.php`, le formulaire s'affiche dans le navigateur avec ses champs initialisés. Modifiez l'un des prénoms et cliquez sur le bouton Modifier. Après validation du formulaire, le document XML `$fichier_xml` est modifié à partir des valeurs du formulaire puis enregistré dans le fichier `nomEnfants.xml`. Le fichier `nomEnfants.xml` modifié s'affiche dans la fenêtre du navigateur (voir la figure 22-4).

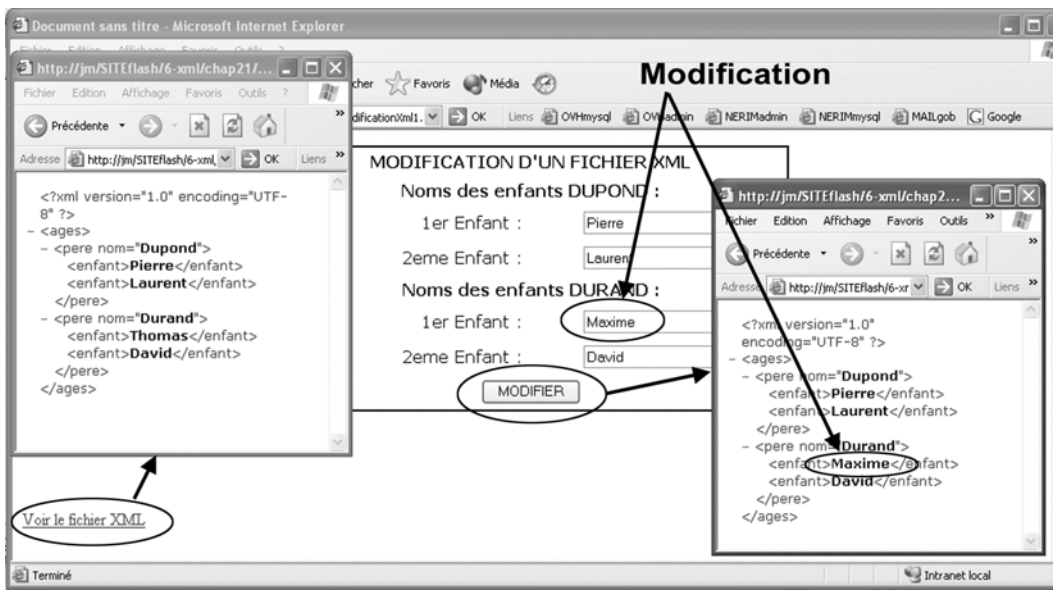

Figure 22-4

Test du fichier modificationXml1.php dans le Web local

Interface Flash-XML

Avec les interfaces Flash-XML, les fichiers XML sont exploités pour le stockage de données structurées. Cette technique, souvent utilisée pour mémoriser les informations de configuration d'une application Flash, est aussi une bonne alternative à l'usage d'une base de données (exploitée exclusivement en lecture) car elle est relativement simple à mettre en œuvre et peut être exploitée sur tout type de plate-forme en ligne ou en local.

> **À noter**
>
> Cette interface Flash-XML ne peut pas être considérée comme un interfaçage (tel que nous l'avons défini dans les terminologies des sources de données du chapitre 11) entre l'application Flash et XML car le système n'assure pas un transfert bidirectionnel mais un simple chargement du fichier XML (cette application ne pourra donc pas ajouter ou modifier les données stockées dans le fichier XML).

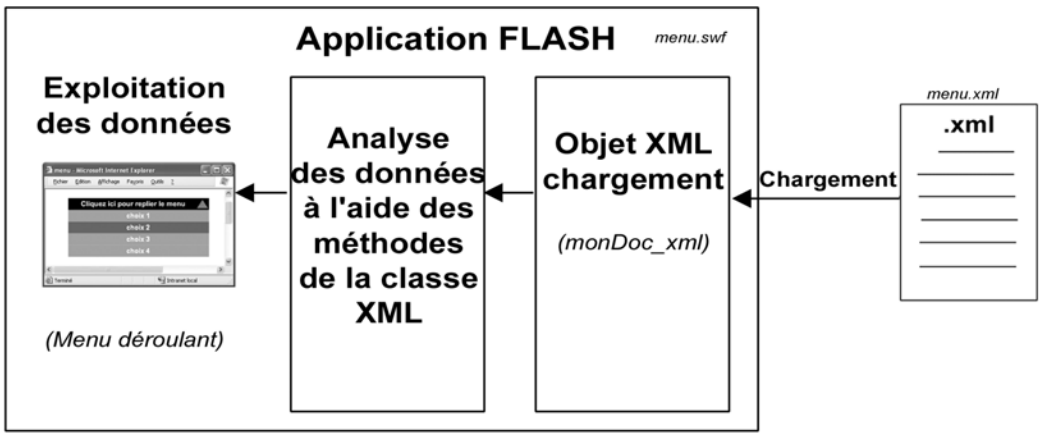

Figure 22-5

Principe de l'interface Flash-XML utilisée pour l'application du menu déroulant XML

Un menu déroulant XML

Nous vous proposons de réaliser un menu déroulant dont la configuration sera définie dans un fichier XML. Ce menu permet d'ouvrir différentes pages Web grâce à un simple clic sur l'option désirée. Le nombre d'options proposées par le menu peut varier selon la structure du fichier XML chargé.

Le fichier XML

Le fichier XML comprend un élément racine `<menu>` dans lequel on retrouve les différentes options proposées dans le menu. Chaque élément `<option>` comprend deux attributs, `titre` et `lien`, qui correspondent à l'étiquette affichée dans le menu et à l'URL ciblée si l'on clique dessus.

Fichier `menu.xml` :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <menu>
                <option titre="choix 1" lien="page1.htm" />
                <option titre="choix 2" lien="page2.htm" />
                <option titre="choix 3" lien="page3.htm" />
                <option titre="choix 4" lien="page4.htm" />
  </menu>
```

1. Créez un nouveau document XML avec Dreamweaver et sauvegardez-le sous le nom `menu.xml` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local. Nous avons sélectionné le répertoire `SITEflash/6-xml/chap22/menuXml/` mais vous pouvez utiliser tout autre répertoire de votre choix.
2. Vérifiez que l'en-tête XML créé automatiquement est conforme à celui du code ci-dessous et modifiez-le si nécessaire :

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   ```

3. À la suite de l'en-tête, ajoutez une balise ouvrante `<menu>`.
4. Sous ce premier élément, créez une balise `<option>` avec ses deux attributs `choix` et `lien` puis dupliquez deux fois cette ligne. Personnalisez chacun des attributs en vous assurant que les URL des pages Web sont bien accessibles depuis le répertoire du fichier XML.
5. Clôturez le fichier en ajoutant une balise fermante `</menu>` puis enregistrez votre nouveau fichier (voir figure 22-6).

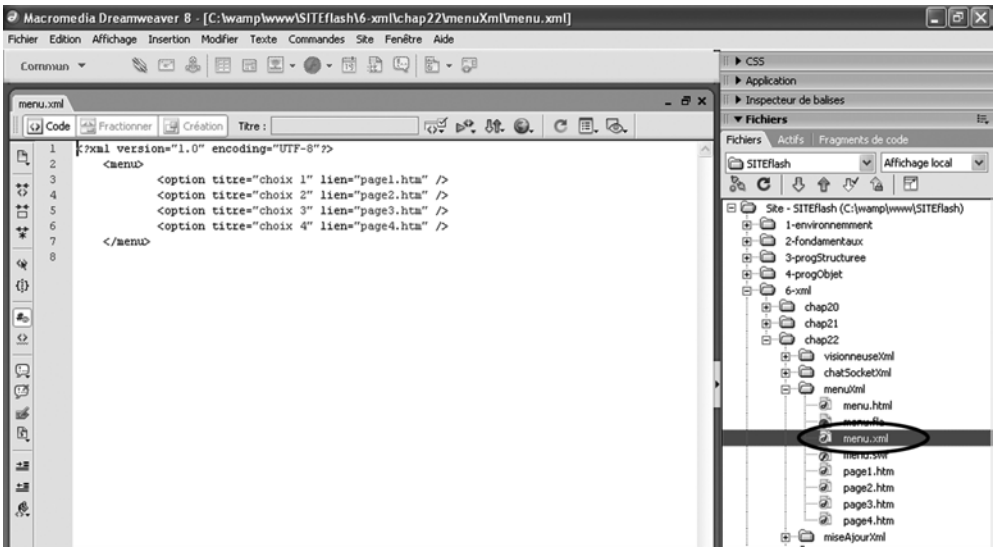

Figure 22-6

Création du fichier XML menu.xml avec Dreamweaver

Le document Flash

Le document Flash doit afficher le menu et ses options selon la configuration du fichier `menu.xml` créé précédemment.

> **À noter**
>
> L'application réalisée ci-dessous a pour seul but de tester le fonctionnement du menu, mais la procédure serait identique si vous deviez intégrer le menu déroulant dans une application existante.

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `menu.fla` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local (utilisez le même répertoire que celui dans lequel vous avez précédemment enregistré le fichier XML).
2. Créez trois calques : Label, Action et Menu.
3. Dans le calque Label, créez une image clé dans l'image 10 puis une autre dans l'image 20. Nommez l'image clé 10 `affichage` (dans le champ `Image` du panneau des propriétés) et l'image clé 20 `chargement`.
4. Placez-vous dans l'image clé 1 du calque Action et saisissez le code ci-dessous. Les deux premières lignes permettent de créer l'objet XML et de le configurer afin d'ignorer les éventuels espaces dans le fichier chargé. La partie qui suit correspond à la déclaration du gestionnaire `onLoad()`. Ainsi, dès que le fichier XML sera complètement chargé, la tête de lecture sera placée sur l'étiquette affichage. La dernière ligne appelle la méthode `Load()` afin de démarrer le chargement du fichier `menu.xml` dans l'objet `monDoc_xml` (voir figure 22-7).

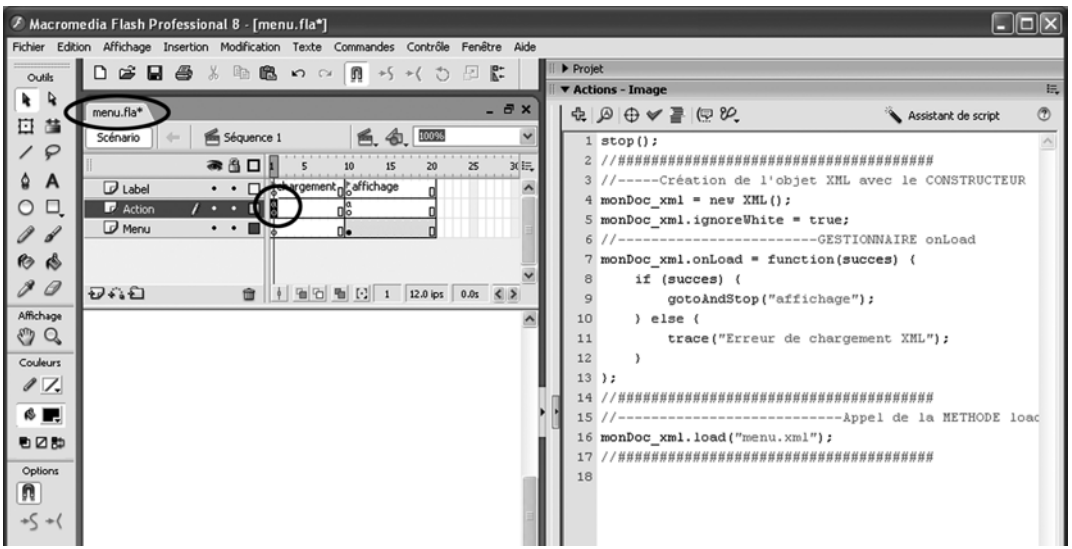

Figure 22-7

Configuration de l'image clé 1 (étiquette chargement) du calque Action

```
//##########################################
//-----Création de l'objet XML avec le CONSTRUCTEUR
monDoc_xml = new XML();

monDoc_xml.ignoreWhite = true;
//-----------------------GESTIONNAIRE onLoad
monDoc_xml.onLoad = function(succes) {
  if (succes) {
    gotoAndStop("affichage");
};
//##########################################
//--------------------------Appel de la MÉTHODE load()
monDoc_xml.load("menu.xml");
//##########################################
stop();
```

5. Placez-vous ensuite sur l'image clé 10 et créez une image clé. Dans cette même image, saisissez le code ci-dessous (voir figure 22-8). Cette partie correspond aux déclarations des différentes fonctions utilisées dans l'application : `initBtnDeplier()`, `initBtnReplier()` (ces deux fonctions sont appelées par les boutons deplier1_btn et replier1_btn placés dans le clip de la tête du menu), `afficheOptions(xPos, yPos)`, `effaceOptions()` (ces deux fonctions permettent d'afficher ou d'effacer les différentes options du menu) et `appelURL(url)` (cette dernière fonction permet d'ouvrir une fenêtre de navigateur lors d'un clic sur l'une des options).

