11

Interfaçage Flash-PHP-Txt

Utiliser une interface Flash en interaction avec des scripts PHP permet de réaliser de nombreuses applications qui exploitent à la fois la convivialité des animations Flash et la puissance des scripts PHP. Ce premier type d'interfaçage permet d'envoyer et de charger des données dans un simple fichier texte via un script PHP adapté. Par la suite, toujours par l'intermédiaire de scripts PHP, nous aborderons l'interfaçage Flash-PHP-MySQL qui relie une application Flash avec les données d'une base MySQL afin de mémoriser les informations d'une animation Flash en dehors des limites de la session d'un internaute ou de récupérer des informations complexes depuis une base de données MySQL (se reporter au chapitre 18).

Pour illustrer le principe de ces échanges, nous allons vous présenter plusieurs modes d'interfaçage Flash-PHP-Txt utilisant différentes fonctions Flash.

Terminologie spécifique aux sources de données

Source de données

Dans Flash, on appelle source de données la localisation d'un ensemble de données externes qui pourront être chargées puis exploitées dans l'animation Flash. Une source de données peut être un simple fichier texte, un fichier XML ou encore la résultante de l'exécution d'un script PHP mettant au format d'URL des données issues d'une base de données. Pour que ces sources de données puissent être exploitées au sein d'une application Flash, elles doivent être formatées selon leur type et l'objet Flash qui aura en charge de les récupérer (par exemple, un fichier texte géré par la classe `loadVars` ou encore un fichier XML géré par la classe `XML`…).

Transfert de données : chargement et envoi

On appelle transfert de données l'action de transférer les valeurs d'un ensemble de données d'une entité à l'autre. Il peut être effectué depuis une source de données externe (informations au format texte ou XML mémorisées dans un fichier ou générées dynamiquement) vers une animation Flash (on parle alors de chargement). Il peut également être effectué depuis l'animation Flash vers une autre application (un script serveur par exemple ; on parle alors d'envoi de données).

Interfaçage entre deux applications

On appelle interfaçage un système assurant le transfert bidirectionnel des données entre deux entités (envoi et chargement). Ainsi, il peut y avoir des interfaçages entre une animation Flash et une application serveur (script PHP, par exemple), entre une animation Flash et une base de données MySQL (via un script PHP) ou encore entre une animation Flash et une structure XML. Selon les cas, ces interfaçages pourront être plus ou moins complexes.

Compléments techniques concernant les transferts de données

L'encodage UTF-8

L'UTF-8 est un type d'encodage utilisé par Unicode qui permet de gérer des flux de données quelle que soit la langue utilisée, ce qui évite les problèmes d'interprétation des accents ou autres caractères spéciaux spécifiques à chaque langue. Ce type d'encodage est utilisé par défaut dans les animations Flash. L'encodage usuel des données en langue française étant l'ISO-8859-1 (jeu de caractères `Europe occidentale, Latin-1`), il est souvent nécessaire de décoder (ou d'encoder) les données échangées avec une animation Flash.

Selon l'origine de la source de données, voici les actions à effectuer afin d'obtenir des données au format UTF-8 :

- Pour les données issues d'un fichier créé par un éditeur, il suffit d'enregistrer les fichiers texte (.txt) ou XML (.xml) en sélectionnant le format UTF-8 (Unicode). Les fichiers XML devront en outre commencer par une balise spécifiant l'encodage utilisé :

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

- Pour les fichiers générés par un script PHP, il faut utiliser la fonction `utf8_encode()`. Si ces données sont transmises par l'URL (méthode `GET`), il faut en outre utiliser la fonction `urlencode()` comme dans l'exemple ci-dessous :

```
echo "maDonnee=".urlencode(utf8_encode($maChaineIso));
```

Fonctions PHP pour l'encodage et le décodage UTF-8

Tableau 11-1. Syntaxe de la fonction utf8_encode()

utf8_encode()
Cette fonction PHP convertit une chaîne ISO-8859-1 en UTF-8
Syntaxe de la fonction : `utf8_encode(nomDeLaChaine_ISO);`
Exemple : `$maVariable=utf8_encode($chaineIso);` Dans cet exemple, la chaîne `$chaîneIso` est décodée en UTF-8 et enregistrée dans la variable `$maVariable` avant d'être envoyée à l'animation Flash.