```
//##########################################
function initBtnDeplier() {
      // Gestionnaire du bouton deplier1_btn
      tete1_mc.deplier1_btn.onRelease = function() {
            tete1_mc.gotoAndStop("Déplier"); // Aller à l'image Déplier
            initBtnReplier(); // Initialiser le gestionnaire du bouton replier1_btn
            afficheOptions(tete1_mc._x, tete1_mc._y); // afficher les options
      }; // Fin du gestionnaire
} // Fin de la fonction
//##########################################
function initBtnReplier() {
      // Gestionnaire du bouton replier1_btn
      tete1_mc.replier1_btn.onRelease = function() {
            tete1_mc.gotoAndStop("Replier"); // Aller à l'image Replier
            initBtnDeplier(); // Initialiser le gestionnaire du bouton deplier1_btn
            effaceOptions(); // Effacer toutes les options
      }; // Fin du gestionnaire
} // Fin de la fonction
```

```
//########################################
// Fonction de construction et de gestion du sous-menu
function afficheOptions(xPos, yPos) {
      tableauOption_array = new Array(); // Création du tableau des options
      for (i=0; i<monDoc_xml.firstChild.childNodes.length; i++) {
             var nomOption:String = "option"+i+"_mc"; // Nom du MC option à créer
             _root.attachMovie("option_mc", nomOption, i); // Création du MC option
             var cetteOption:MovieClip = _root[nomOption]; // MC créé
             tableauOption_array.push(nomOption);//ajout de l'option dans le tableau
             cetteOption.titre_txt.text = (monDoc_xml.firstChild.childNodes[i].
             ➥attributes.titre);
             // Initialisation du champ texte de l'option
            cetteOption.lien = (monDoc_xml.firstChild.childNodes[i].attributes.lien);
             // Initialisation du lien URL de l'option
             cetteOption._x = xPos; // Initialisation de la position X de l'option
             cetteOption._y = yPos += cetteOption._height; // Initialisation
             ➥de la position Y de l'option
             cetteOption._alpha = 60; // Règle l'alpha des options à 60 % par défaut
            // Gestionnaire d'événements si l'on survole cette option
             cetteOption.onRollOver = function() {
                    this._alpha = 100; // Augmente l'alpha à 100 %
             };
           // Gestionnaire d'événements si l'on sort de cette option
             cetteOption.onRollOut = function() {
                    this._alpha = 60;//rétablit l'alpha à 60 %
             };
             // Gestionnaire d'événements si l'on clique sur cette option
             cetteOption.onRelease = function() {
                    appelURL(this.lien);
                    // Appelle la fonction d'ouverture d'une fenêtre
             };
      } // Fin du for
} // Fin de la fonction
//########################################
// Fonction de suppression du sous-menu
function effaceOptions() {
     for (i=0; i<tableauOption_array.length; i++) {
             unloadMovie(tableauOption_array[i]); // Supprime les MC option du menu
      } // Fin du for
      delete tableauOption_array; // Supprime le tableau de mémorisation des options
} // Fin de la fonction
```

```
//##########################################
// Fonction d'ouverture d'un fenêtre avec appel de l'URL de l'option sélectionnée
function appelURL(url) {
	if (url != null) { // si l'URL existe
		getURL(url, "_blank"); // Ouverture de la page dans un nouveau navigateur
	} else {
		trace("il n'y a pas de lien pour cette option");
	}
}
//##########################################
```

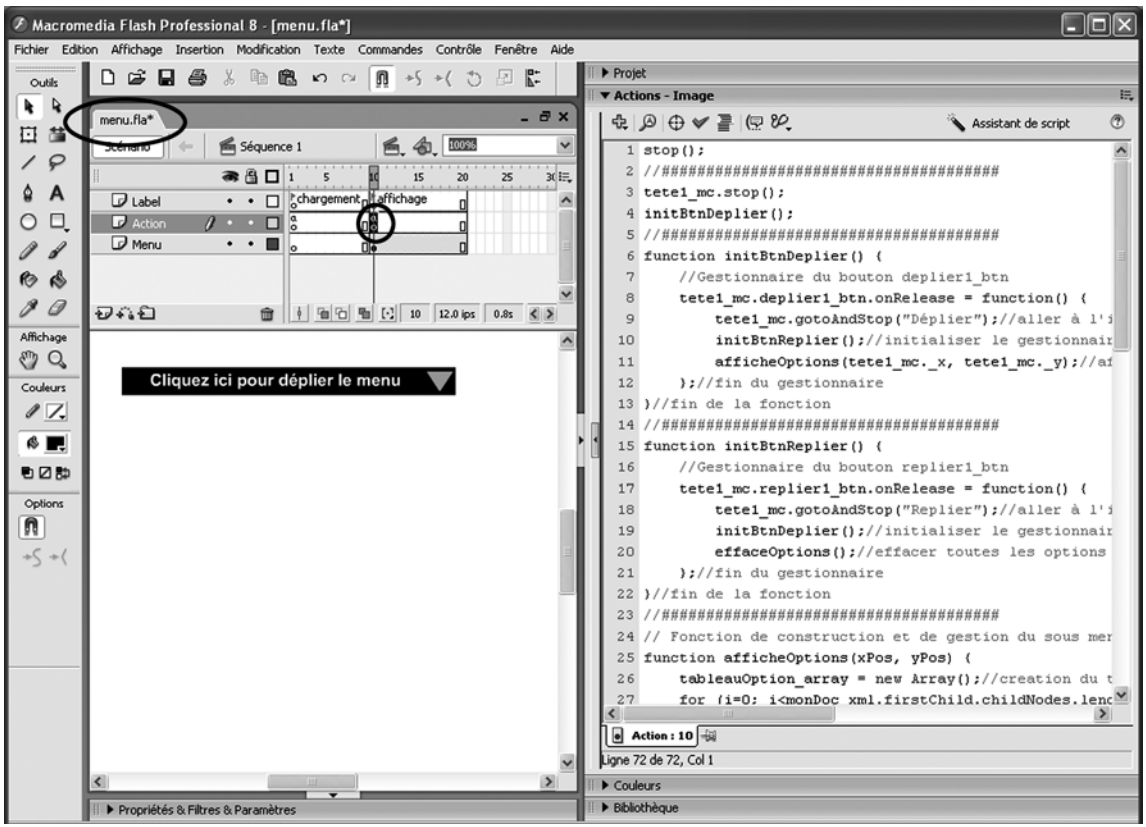

Figure 22-8

Configuration de l'image clé 10 (étiquette affichage) du calque Action

6. À la suite de ces déclarations de fonction, saisissez le code ci-dessous. La première instruction permet d'arrêter la tête de lecture du scénario principal sur cette image clé. La deuxième instruction positionne la tête de lecture du clip `tete1_mc` (clip situé à la tête du menu déroulant

qui gère les deux boutons deplier1_btn et replier1_btn) sur la première image clé de son scénario. La troisième instruction permet d'appeler la fonction d'initialisation du bouton deplier1_btn actuellement accessible par le clip `tete1_mc` afin qu'il soit opérationnel si l'on clique dessus pour déplier le menu :

```
//##########################################
stop();
tete1_mc.stop();
initBtnDeplier();
//##########################################
```

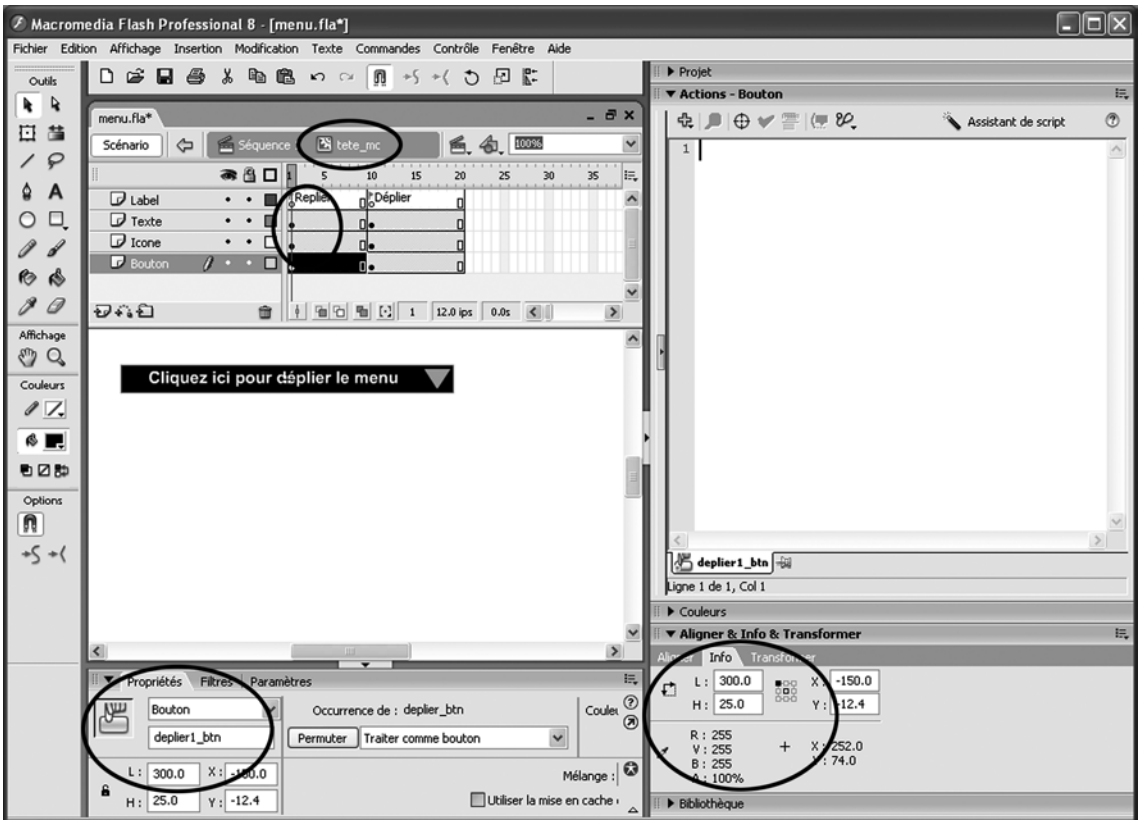

Figure 22-9

Configuration du scénario du clip tete1_mc (étiquette Replier)

7. Création de la tête du menu : placez-vous ensuite sur l'image clé 10 du calque Menu et insérez une image clé. Sur la scène de cette image, créez un nouveau clip à partir d'un rectangle noir puis nommez son occurrence `tete1_mc`. Ouvrez ce clip et créez quatre calques sur son scénario : Label, Texte, Icône et Bouton (voir figure 22-9). Dans l'image clé 1 du calque Label créez une image clé et nommez-la `Replier`. Créez une image clé dans l'image 10 de ce même calque et nommez-la `Déplier` (voir figure 22-10).

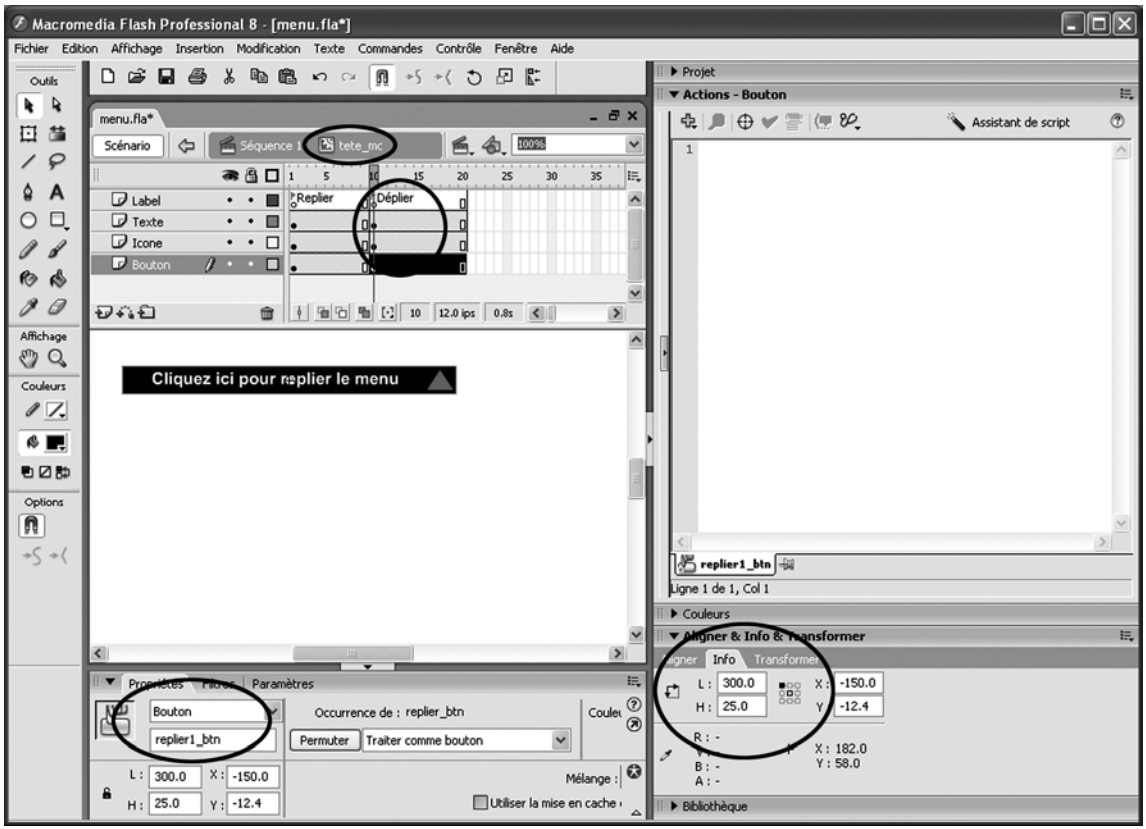

Figure 22-10

Configuration du scénario du clip tete1_mc (étiquette Déplier)

8. Personnalisez les deux calques Texte et Icône afin de distinguer les deux phases `Déplier` et `Replier`. Dans l'image clé 1, convertissez le rectangle noir en bouton et nommez son occurrence `deplier1_btn`. Réalisez la même opération dans l'image clé 10 mais nommez cette fois le bouton replier1_btn.

9. Création du clip d'option : revenez sur le scénario principal et créez un nouveau clip (touches `Ctrl + F8`) que vous nommerez `option_mc`. Dans l'image clé 1 du scénario de ce nouveau clip, ajoutez une zone de texte dynamique dont l'occurrence sera nommée `titre_txt`. Ouvrez la bibliothèque (touche `F11`). Dans la liste, cliquez avec le bouton droit sur le clip `option_mc` et sélectionnez `Liaison`. Dans la boîte de dialogue Propriétés de liaison, validez les cases à cocher Exporter pour ActionScript et Exportez pour la première fois (voir figure 22-11) afin de pouvoir par la suite créer des occurrences de ce clip à l'aide de la méthode `attachMovie()`.

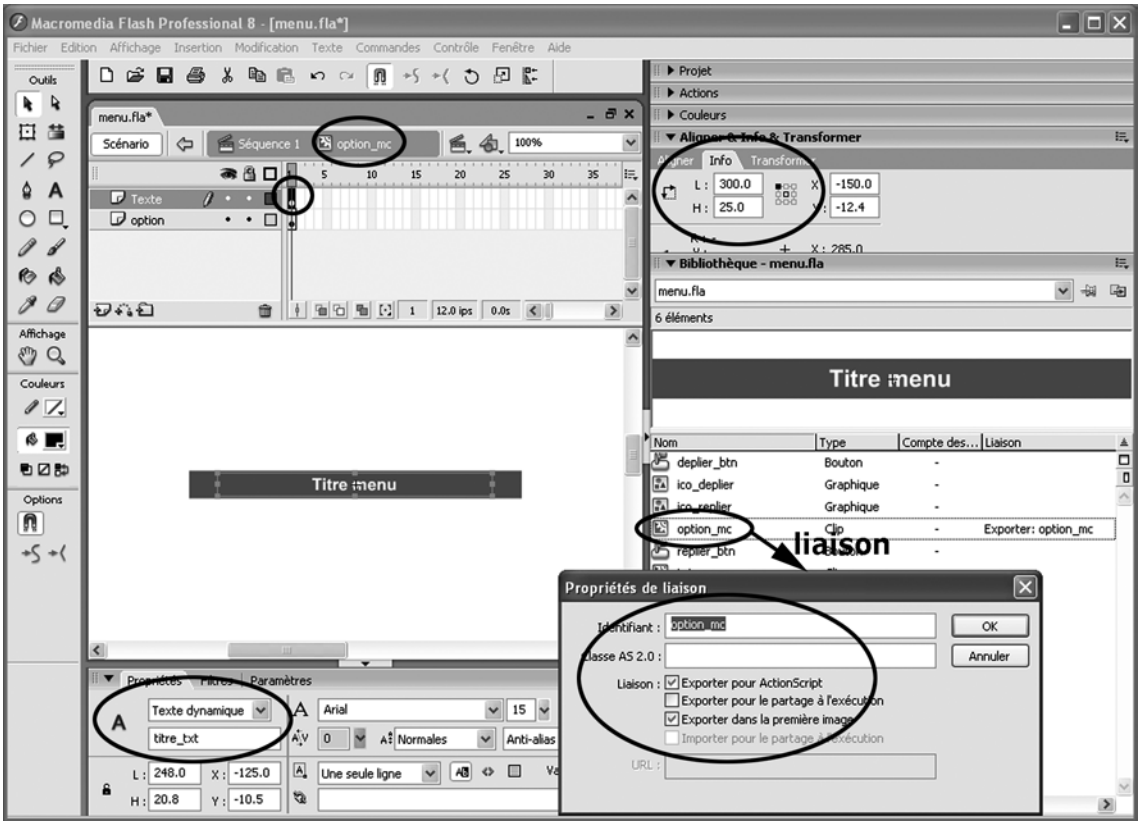

Figure 22-11

Configuration du clip option_mc

10. Enregistrez votre document et publiez-le dans le même répertoire. Testez ensuite l'application avec le mode test de Flash (`Ctrl + Entrée`) afin de vous assurer de son bon fonctionnement (voir figure 22-12). Si vous cliquez sur la tête du menu, il doit se déplier (ou se replier).

En position dépliée, si vous cliquez sur l'une des options, une fenêtre de navigateur doit s'ouvrir et afficher la page Web correspondante (voir figure 22-12).

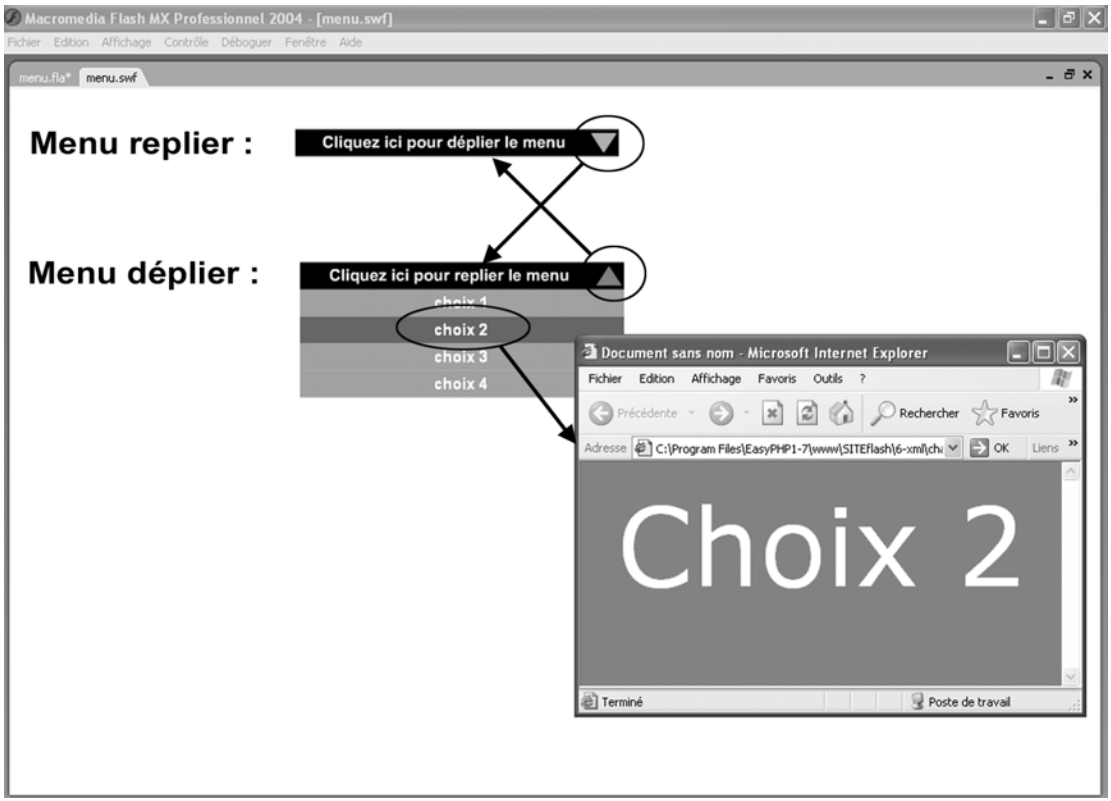

Figure 22-12
Test du menu XML

Interface Flash-PHP-XML

Contrairement aux interfaces Flash-XML (lecture seule du fichier XML), les interfaces Flash-PHP-XML permettent d'ajouter ou de modifier les données stockées dans le fichier XML mais nécessitent l'usage de scripts serveur PHP et ne sont donc plus indépendantes du type de la plate-forme. Cette technique est souvent utilisée comme alternative aux bases de données lorsque l'on désire consulter, ajouter ou actualiser des informations structurées dans un fichier XML.

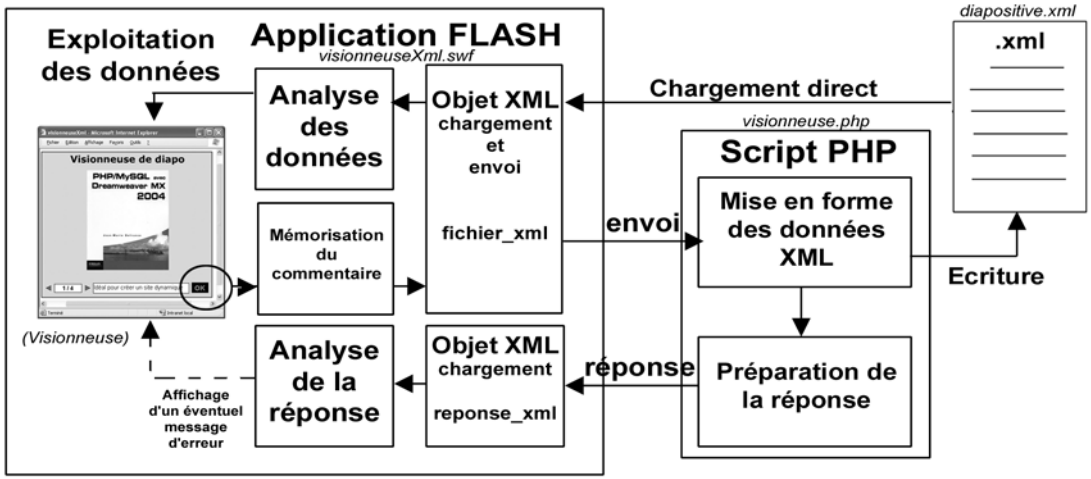

Figure 22-13

Principe de l'interface Flash-PHP-XML utilisée pour l'application de la visionneuse

Une visionneuse de diapositives XML

Nous vous proposons de réaliser une visionneuse de diapositives. Dans le contexte de cette application, nous appellerons « diapositive » une image à laquelle on peut ajouter un commentaire. Cette application peut, par exemple, être utilisée par le directeur artistique d'une agence de design qui souhaite annoter les différentes propositions de visuels d'un projet.

L'application comporte deux objets XML. Le premier est nommé `fichier_xml` car il sera à la fois utilisé comme objet de chargement lors de la lecture du fichier XML (utilisation de la méthode `Load()`) et comme objet d'envoi lors de la mémorisation d'un commentaire (utilisation de la méthode `sendAndLoad()`). Le second objet XML est nommé `reponse_xml` car il réceptionnera un message du script PHP confirmant le bon enregistrement du fichier ou un message d'erreur dans le cas contraire.

Le fichier XML

Le fichier XML est structuré à partir d'un élément racine `<visionneuse>` dans lequel on retrouve autant d'éléments `<diapo>` que de visuels à commenter. À l'intérieur d'un élément `<diapo>`, chaque information relative à la diapositive est intégrée dans des balises spécifiques (`<image>` et `<commentaire>`).

Fichier `diapositive.xml` :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <visionneuse>
        <diapo>
              <image>photo1.jpg</image>
              <commentaire>Idéal pour créer un site dynamique</commentaire>
```

```
        </diapo>
        <diapo>
            <image>photo2.jpg</image>
            <commentaire>PHP-MySQL et Dreamweaver 8</commentaire>
        </diapo>
        <diapo>
            <image>photo3.jpg</image>
            <commentaire>FLASH 8 et les jeux en réseau</commentaire>
        </diapo>
        <diapo>
            <image>photo4.jpg</image>
            <commentaire>10 jeux avec FLASH 8</commentaire>
        </diapo>
</visionneuse>
```