Tableau 11-2. Syntaxe de la fonction utf8_decode()

utf8_decode()
Cette fonction PHP convertit une chaîne UTF-8 en ISO-8859-1
Syntaxe de la fonction : `utf8_decode(nomDeLaChaîne_UTF8);`
Exemple : `$maVariable=utf8_decode($chaîneUtf8);` Dans cet exemple, la chaîne `$chaîneUtf8` (issue, par exemple, d'un formulaire Flash) est décodée en ISO-8859-1 et enregistrée ensuite dans la variable `$maVariable`.

Fonctions AS pour la gestion de l'encodage

Tableau 11-3. Syntaxe de la fonction useCodePage()

System.useCodePage
Si vous affectez la valeur vrai (`true`) à ce paramètre dans la première image clé de votre scénario principal, le format des données ne sera plus le format par défaut UTF-8 mais l'ISO-8859-1. Si l'utilisation de l'UTF-8 est impossible dans votre application, vous pourrez quand même gérer des données importées au format ISO-8859-1 dans votre animation Flash en ajoutant cette instruction.
Syntaxe de l'affectation de ce paramètre : `System.useCodePage=true;`

Les méthodes GET et POST

Pour échanger des données entre une application client (une animation Flash ou un simple formulaire HTML) et une application serveur (un script PHP, par exemple), il faut utiliser des méthodes `HTTP` pour transférer les couples variable/valeur à l'application serveur afin qu'ils soient traités. Deux techniques différentes peuvent être utilisées : la méthode `GET` ou la méthode `POST`.

La méthode GET

Si vous utilisez la méthode `GET`, les couples variable/valeur seront ajoutés dans l'URL à la suite du nom du fichier cible selon une syntaxe spécifique (format d'URL). Si vous envoyez à un script `affiche.php` deux champs nommés `nom` et `message` depuis un formulaire HTML (ou depuis une animation

Flash) en utilisant la méthode GET, vous verrez apparaître l'URL suivante dans la zone d'adresse du navigateur :

```
affiche.php?nom=toto&message=bonjour
```

Le fait que toutes les données soient visibles dans l'URL empêche d'utiliser cette méthode pour envoyer des informations confidentielles. D'autre part, le nombre de données envoyées avec la méthode GET est limité à 255 caractères, ce qui représente une seconde contrainte. Cependant, la méthode GET peut être facilement construite à l'aide d'un script PHP, ce qui est un avantage (revoir les instructions de concaténation de PHP). En outre, l'URL de sa requête peut être mémorisée dans vos favoris (ce qui n'est pas le cas avec la méthode POST).

Tableau 11-4. Syntaxe de la méthode GET

Méthode GET
Méthode HTTP d'envoi de données dans l'URL
Syntaxe de la fonction : `NomDuFichierCible.php?nomVar1=valeur1& nomVar2=valeur2& nomVar3=valeur3`
Légende : Le signe ? placé après le nom du fichier cible introduit les couples variable/valeur. Dans les couples variable/valeur, la valeur est reliée à la variable par un signe =. Si plusieurs couples doivent être transmis, le séparateur & est utilisé entre chaque couple.
Remarque : L'utilisation de la méthode GET est très simple. Cependant, les données transmises sont visibles aux yeux de tous dans l'URL et le nombre de caractères est limité à 255.

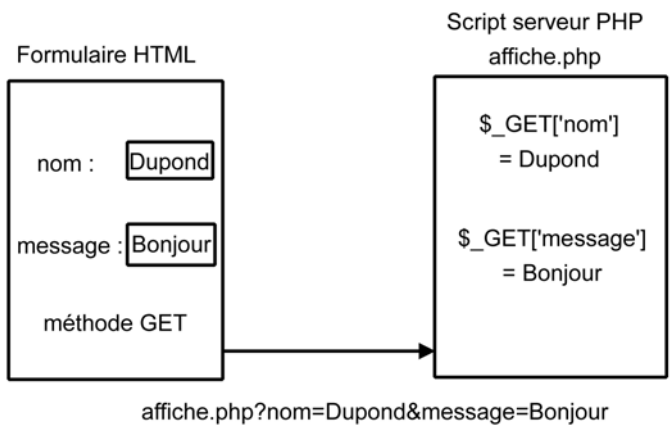


Figure 11-1
Envoi de données à l'aide de la méthode GET

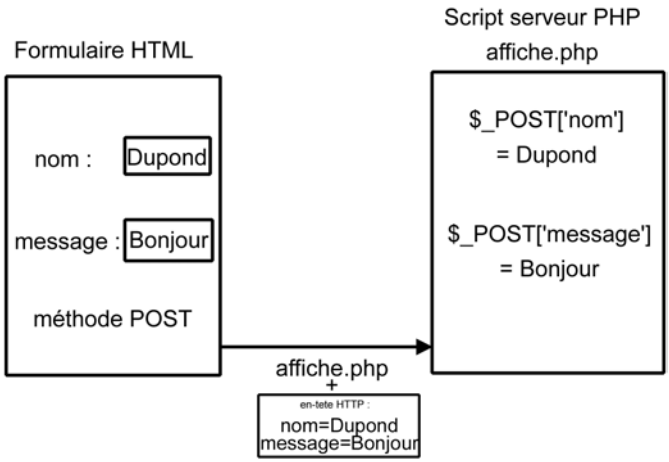
Les données transmises à un script PHP par la méthode GET sont ensuite disponibles depuis un tableau `$_GET[ ]`. Si nous reprenons l'exemple précédent, les valeurs des variables `nom` et `message` pourront être récupérées en utilisant les éléments de tableau `$_GET['nom']` et `$_GET['message']`.

La méthode POST

Si vous utilisez la méthode `POST`, les couples variable/valeur sont regroupés dans l'en-tête de la requête HTTP. Si vous envoyez à un script `affiche.php` deux champs nommés `nom` et `message` depuis un formulaire HTML (ou depuis une animation Flash) en utilisant la méthode `POST`, rien n'apparaît dans l'URL puisque les données sont envoyées dans l'en-tête de la requête. Cette méthode préserve vos données des regards indiscrets et n'est pas limitée à 255 caractères.

Figure 11-2

Envoi de données à l'aide de la méthode POST



Les données transmises à un script PHP par la méthode `POST` sont ensuite disponibles depuis un tableau `$_POST[ ]`. Si nous reprenons l'exemple précédent, les valeurs des variables `nom` et `message` pourront être récupérées en utilisant les éléments de tableau `$_POST['nom']` et `$_POST['message']`.

Fonctions PHP pour le transfert des données

Tableau 11-5. Syntaxe de la fonction urlencode()

`urlencode()`
Cette fonction PHP retourne une chaîne dont laquelle les caractères spéciaux (les lettre accentuées, les guillemets ou encore les caractères < et >, par exemple) sont remplacés par une séquence commençant par un caractère `%` suivi de deux chiffres hexadécimaux. Si la chaîne comporte des espaces, ils seront remplacés par des signes +.
Syntaxe de la fonction : `urlencode(nomDeLaChaîne);`

Exemple :
`$chaîne="Bonjour à tous";`
`echo '<a href="maPage.php?maVariable='.urlencode($chaîne).' ">clic ICI</a>';`
Dans cet exemple, la chaîne envoyée en paramètre dans l'URL sera codée de la manière suivante (le caractère `à` est remplacé par `%E0` et des signes + sont ajoutés pour lier les mots entre eux) :
`maVariable=Bonjour+%E0+tous`

Tableau 11-6. Syntaxe de la fonction urldecode()

urldecode()
Cette fonction PHP décode les séquences d'URL (au format %xx) et les remplace par leur valeur d'origine.
Syntaxe de la fonction : `urldecode(nomDeLaChaîne);`

Exemple :
`$chaîne=" Bonjour+%E0+tous";`
`echo urldecode($chaîne);`
Dans cet exemple, la chaîne est décodée et affiche le texte suivant :
`Bonjour à tous`

Les problèmes de cache

Dans vos futures applications Flash dynamiques, vous serez certainement confronté à des problèmes liés au cache du navigateur (surtout avec IE) ou aux serveurs proxy-cache. En effet, dès qu'un fichier (simple fichier texte, fichier XML ou fichier généré par un script PHP) est chargé dans une animation Flash, il est également copié dans le cache du navigateur (ou dans des serveurs proxy-cache selon la configuration de votre réseau). Si vous rappelez cette ressource et que les données contenues dans le fichier ont été modifiées, vous risquez de ne pas voir ces modifications apparaître dans votre application Flash.

Pour résoudre ce problème, il existe plusieurs solutions.