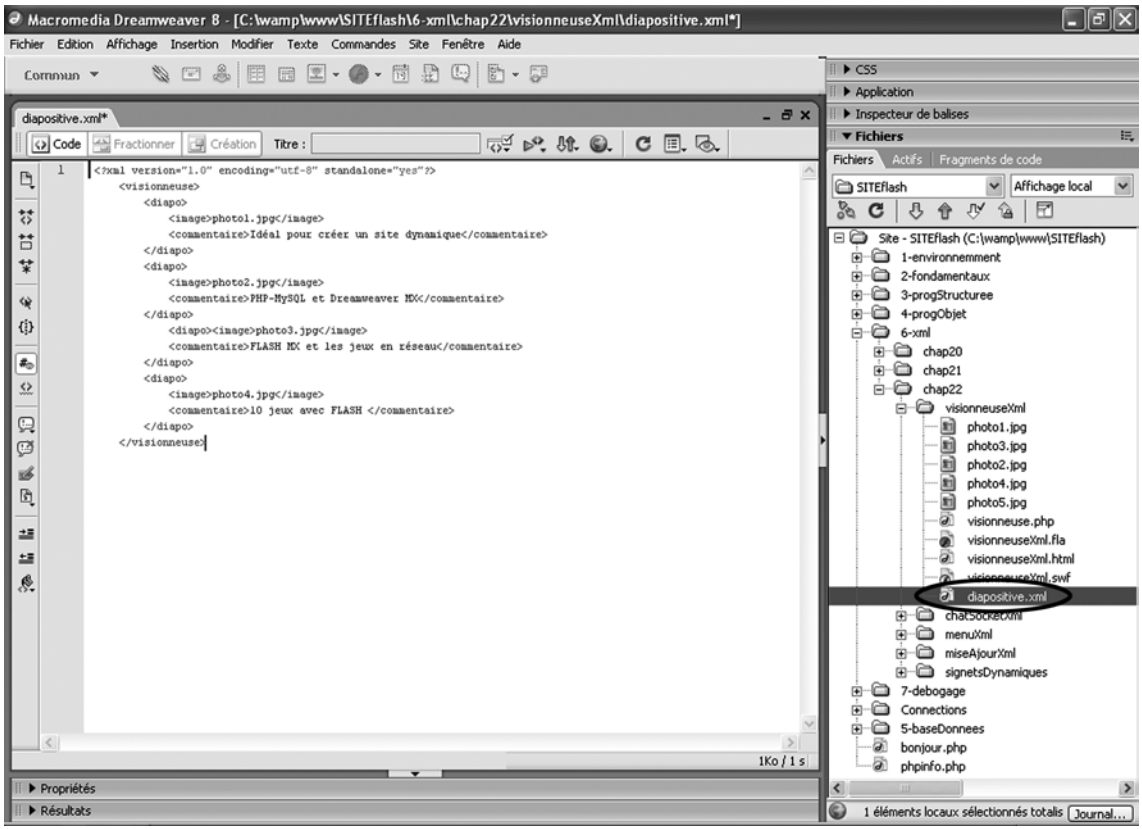

Figure 22-14

Création du fichier XML diapositive.xml avec Dreamweaver

1. Créez un nouveau document XML avec Dreamweaver et sauvegardez-le sous le nom `diapositive.xml` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local. Nous avons enregistré le fichier XML dans le répertoire `SITEflash/6-xml/chap22/visionneuseXml/` mais vous pouvez utiliser tout autre répertoire de votre choix dans la mesure où il se trouve dans le dossier `www/SITEflash/`.
2. Vérifiez que l'en-tête XML créé automatiquement est conforme à celui du code ci-dessous et modifiez-le si nécessaire :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

3. À la suite de l'en-tête, ajoutez une balise ouvrante `<visionneuse>`.
4. À l'intérieur de ce premier élément, créez autant de balises `<diapo></diapo>` que de diapositives à visionner.
5. À l'intérieur de chaque élément diapositive, ajoutez deux types de balises `<image></images>` et `<commentaire></commentaire>` puis renseignez les valeurs de l'élément image en saisissant l'URL du visuel à afficher (dans notre exemple, toutes les images se trouvent dans le même répertoire que l'application mais si ce n'est pas le cas, assurez-vous que les fichiers JPEG sont bien accessibles depuis ce répertoire).

À noter

Les photos doivent être au format JPEG sans optimisation (format standard). Leur dimension doit être en rapport avec l'espace d'affichage de l'application (prendre par exemple des photos d'une largeur de 200 pixels).

6. Clôturez le fichier en ajoutant une balise fermante `</visionneuse>` puis enregistrez-le.

Le fichier PHP

Le fichier `visionneuse.php` a pour fonction de récupérer le document XML modifié, de l'enregistrer dans le fichier `diapositive.xml` puis de construire et de retourner une réponse XML qui confirme que l'opération d'enregistrement s'est bien déroulée ou qui indique la nature du problème si ce n'est pas le cas.

1. Créez un nouveau document PHP dans Dreamweaver et sauvegardez-le sous le nom `visionneuse.php` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local (utilisez le même répertoire que celui dans lequel vous avez précédemment enregistré le fichier XML).
2. Saisissez les instructions `header()` suivantes afin que les informations retournées par le script ne soient pas mémorisées dans le cache des navigateurs ou des proxies (voir figure 22-15) :

```
<?php
header("Content-Type: text/xml");
//-------------------------Blocage du cache
header("Expires: Mon, 12 Jul 1995 02:00:00 GMT");
// Date d'expiration antérieure à la date actuelle
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
// Indique de toujours modifier la date
header("Cache-Control: no-cache, must-revalidate");
// no-cache pour HTTP/1.1
header("Pragma: no-cache");
// no-cache pour HTTP/1.0
```

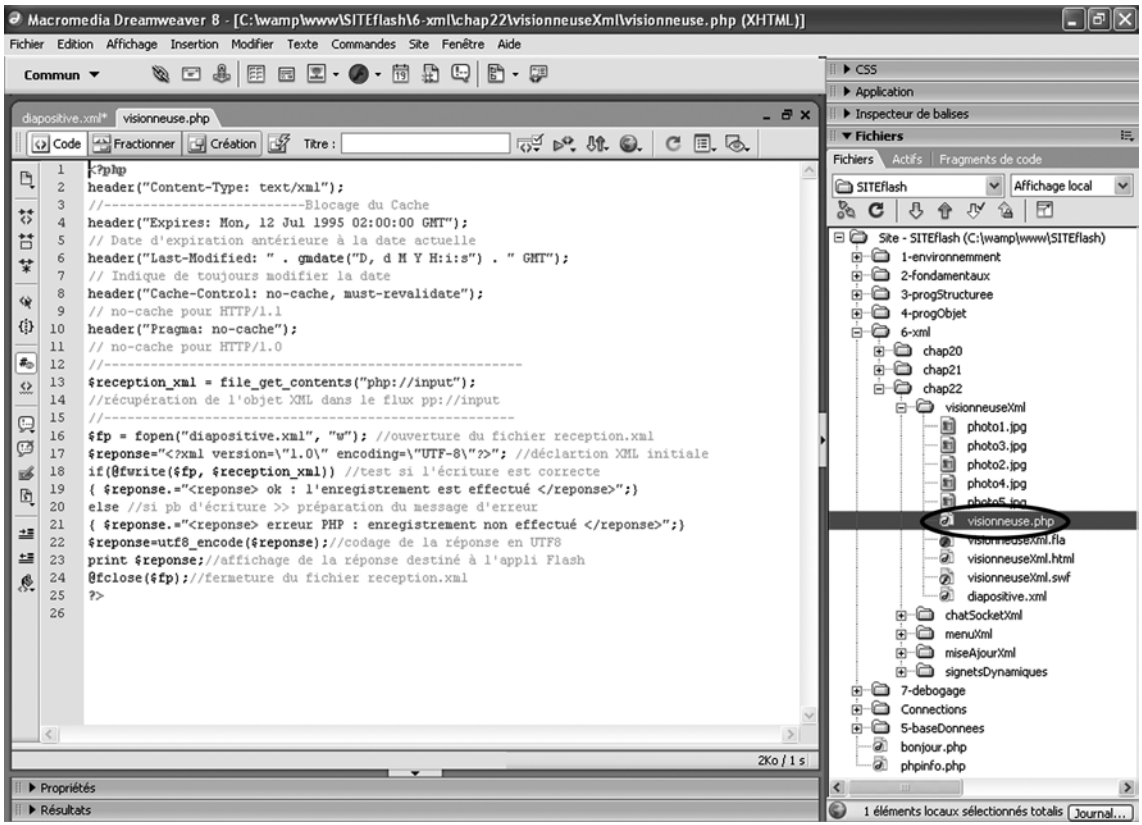

Figure 22-15

Création du fichier PHP visionneuse.php avec Dreamweaver

3. Sous ce code, ajoutez l'instruction qui sert à récupérer le document XML envoyé par l'application Flash. Tout le document XML sera ensuite sauvegardé dans la variable `$reception_xml` (Attention ! la récupération de données par le flux PHP ne peut être utilisée qu'avec la version 4.3 de PHP ou des versions ultérieures) :

```
$reception_xml = file_get_contents("php://input");
// Récupération de l'objet XML dans le flux php://input
```

4. Dans la dernière partie de code, on retrouve alternativement des instructions d'écriture du fichier XML (`fopen()`, `fwrite()` et `fclose()`) et des instructions destinées à construire la réponse XML dont le contenu sera conditionné par le succès de l'opération d'écriture. Si l'opération d'écriture du fichier XML se déroule correctement, le script affiche à l'écran le document XML suivant :

```
<reponse> ok : l'enregistrement est effectué </reponse>
```

5. Si l'écriture n'est pas possible, le document XML retourné signale une erreur :

```
<reponse> erreur PHP : enregistrement non effectué </reponse>
```

À noter

Pour éviter de perturber l'envoi du message d'erreur, les fonctions d'écriture seront précédées du caractère @ afin de neutraliser l'affichage automatique des messages d'erreur à l'écran.

```
// ÉCRITURE DU FICHIER DIAPOSITIVE.XML ET CONSTRUCTION DE LA RÉPONSE XML
$fp = fopen("diapositive.xml", "w"); // Ouverture du fichier reception.xml
$reponse="<?xml version=\"1.0\" encoding=\"UTF-8\"?>"; // Déclaration XML initiale
if(@fwrite($fp, $reception_xml)) // Teste si l'écriture est correcte
{ $reponse.="<reponse> ok : l'enregistrement est effectué </reponse>";}
else // Si problème d'écriture >> préparation du message d'erreur
{ $reponse.="<reponse> erreur PHP : enregistrement non effectué </reponse>";}
$reponse=utf8_encode($reponse); // Codage de la réponse en UTF8
print $reponse; // Affichage de la réponse destinée à l'application Flash
@fclose($fp); // Fermeture du fichier reception.xml
?>
```

Code complet du fichier `visionneuse.php` :

```
<?php
header("Content-Type: text/xml");
//--------------------------Blocage du cache
header("Expires: Mon, 12 Jul 1995 02:00:00 GMT");
// Date d'expiration antérieure à la date actuelle
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
// Indique de toujours modifier la date
header("Cache-Control: no-cache, must-revalidate");
// no-cache pour HTTP/1.1
header("Pragma: no-cache");
// no-cache pour HTTP/1.0
//-----------------------------------------------------
$reception_xml = file_get_contents("php://input");
// Récupération de l'objet XML dans le flux php://input
//-------------ÉCRITURE DU FICHIER DIAPOSITIVE.XML ET CONSTRUCTION DE LA RÉPONSE XML
$fp = fopen("diapositive.xml", "w"); // Ouverture du fichier reception.xml
$reponse="<?xml version=\"1.0\" encoding=\"UTF-8\"?>"; // Déclaration XML initiale
if(@fwrite($fp, $reception_xml)) // Teste si l'écriture est correcte
{ $reponse.="<reponse> ok : l'enregistrement est effectué </reponse>";}
else // Si problème d'écriture >> préparation du message d'erreur
{ $reponse.="<reponse> erreur PHP : enregistrement non effectué </reponse>";}
$reponse=utf8_encode($reponse); // Codage de la réponse en UTF8
print $reponse; // Affichage de la réponse destinée à l'application Flash
@fclose($fp); // Fermeture du fichier reception.xml
?>
```

Le document Flash

Le document Flash doit permettre à l'utilisateur d'afficher les différentes photos et de leur associer un commentaire.