Ajouter des balises meta dans la page HTML

Vous pouvez ajouter des balises `meta` dans l'en-tête de vos pages HTML pour indiquer au navigateur de ne pas stocker vos pages dans les répertoires caches comme il le fait habituellement pour optimiser l'affichage des pages Web :

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta HTTP-EQUIV="expires" content="0">
<meta HTTP-EQUIV="Pragma" content="no-cache">
<meta http-equiv="Cache-Control" content="no-cache" />
<title>compteur</title>
</head>
```

La première ligne à ajouter dans la balise `head` (en gras) met à 0 la valeur `expires` (cette balise `META` permet d'indiquer une date de péremption pour la page HTML. Les navigateurs ne doivent pas conserver cette page dans leur cache au-delà de la période d'expiration, soit 0 dans notre cas). Les deux autres lignes interdisent la mise en cache dans le navigateur (pour HTTP 1.0 et HTTP 1.1).

Ajouter des header dans vos pages PHP

Si vos données sont générées par des scripts PHP, vous pouvez créer une en-tête à l'aide de l'instruction `header()` afin de bloquer la mise en cache des informations de la page comme dans le script ci-dessous (Attention ! Ce script doit être placé au début de la page PHP) :

```
//-------------------------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
```

Nous vous recommandons de placer ces lignes de code dans un fichier externe et de l'appeler au début de chaque fichier PHP. Par exemple, si vous enregistrez ces lignes de code dans un fichier nommé `blocageCache.php` (n'oubliez pas de placer les balises PHP au début et à la fin du fichier), utilisez l'instruction suivante pour l'inclure dans chaque fichier PHP :

```
require_once('blocageCache.php');
```

Utiliser une variable aléatoire dans Flash

Une autre solution consiste à ajouter une variable aléatoire à la fin du nom de fichier appelé par une méthode `LoadVariables`, `LoadVariablesNum` ou la classe `LoadVars`. La valeur de la variable étant différente à chaque appel, cela force le navigateur, qui croit qu'il s'agit d'un fichier différent, à récupérer la nouvelle version du fichier.

À noter

Dans notre exemple, nous avons utilisé `maVar` comme nom de variable aléatoire, mais vous pouvez utiliser le nom de votre choix, cela n'a aucune importance. D'autre part, vous ne pourrez pas tester ce système depuis l'environnement test de Flash, car la variable aléatoire est passée dans l'URL. Il est donc impératif de publier votre animation et de l'appeler ensuite depuis le WebLocal.

```
// Exemple avec la méthode LoadVariablesNum
LoadVariablesNum("monFichier.php?maVar="+GetTimer(),2);
// Exemple avec la classe LoadVars
monObjet_lv = new LoadVars();
monObjet_lv.load("monFichier.php?maVar="+GetTimer());
```

Restriction d'accès aux données d'un autre domaine

Depuis Flash 7, la restriction de sécurité concernant l'usage de sources de données situées sur un domaine différent de celui de l'animation est renforcée. En effet, avec Flash 6, toutes les sources de données situées dans le même superdomaine que le fichier SWF pouvaient être exploitées. Si le domaine du fichier SWF était `www.eyrolles.com`, le chargement des données pouvait être réalisé depuis une source de données placée dans des domaines `ftp.eyrolles.com` ou `data.eyrolles.com`. Depuis la version 7 de Flash Player, les domaines doivent être strictement identiques et seules des données situées dans les domaines `www.eyrolles.com` pourront désormais être chargées par le SWF.

Il existe heureusement des solutions pour contourner cette restriction. La première consiste à configurer un fichier `crossdomain` à la racine du domaine fournissant les données.

La seconde solution s'appuie sur une astuce qui consiste à confier la récupération des données à un fichier PHP (les fichiers PHP n'ayant pas de contrainte de sécurité de ce type).

Le fichier crossdomain

Il est possible d'obtenir une autorisation pour utiliser des données d'un autre domaine d'accès grâce à un fichier `crossdomain.xml` placé à la racine du domaine fournissant les données. Votre domaine doit figurer dans ce fichier de régulation si vous désirez accéder aux données.

Exemple de fichier `crossdomain.xml` autorisant les deux domaines `www.phpmx.com` et `www.agencew.com` à exploiter les données situées dans le domaine où a été placé ce fichier :

```
<cross-domain-policy>
        <allow-access-form-domain="www.phpmx.com" />
        <allow-access-form-domain="www.agencew.com" />
</cross-domain-policy>
```

À noter

Si vous désirez mettre vos données à la disposition de tous les domaines (et donc sans aucune restriction), utilisez le caractère * à la place du nom de domaine, comme dans le fichier `crossdomaine.xml` ci-dessous :

```
<cross-domain-policy>
        <allow-access-form-domain="*" />
</cross-domain-policy>
```

Connexion à un site distant à l'aide d'un fichier PHP

Voici une seconde solution qui vous permettra d'accéder à des fichiers de données distants situés sur un autre domaine. Le principe consiste à ne pas accéder aux données directement avec Flash mais de confier la récupération des données à un fichier PHP.

Dans un premier temps, l'adresse du fichier de données sera envoyée par le Flash à ce fichier PHP que nous nommerons liaison.php dans notre exemple. Ensuite, le fichier PHP récupérera les données du domaine externe puis les renverra au Flash pour les exploiter.

Dans notre exemple, le fichier de données `info.txt` se trouve à la racine du domaine `www.dineraparis.com` (notez que si le fichier de données était un fichier XML la démarche serait identique). Le fichier Flash qui désire exploiter les données se nomme `affiche.swf` et le fichier PHP qui assure la récupération des données s'appelle `liaison.php`.

Le fichier Flash `affiche.swf` doit comporter un premier objet `LoadVars`, qui servira à envoyer l'adresse du fichier de données à récupérer, et un second objet `LoadVars`, qui servira à la réception des données envoyées par le fichier PHP. Dans notre exemple, la récupération des données est déclenchée par un bouton bouton1_btn placé sur la scène.

`affiche.fla`

```
var envoi_lv:LoadVars=new LoadVars();
var reception_lv:LoadVars=new LoadVars();
bouton1_btn.onRelease = function() {
envoi_lv.adresse="http://www.dineraparis.com/info.txt";
reception_lv.onLoad=function(success) {
if(success) {
_root.info1= reception_lv.info1;
_root.info2= reception_lv.info2;
```

```
// Traitement des données récupérées…
}
}
}
envoi_lv.senAndAndLoad("liaison.php",reception_lv,POST);
```

Le fichier PHP `liaison.php` doit commencer par récupérer l'adresse du fichier de données. Ensuite, il accède aux données et les mémorise dans une variable `$donnees`. Enfin, il les renvoie au fichier Flash grâce à une simple instruction `echo()`. Notez que le fichier de données de notre exemple `infos.txt` contient deux données, `info1` et `info2`, déjà formatées pour être envoyées et interprétées par le fichier Flash.

`liaison.php`

```
<?php
// Récupération de l'adresse du fichier de données envoyé par le Flash
if(isset($_POST['adresse'])) $adresse=utf8_decode($_POST['adresse'])
// Récupération des données du fichier distant
$fichier=fopen($adresse,"r+");
$donnees=fgets($fichier,10);
fclose($fichier);
// Formatage et envoi des données à Flash
$donnees = utf8_encode($donnees);
echo $donnees;
?>
```

`infos.txt`