Le scénario principal ne comporte qu'une seule image clé dans laquelle les différents symboles sont répartis sur plusieurs calques. Le code ActionScript est centralisé dans l'image 1 du calque Action (voir figure 22-16).

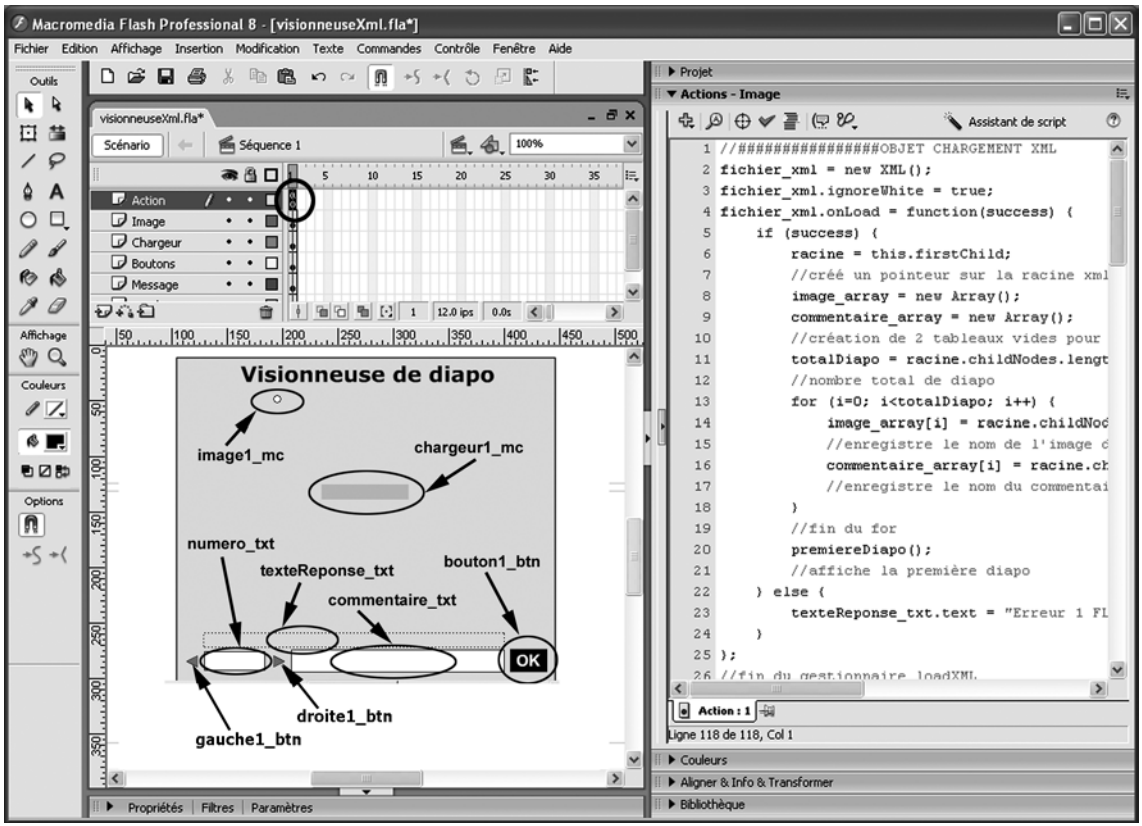

Figure 22-16

Création du document Flash visionneuseXml.fla

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `visionneuseXml.fla` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local (utilisez le même répertoire que celui dans lequel vous avez précédemment enregistré le fichier XML).

2. Créez six calques : Action, Image, Chargeur, Boutons, Message et Fond (voir figure 22-16).

3. Dans le calque Fond, délimitez l'interface à l'aide d'un élément graphique de votre choix et ajoutez un titre (`Visionneuse de diapositives`, par exemple).

4. Dans le calque Message, créez deux champs texte dynamiques (`numero_txt` et `texteReponse_txt`) destinés à afficher le numéro de la diapositive et la réponse renvoyée par le fichier PHP lors de l'enregistrement. Créez ensuite un champ texte de saisie (`commentaire_txt`) destiné à afficher le commentaire accompagnant chaque diapositive.

5. Dans le calque Boutons, créez un premier bouton de validation d'enregistrement OK et nommez son occurrence `bouton1_btn`. Créez ensuite deux boutons en forme de flèche afin de circuler d'une diapositive à l'autre (suivante ou précédente) et nommez-les `droite1_btn` et `gauche1_btn`.

6. Dans le calque Chargeur, créez un clip à partir d'un rectangle qui fera office de barre de chargement. Nommez son occurrence `chargeur1_mc`.

7. Dans le calque Image, créez une occurrence de clip vide que vous nommerez `image1_mc`. Ce clip est destiné à accueillir les images JPEG des différentes diapositives.

8. Dans le calque Action, saisissez les différentes parties de code ci-dessous dans l'image clé 1. La première partie est destinée à créer un objet XML nommé `fichier_xml` dans lequel sera chargé le contenu du fichier XML `diapositive.xml`. Le gestionnaire qui contrôle le chargement crée un pointeur `racine` qui correspond exactement à l'élément racine du fichier `diapositive.xml` (cela évite d'avoir à gérer l'élément Flash-root généré automatiquement par Flash dans l'objet XML). Ensuite, deux tableaux de variables (`image_array` et `commentaire_array`) sont déclarés afin d'accueillir les contenus des éléments `<image>` et `<commentaire>` de chaque diapositive. Une variable `totalDiapositives` est initialisée avec le nombre d'éléments `<diapo>` chargés. À la ligne suivante, cette même variable est utilisée dans l'expression de condition de la structure de boucle `for`. Cette structure de boucle permet d'affecter successivement les différentes images et commentaires aux deux tableaux précédemment créés. Une fois que la boucle a parcouru tous les éléments `<diapo>` chargés, une fonction `premiereDiapositive()` est appelée afin d'afficher la première diapositive de la série. Enfin, en cas de problème de chargement, la zone de texte `texteReponse_txt` affichera un message d'erreur :

```
//##################OBJET CHARGEMENT XML
fichier_xml = new XML();
fichier_xml.ignoreWhite = true;
fichier_xml.onLoad = function(succes) {
  if (succes) {
        racine = this.firstChild;
        // Crée un pointeur sur la racine xml du document source
        image_array = new Array();
```

```
        commentaire_array = new Array();
        // Création de deux tableaux vides pour les images et commentaires
        totalDiapositives = racine.childNodes.length;
        // Nombre total de diapositives
        for (i=0; i<totalDiapositives; i++) {
              image_array[i] = racine.childNodes[i].childNodes[0].firstChild.nodeValue;
               // Enregistre le nom de l'image dans le tableau image_array
               commentaire_array[i] = racine.childNodes[i].childNodes[1].firstChild.
               ➥nodeValue;
               // Enregistre le nom du commentaire dans le tableau commentaire_array
        }
        // Fin du for
        premiereDiapositive();
        // Affiche la première diapositive
  } else {
        texteReponse_txt.text = "Erreur 1 FLASH";
  }
};
// Fin du gestionnaire loadXML
```

9. La seconde partie du code a pour but de créer un autre objet XML, `reponse_xml`, qui sera exploité pour récupérer la réponse renvoyée au format XML par le script PHP. Cette réponse peut être soit un message de confirmation si l'enregistrement du fichier XML est correct, soit un message d'erreur dans le cas contraire. La déclaration de l'objet et le gestionnaire de chargement sont identiques à ceux de l'objet XML précédent, hormis le fait qu'ici on récupère le contenu du message pour l'affecter au champ texte `texteReponse_txt` afin que le message soit affiché dans l'interface (au-dessus des deux autres champs texte) :

```
//##################OBJET RÉPONSE XML
// Création de l'objet reponse_xml
var reponse_xml = new XML();
reponse_xml.ignoreWhite = true;
reponse_xml.onLoad = function(succes) {
  if (succes) {
        texteReponse_txt.text = this.firstChild.firstChild.nodeValue;
        chargement();
  } else {
        texteReponse_txt.text = "Erreur 2 FLASH";
  }
};
```

10. La partie suivante est dédiée au moteur de chargement des fichiers JPEG. Cette structure s'appuie sur un gestionnaire `onEnterFrame` qui scrute en permanence le chargement des images afin de faire évoluer la dimension de la barre de chargement tant que celui-ci est en cours, puis qui augmente progressivement l'alpha des images à partir du moment où elles sont complètement chargées :

```
//######################MOTEUR DU CHARGEUR
_root.onEnterFrame = function() {
  chargeur1_mc._visible = true;
  chargeTotale = image1_mc.getBytesTotal();
  chargeActuelle = image1_mc.getBytesLoaded();
  if (chargeActuelle != chargeTotale) {
        _root.chargeur1_mc._xscale = 100*chargeActuelle/chargeTotale;
        // Augmente progressivement la taille du voyant pendant le chargement
  } else {
        chargeur1_mc._visible = false;
        if (image1_mc._alpha<100) {
              image1_mc._alpha += 10;
          // Augmente progressivement l'alpha de la photo dès qu'elle est complètement
              ➥chargée
        }
  }
};
```

11. Ajoutez à la suite le code ci-dessous afin d'assurer la gestion du bouton de mémorisation (bouton OK), du bouton Diapositive suivante et du bouton Diapositive précédente :

```
//#####################GESTION DES BOUTONS
bouton1_btn.onRelease = function() {
  memorisation();
};
gauche1_btn.onRelease = function() {
  precedentDiapositive();
};
droite1_btn.onRelease = function() {
  suivantDiapositive();
};
```

12. La fonction de chargement est exécutée dès l'appel de l'animation et à chaque fois qu'un commentaire est mémorisé. Elle initialise à zéro l'indice `n` qui contrôle les diapositives et déclenche le chargement du fichier `diapositive.xml` dans l'objet `fichier_xml` grâce à la méthode `load()` :

```
//#####################FONCTION CHARGEMENT
function chargement() {
  n = 0;
  // Initialisation de l'indice de diapositive
  fichier_xml.load("diapositive.xml");
}
```

13. La fonction de mémorisation est appelée par une action sur le bouton OK (bouton1_btn). Elle permet d'affecter le contenu du commentaire modifié à la valeur de l'élément `<commentaire>` correspondant à la diapositive en cours de consultation (spécifiée par l'indice `n`). La méthode `sendAndLoad()` est ensuite appelée afin d'envoyer le document modifié vers le script `visionneuse.php` et d'assurer la récupération de la réponse de ce même script dans le second objet XML `reponse_xml`. Enfin, l'ancien objet `fichier_xml` est supprimé avant d'être recréé avec le contenu téléchargé depuis le fichier XML modifié (chargement effectué par l'appel de la fonction `chargement()` appelée à la fin du gestionnaire de l'objet `reponse_xml`) :

```
//####################FONCTION MÉMORISATION
// Mémorisation du commentaire d'une diapositive
function memorisation() {
  fichier_xml.firstChild.childNodes[n].childNodes[1].firstChild.nodeValue
  ➥= _root.commentaire_txt.text;
  fichier_xml.sendAndLoad("visionneuse.php", reponse_xml);
  delete fichier_xml; // Suppression de l'objet XML
}
```

14. Les autres fonctions destinées à la gestion des diapositives sont rassemblées à la fin du script de l'image clé (voir le code ci-dessous). Elles permettent de passer à la diapositive suivante lors d'un clic sur la flèche de droite (`suivantDiapositive()`), de revenir à la diapositive précédente lors d'un clic sur la flèche de gauche (`precedentDiapositive()`) ou encore de retourner à la première diapositive après l'enregistrement d'un commentaire ou lors de la première ouverture de l'application Flash (`premiereDiapositive()`). Pour chacune de ces trois fonctions, le déroulement est semblable :

   – l'alpha du clip conteneur est initialisé à zéro ;

   – l'image est chargée à l'aide d'une méthode `loadMovie()` ;

   – le texte du commentaire de la diapositive est affecté à la zone texte `commentaire_txt` ;

   – l'affichage du numéro de la diapositive est assuré par l'appel de la fonction `numeroDiapositive()` placée à la fin de cette partie de code :

```
//###############TOUTES LES FONCTIONS GESTION DES DIAPOSITIVES
function suivantDiapositive() {
  texteReponse_txt.text = '';
  // Initialisation zone réponse
  if (n<(totalDiapositives-1)) {
        n++;
        if (chargeActuelle == chargeTotale) {
              image1_mc._alpha = 0;
              image1_mc.loadMovie(image_array[n], 1);
```

```
                commentaire_txt.text = commentaire_array[n];
                numeroDiapositive();
          }
  }
}
//---------------------------------------
function precedentDiapositive() {
  if (n>0) {
        n--;
        image1_mc._alpha = 0;
        image1_mc.loadMovie(image_array[n], 1);
        commentaire_txt.text = commentaire_array[n];
        numeroDiapositive();
  }
}
//---------------------------------------
function premiereDiapositive() {
  if (chargeActuelle == chargeTotale) {
        image1_mc._alpha = 0;
        image1_mc.loadMovie(image_array[0], 1);
        commentaire_txt.text = commentaire_array[0];
        numeroDiapositive();
  }
}
//---------------------------------------
function numeroDiapositive() {
  num = n+1;
  numero_txt.text = num+" / "+totalDiapositive;
}
```

15. Enfin, à la dernière ligne du script, se trouve l'appel de la fonction de chargement initial (`chargement()`) qui permet de charger dans l'objet `fichier_xml` le contenu du fichier `diapositive.xml` dès l'ouverture de l'application Flash :

```
//############################APPEL FCT
chargement();
// Appelle la fonction au début de l'application
```

16. Une fois que vous avez terminé de saisir le script dans l'image clé, enregistrez votre document Flash et publiez-le dans le même répertoire. Vous pouvez désormais passer dans le Web local pour tester le fonctionnement de l'ensemble du système (voir figure 22-17). Si vous installez cette application sur un serveur distant, assurez-vous que le fichier `diapositive.xml` dispose bien

des droits d'écriture (utilisez si besoin CHMOD pour les modifier) et que la version du PHP est ultérieure à la version 4.3 (utilisez si besoin la fonction `phpinfo()` pour ce faire) afin que l'instruction de récupération du flux PHP puisse être interprétée correctement (`file_get_contents("php://input")`).

Figure 22-17

Test de la visionneuse de diapositives dans le Web local

Interface Flash-XML-PHP-MySQL

Dans le chapitre 18 sont présentés des interfaces Flash-PHP-MySQL utilisant la classe `LoadVars` et ses méthodes pour récupérer des informations retournées par une base de données. Cependant, les applications de cette technique d'interfaçage restent limitées à des échanges de petites quantités de variables simples.

Une autre solution consiste à utiliser la classe XML couplée à un script PHP qui a pour fonction de mettre les informations issues de la base de données au format XML. Les informations extraites de la base de données peuvent ainsi conserver leur structure grâce à une construction adaptée du document XML. D'autre part, comme pour toutes les informations récupérées par Flash au format XML, il est beaucoup plus facile d'en extraire les données désirées à l'aide des nombreuses méthodes de la classe XML.

Avec cette technique, le XML n'est plus exploité comme une solution d'archivage des données qui serait une alternative à l'utilisation d'une base de données, mais comme une solution de transfert parfaitement adaptée à la structure d'une base MySQL.

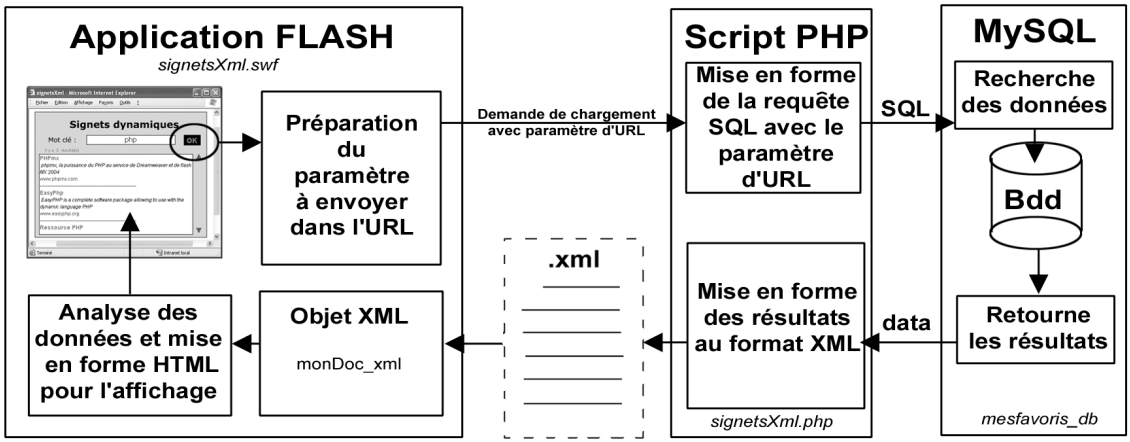

Figure 22-18

Principe de l'interface Flash-XML-PHP-MySQL utilisée pour l'application du système de signets dynamiques

Système de signets dynamiques

Nous allons utiliser cette technique d'interfaçage pour réaliser un système de signets dynamiques.

Les signets seront archivés dans la table d'une base de données MySQL. La mise à jour des données dans une base MySQL ayant déjà été traitée à plusieurs reprises (revoir le chapitre 18), nous ne

détaillerons pas l'actualisation ou l'ajout des signets dans une base mais cette fonctionnalité pourra être facilement intégrée au système dans un second temps.

Un script PHP assure la jonction entre la base MySQL et l'application Flash. Lorsque l'application Flash lui envoie (en paramètre d'URL) un mot-clé à rechercher parmi les noms des signets, il l'intègre dans une requête SQL qu'il soumet à la base de données. En sens inverse, lorsque la base de données lui retourne le résultat dans un jeu d'enregistrements, il construit une structure de document XML dans laquelle les informations de chaque enregistrement sont intégrées. Dès que ce document est terminé, il est affiché dans le navigateur afin que l'application Flash puisse le charger dans un objet XML adapté (voir figure 22-18).

Une fois les données disponibles dans l'application Flash, il suffit d'utiliser les procédures d'analyse XML présentées dans le chapitre précédent pour en extraire les informations. Celles-ci seront ensuite mises en forme (au format HTML afin de pouvoir créer des liens hypertextes) avant d'être affichées. Le contrôle des erreurs éventuelles et le nombre de signets trouvés seront affichés dans un champ texte situé au-dessus de la zone d'affichage des résultats.

La base de données MySQL

La base de données MySQL utilisée dans cette application se nomme `mesfavoris_db`. Elle stocke les paramètres de chaque signet et permet d'effectuer une recherche du ou des signets correspondants. Elle est constituée d'une seule table nommée `signets` qui comporte elle-même quatre champs : une clé primaire (`ID`), l'URL de chaque signet, le nom et la description du signet.

1. Création de la base de données : ouvrez le gestionnaire de base de données phpMyAdmin (depuis le manager de Wamp, sélectionnez l'option `phpMyAdmin`). Dans le champ de saisie du cadre central, saisissez le nom de la nouvelle base `mesfavoris_db`, puis cliquez sur le bouton Créer (revoir si nécessaire le chapitre 16 dédié à phpMyAdmin).
2. Création de la table `signets` : sélectionnez la base de données `mesfavoris_db` dans le menu déroulant du cadre de gauche. Dans la partie droite, sélectionnez l'onglet Structure puis, au niveau de la rubrique Créez une nouvelle table, saisissez `signets` (nom de la table à créer) dans le champ nom puis le chiffre `4` pour préciser le nombre de champs à créer. Cliquez sur Exécuter afin d'accéder au formulaire de création des champs. Renseignez le formulaire comme indiqué par la figure 22-19 puis validez vos choix en cliquant sur le bouton Sauvegarder.

À noter

Une autre solution pour créer la table signets consiste à saisir (ou à coller) le code SQL indiqué ci-dessous dans le champ dédié aux requêtes de l'onglet SQL (revoir si besoin les techniques de restauration d'une base de données au chapitre 6). Si vous souhaitez utiliser cette méthode, un fichier nommé `signets.sql` comportant ce code SQL se trouve à la racine des ressources concernant cette application.

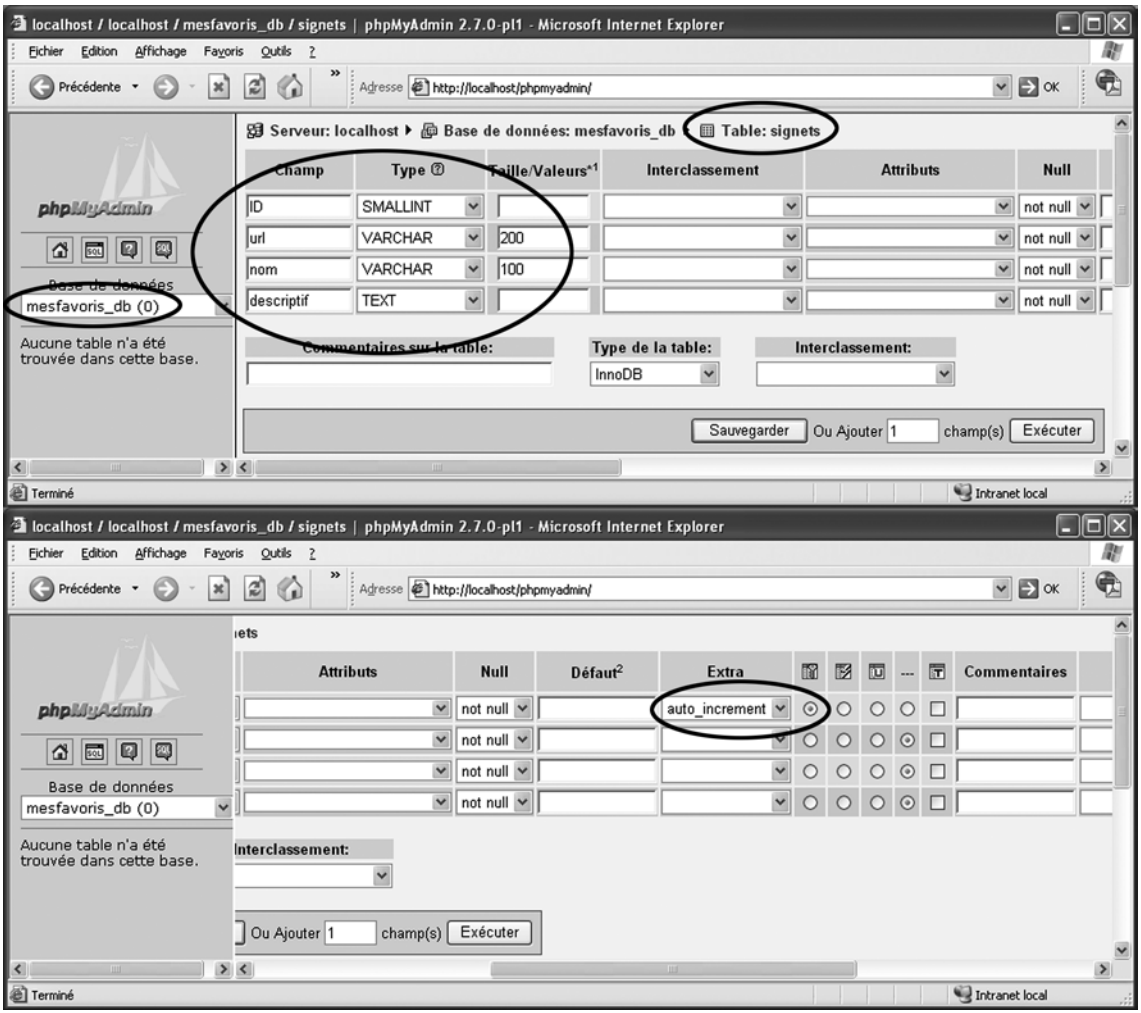

Figure 22-19

Création de la table signets dans le gestionnaire phpMyAdmin

3. Insérez des enregistrements dans la table : afin de pouvoir tester le système, vous devez insérer plusieurs enregistrements dans la table en vous assurant au préalable que les URL sont bien valides (voir les exemples d'enregistrement de la figure 22-20). Une fois que vous aurez ajouté vos signets dans la base, fermez le gestionnaire et passez à la création du fichier PHP.

Code SQL correspondant à la création de la table signets :

```
CREATE TABLE `signets` (
  `ID` smallint(6) NOT NULL auto_increment,
  `url` varchar(200) NOT NULL default '',
```

```
  `nom` varchar(100) NOT NULL default '',
  `descriptif` text NOT NULL,
  PRIMARY KEY  (`ID`)
) TYPE=MyISAM AUTO_INCREMENT=6 ;
```

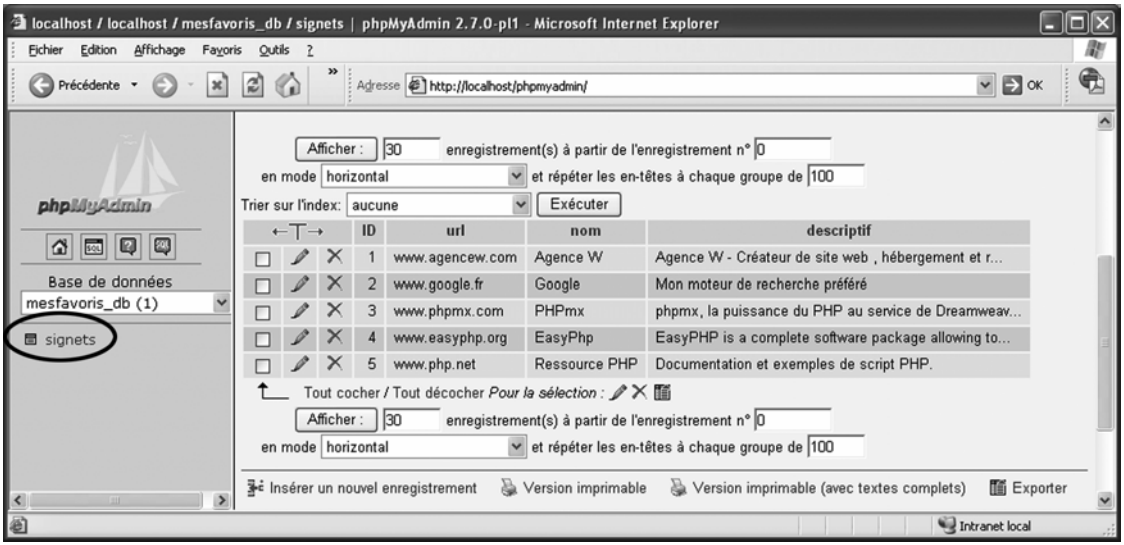

Figure 22-20

Exemple d'enregistrements de signets ajoutés dans la base de données pour les tests

Le fichier PHP

Le fichier `visionneuse.php` a pour fonction de récupérer le document XML modifié, de l'enregistrer dans le fichier `diapositive.xml` puis de construire et de retourner une réponse XML qui confirme que l'opération d'enregistrement s'est bien déroulée ou qui indique la nature du problème si ce n'est pas le cas.

1. Créez un nouveau document PHP dans Dreamweaver et sauvegardez-le sous le nom `signetsXml.php` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local (utilisez par exemple le sous-répertoire `/6-xml/chap22/signetsDynamiques/`).

2. Saisissez les instructions ci-dessous afin d'initialiser la variable `$clause` qui sera envoyée en paramètre d'URL par l'application Flash (voir figure 22-21) :

```
<?php
//----------------------Initialisation des variables
if(isset($_GET['clause'])) $clause=$_GET['clause']; else $clause="inconnu";
```

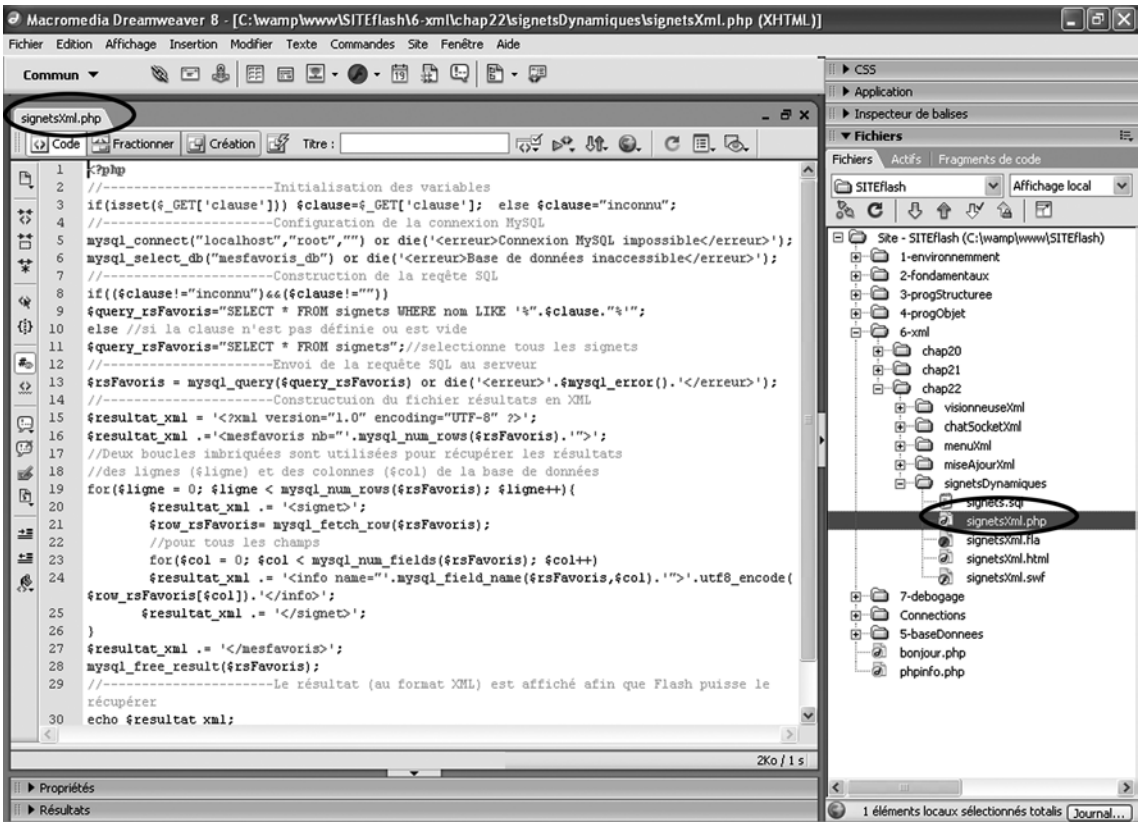

Figure 22-21

Création du fichier PHP signetsXml.php avec Dreamweaver

3. Sous ce code, ajoutez les instructions nécessaires à la connexion à la base MySQL.

```
//----------Configuration de la connexion MySQL
mysql_connect("localhost","root","") or die('<erreur>Connexion MySQL impossible</erreur>');
mysql_select_db("mesfavoris_db") or die('<erreur>Base de données inaccessible</erreur>');
```

À noter

Cette application étant isolée, nous n'utiliserons pas de fichier de connexion externe, mais intégrerons directement dans le script les instructions et paramètres de connexion. Pour les tests, nous utilisons l'utilisateur `root` (sans mot de passe) présent par défaut dans la base de données MySQL. Si vous décidez d'utiliser cette application en ligne, vous devrez créer un utilisateur spécifique et modifier le nom de l'utilisateur et son mot de passe dans ces paramètres.

4. La partie de code ci-dessous permet de construire la requête SQL en intégrant la variable `$clause`. L'intégration de cette variable est conditionnée par sa disponibilité. Ainsi si la variable n'existe pas ou est vide, la requête sera créée sans clause `WHERE` afin de sélectionner tous les enregistrements de la table :

```
//----------------------Construction de la requête SQL
if(($clause!="inconnu")&&($clause!=""))
$query_rsFavoris="SELECT * FROM signets WHERE nom LIKE '%".$clause."%'";
else // Si la clause n'est pas définie ou est vide
$query_rsFavoris="SELECT * FROM signets"; // Sélectionne tous les signets
```

5. Une fois la requête élaborée, elle doit être envoyée au serveur MySQL à l'aide de la fonction `mysql_query()` :

```
//----------------------Envoi de la requête SQL au serveur
$rsFavoris = mysql_query($query_rsFavoris) or die('<erreur>'.$mysql_error().'</erreur>');
```

6. Le jeu d'enregistrements retourné par la base MySQL doit ensuite être intégré dans un fichier XML afin que ses données puissent être analysées par l'application Flash. Le script suivant permet de construire ce fichier XML d'une manière dynamique. Il est constitué de deux boucles imbriquées qui permettent de créer des éléments `signets` en récupérant les lignes et les colonnes du jeu d'enregistrements (`$rsFavoris`) retourné par la base de données :

```
//----------------------Construction du fichier résultats en XML
$resultat_xml = '<?xml version="1.0" encoding="UTF-8" ?>';
$resultat_xml .='<mesfavoris nb="'.mysql_num_rows($rsFavoris).'">';
//Deux boucles imbriquées sont utilisées pour récupérer les résultats
// Des lignes ($ligne) et des colonnes ($col) de la base de données
for($ligne = 0; $ligne < mysql_num_rows($rsFavoris); $ligne++){
        $resultat_xml .= '<signet>';
        $row_rsFavoris= mysql_fetch_row($rsFavoris);
        // Pour tous les champs
        for($col = 0; $col < mysql_num_fields($rsFavoris); $col++)
       $resultat_xml .= '<info name="'.mysql_field_name($rsFavoris,$col).'">'.utf8_encode
        ➥($row_rsFavoris[$col]).'</info>';
       $resultat_xml .= '</signet>';
}
$resultat_xml .= '</mesfavoris>';
mysql_free_result($rsFavoris);
```

7. La dernière ligne du script PHP affiche le document XML précédemment créé afin qu'il puisse être récupéré puis analysé par l'application Flash :

```
//----------------------Le résultat (au format XML) est
// Affiché afin que Flash puisse le récupérer
echo $resultat_xml;
```

Code complet du fichier `signetsXml.php` :

```
<?php
//----------------------Initialisation des variables
if(isset($_GET['clause'])) $clause=$_GET['clause'];  else $clause="inconnu";
//----------------------Configuration de la connexion MySQL
mysql_connect("localhost","root","") or die('<erreur>Connexion MySQL impossible</erreur>');
mysql_select_db("mesfavoris_db") or die('<erreur>Base de données inaccessible</erreur>');
//----------------------Construction de la requête SQL
if(($clause!="inconnu")&&($clause!=""))
$query_rsFavoris="SELECT * FROM signets WHERE nom LIKE '%".$clause."%'";
else // Si la clause n'est pas définie ou est vide
$query_rsFavoris="SELECT * FROM signets"; // Sélectionne tous les signets
//----------------------Envoi de la requête SQL au serveur
$rsFavoris = mysql_query($query_rsFavoris) or die('<erreur>'.$mysql_error().'</erreur>');
//----------------------Construction du fichier résultats en XML
$resultat_xml = '<?xml version="1.0" encoding="UTF-8" ?>';
$resultat_xml .='<mesfavoris nb="'.mysql_num_rows($rsFavoris).'">';
// Deux boucles imbriquées sont utilisées pour récupérer les résultats
// des lignes ($ligne) et des colonnes ($col) de la base de données
for($ligne = 0; $ligne < mysql_num_rows($rsFavoris); $ligne++){
        $resultat_xml .= '<signet>';
        $row_rsFavoris= mysql_fetch_row($rsFavoris);
        // Pour tous les champs
        for($col = 0; $col < mysql_num_fields($rsFavoris); $col++)
        $resultat_xml.='<info name="'.mysql_field_name($rsFavoris,$col).'">'
        ➥.utf8_encode($row_rsFavoris[$col]).'</info>';
       $resultat_xml .= '</signet>';
}
$resultat_xml .= '</mesfavoris>';
mysql_free_result($rsFavoris);
//----------------------Le résultat (au format XML) est affiché afin que Flash puisse
➥le récupérer
echo $resultat_xml;
?>
```

Le document XML renvoyé par le script PHP

Dans cette interface, il n'existe réellement pas de fichier XML. Cependant, il est intéressant de connaître la structure du document XML renvoyé par le script PHP. Pour l'afficher, il suffit d'appeler le fichier PHP `signetsXml.php`. Vous obtiendrez alors un document XML comprenant tous les signets enregistrés dans la table. Si vous ajoutez le paramètre d'URL `clause` à la suite du nom du fichier comme dans l'exemple ci-dessous, vous obtiendrez la sélection des enregistrements correspondants formatés en XML (voir figure 22-22) :

```
signetsXml.php?clause=php
```

À noter

Ce test est aussi un excellent moyen de s'assurer du bon fonctionnement du script PHP…

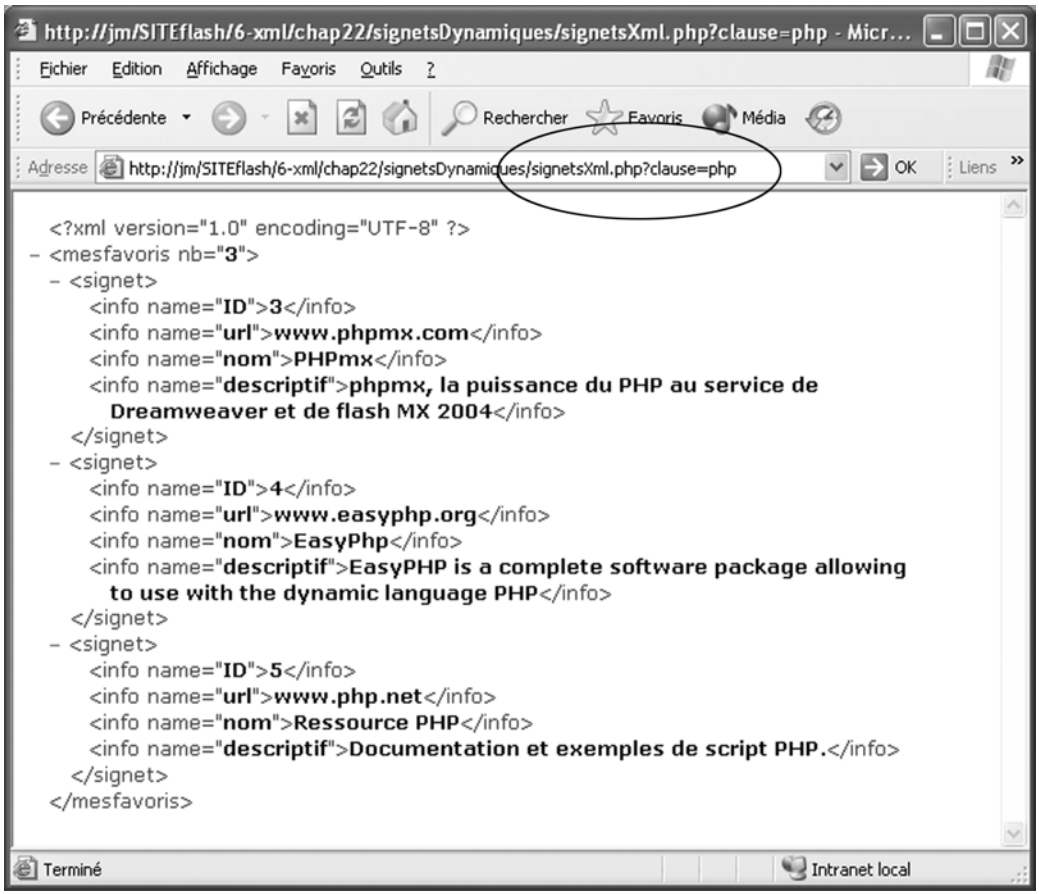

Figure 22-22

Document XML affiché lors de l'appel du fichier PHP signetsXml.php avec le paramètre d'URL clause=php

Le document Flash

Le document Flash permet à l'utilisateur de saisir le nom du signet (ou une partie de ce nom) et de consulter en retour une liste des signets. Si la fenêtre ne peut afficher tous les signets, deux boutons placés latéralement permettent de monter ou de descendre dans la liste afin de consulter les signets non visibles. Le champ de l'URL de chaque signet est paramétré pour ouvrir une nouvelle fenêtre de navigateur affichant le site du signet concerné par un simple clic (voir figure 22-24).

Le scénario principal ne comporte qu'une seule image clé dans laquelle les différents symboles sont répartis sur plusieurs calques. Le code ActionScript est centralisé dans l'image du calque `Action`.

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `signetsXml.fla` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local (utilisez le même répertoire que celui dans lequel vous avez précédemment enregistré le fichier PHP).

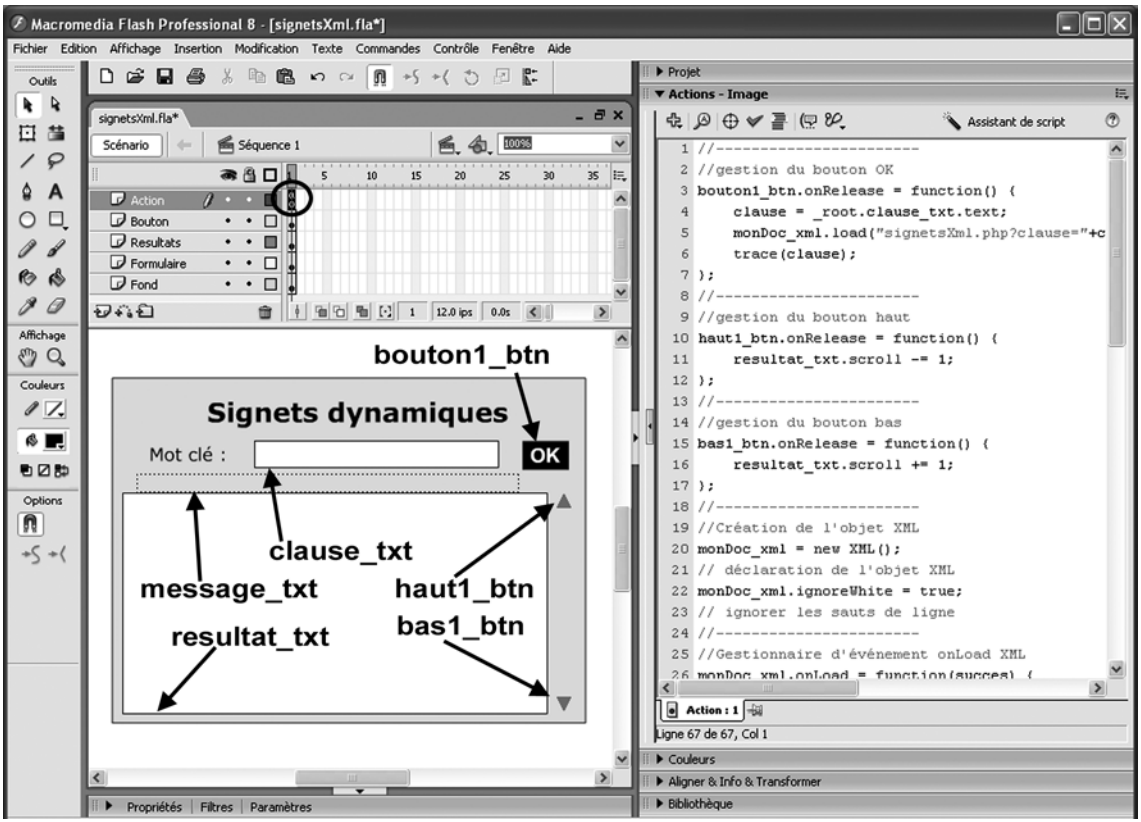

Figure 22-23

Création du document Flash signets.fla

2. Créez cinq calques : Action, Boutons, Resultats, Formulaire et Fond (voir figure 22-23).
3. Dans le calque Fond, délimitez l'interface avec un élément graphique de votre choix et ajoutez un titre (`Signets dynamiques`, par exemple).
4. Dans le calque Formulaire, créez un champ texte de saisie (`clause_txt`) pour saisir le nom du signet à rechercher.
5. Dans le calque Resultats, créez un premier champ dynamique multiligne d'occurrence `resultat_txt` et dont le nom de variable (champ `Var`) sera nommé `resultat`. Configurez ensuite son panneau Propriétés afin que ce champ puisse afficher du texte au format HTML. Créez un second champ dynamique d'occurrence `message_txt` placé au-dessus du premier afin de pouvoir afficher les messages retournés par le fichier PHP (le message retourné peut être soit le nombre d'enregistrements, soit un message d'erreur).
6. Dans le calque Boutons, créez un premier bouton de validation d'enregistrement OK et nommez son occurrence `bouton1_btn`. Créez ensuite deux boutons en forme de flèche afin de parcourir la liste des signets et nommez-les `haut1_btn` et `bas1_btn`.

7. Dans le calque Action, saisissez le code ci-dessous dans l'image clé 1. La première partie est destinée à la gestion des trois boutons de l'application. Le bouton OK permet de déclencher une action de chargement XML (méthode `load()`) en envoyant en paramètre d'URL la clause saisie par l'utilisateur. Les deux autres boutons (les deux flèches) permettent de gérer le scroll de la liste `resultat_txt` :

```
//-----------------------
// Gestion du bouton OK
bouton1_btn.onRelease = function() {
  clause = _root.clause_txt.text;
  monDoc_xml.load("signetsXml.php?clause="+clause);
};
//-----------------------
// Gestion du bouton haut
haut1_btn.onRelease = function() {
  resultat_txt.scroll -= 1;
};
//-----------------------
// Gestion du bouton bas
bas1_btn.onRelease = function() {
  resultat_txt.scroll += 1;
};
```

8. La partie suivante crée puis configure l'objet XML `monDoc_xml` afin de pas tenir compte des éventuels espaces dans le document XML qui sera chargé :

```
//-----------------------
// Création de l'objet XML
monDoc_xml = new XML();
// Déclaration de l'objet XML
monDoc_xml.ignoreWhite = true;
// Ignorer les sauts de ligne
```

9. La création de l'objet XML est suivie du gestionnaire `onLoad()`. Les deux premières instructions du gestionnaire permettent de créer un pointeur `racine` et de récupérer le nombre de signets dans une variable `nbsignets`. Après ces deux instructions, une structure de choix `if` permet d'initialiser la valeur du champ `message_txt` grâce à un texte qui indique le nombre de signets ou un message d'erreur (si l'élément racine retourné par le script PHP se nomme `erreur`). La suite concerne l'élaboration du résultat (au format HTML) qui sera affiché dans la zone de texte dynamique `resultat` :

```
// Gestionnaire d'événements onLoad XML
monDoc_xml.onLoad = function(succes) {
  if (succes) {
         var racine : Object = this.firstChild;
         // Pointe sur l'objet racine (mesfavoris)
         var nbsignet = racine.attributes.nb;
```

```
        // Récupération du nombre de résultats
        if (racine.nodeName == "erreur") {
               // Gestion des éventuelles erreurs
               message_txt.text = racine.firstChild.nodeValue;
         } else {
               // Si pas d'erreur, affiche le nombre de résultats
               message_txt.text = "Il y a  "+nbsignet+"  résultat(s)";
        }
        //--------------------Gestion du résultat
        if (nbsignet == 0) {
               // S'il n'y a aucun résultat
             resultat = "Il n'y a pas de réponse<br>Merci de renouveler votre requête
               ➥avec un autre mot clé";
        } else {
               // Sinon construction HTML des résultats
               resultat = "";
               for (var n = 0; n<racine.childNodes.length; n++) {
                   resultat += "<font color=\"#FF0000\" size=\"14\" Face=\"Arial\"><b>";
                   resultat += racine.childNodes[n].childNodes[2].firstChild.nodeValue;
                      // Nom du signet
                      resultat += "</b></font>";
                      resultat += "<br><i> ";
                   resultat += racine.childNodes[n].childNodes[3].firstChild.nodeValue;
                      // Descriptif du signet
                   resultat += "</i><br> <font color=\"#0000FF\" size=\"12\" Face=\"Arial
                   ➥\"> <a href='http://";
                   resultat += racine.childNodes[n].childNodes[1].firstChild.nodeValue;
                      resultat += "' target='_blank' >";
                      // Lien hypertexte de l'URL du signet
                   resultat += racine.childNodes[n].childNodes[1].firstChild.nodeValue;
                      // Affichage del'URL du signet
                      resultat += "</a></font><br>";
                   resultat += "----------------------------------------------------";
                       resultat += "<br>";
               } // Fin du for
        } // Fin du else
  } // Fin du if(succes)
}; // Fin du gestionnaire
```

10. Une fois que vous aurez saisi tout le code dans l'image clé, enregistrez votre animation et passez dans le Web local pour tester l'interfaçage complet. Pour ce faire, saisissez un mot-clé puis cliquez sur le bouton OK. Tous les signets dont le nom comporte le mot-clé doivent être affichés dans la zone des résultats (voir figure 22-24). Si vous cliquez sur l'URL d'un signet, la page Web correspondante doit s'afficher dans une nouvelle fenêtre.