```
&info1=5555&info2=8888
```

Envoi de données de Flash vers PHP avec GetURL

Initialement, la méthode `GetURL()` est destinée à appeler une URL (une URL est un chemin absolu ou relatif qui permet de localiser une ressource) depuis une animation Flash. Cette méthode permet aussi de transmettre des données depuis Flash vers un script PHP d'une manière unidirectionnelle.

Lorsqu'on utilise `GetURL`, les informations sont envoyées avec la méthode `GET` (passage de variables dans l'URL) ou `POST` (passage de variables dans l'en-tête HTTP). Les variables transmises par l'une de ces deux méthodes sont disponibles dans le fichier PHP comme des variables envoyées par un formulaire HTML traditionnel (les valeurs récupérées sont stockées dans un tableau `$_GET[nomVar]` ou `$_POST[nomVar]` selon la méthode employée).

Tableau 11-7. Syntaxe de la méthode getURL()

`getURL()`
Appelle un document ciblé par son URL puis le charge dans une fenêtre spécifique. Si la méthode est précisée (paramètre variables : `GET` ou `POST`), les variables de l'animation seront transmises au document ciblé.
Syntaxe de la méthode : `getURL("url" [, "fenêtre" [, "variables"] ])`

Tableau 11-7. Syntaxe de la méthode getURL() *(suite)*

Légende	url : URL absolue ou relative du document ciblé. fenêtre : paramètre facultatif qui permet de préciser la fenêtre ou le cadre HTML dans lequel doit s'ouvrir le document ciblé. Il est possible d'indiquer le nom d'une fenêtre spécifique ou l'une des options suivantes : `_self` : pour indiquer la fenêtre courante `_blank` : pour indiquer une nouvelle fenêtre `_parent` : pour indiquer le parent de la fenêtre courante. `_top` : pour indiquer le premier niveau par rapport à la fenêtre courante. variables : permet de spécifier la méthode `HTTP` utilisée (`GET` ou `POST`). Si ce paramètre n'est pas précisé, aucune variable ne sera transmise au document ciblé. [xxx] : le code xxx est facultatif. (Attention ! Vous ne devez surtout pas saisir les crochets `[` et `]` dans le code.)

Pour illustrer l'échange de données depuis Flash vers un script PHP avec la fonction `getURL()`, nous vous proposons de créer un petit formulaire dans une animation Flash (`formulaire.fla`) et un fichier PHP (`affiche.php`) qui permet d'afficher les variables envoyées par le formulaire Flash.

Une zone texte portant le nom de variable `message` est créée dans l'interface Flash. La valeur saisie dans ce champ est ensuite transmise à un fichier PHP, dans lequel on récupère l'information dans une variable nommée `$message`.

Le document Flash

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `formulaire.fla` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local. Pour classer les différents scripts de cet ouvrage, nous avons enregistré le document Flash dans le répertoire `SITEflash/3-progStructuree/chap11/transfertGetUrl/` mais vous pouvez utiliser tout autre répertoire de votre choix dans la mesure où il se trouve dans le dossier `www/SITEflash/`.
2. Créez quatre calques : Fond, Message, Bouton et Action.
3. Personnalisez le calque Fond (texte d'information, couleur de fond…).
4. Ajoutez une zone de texte sur le calque Message. Sélectionnez le type Texte de saisie et saisissez le nom `message` dans le champ `Var` (voir figure 11-3).
5. Créez un symbole de type bouton (`envoi_btn`) puis placez une occurrence nommée `envoi1_btn` sur le calque Bouton (voir figure 11-4).
6. Saisissez ensuite le script suivant dans l'image clé 1 du scénario principal (voir figure 11-5). Dans ce script, le premier paramètre de la méthode `getURL()` indique que le fichier ciblé se trouve dans le même répertoire que l'animation et se nomme affiche.php. Le deuxième paramètre précise que le fichier devra s'ouvrir dans une nouvelle fenêtre (option `_blank`). Enfin, le troisième paramètre spécifie que la méthode `POST` sera utilisée pour le transfert des données :

```
bouton1_btn.onRelease=function() {
        getURL("affiche.php","_blank","POST");
  }
```

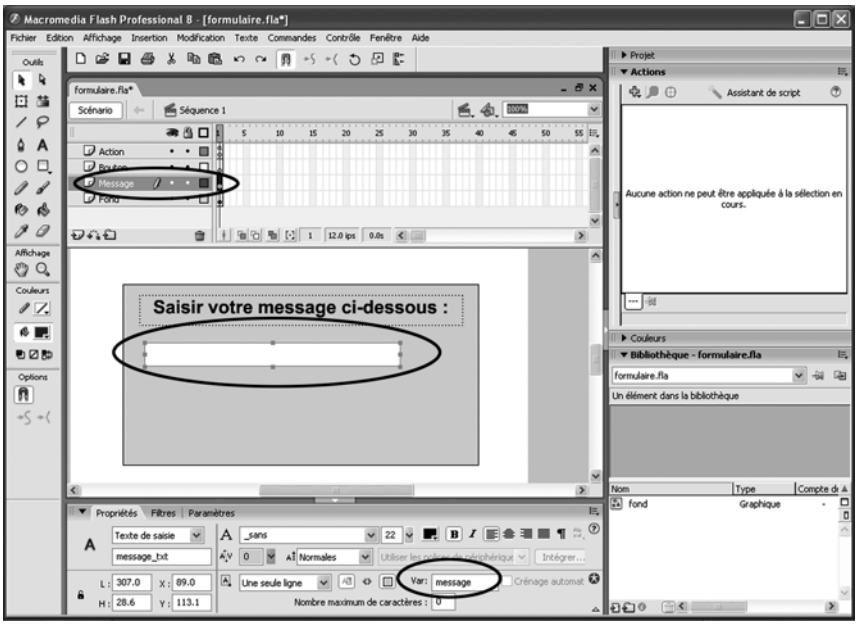


Figure 11-3

Création de la zone texte nommée message qui servira à la saisie du message envoyé au script PHP.

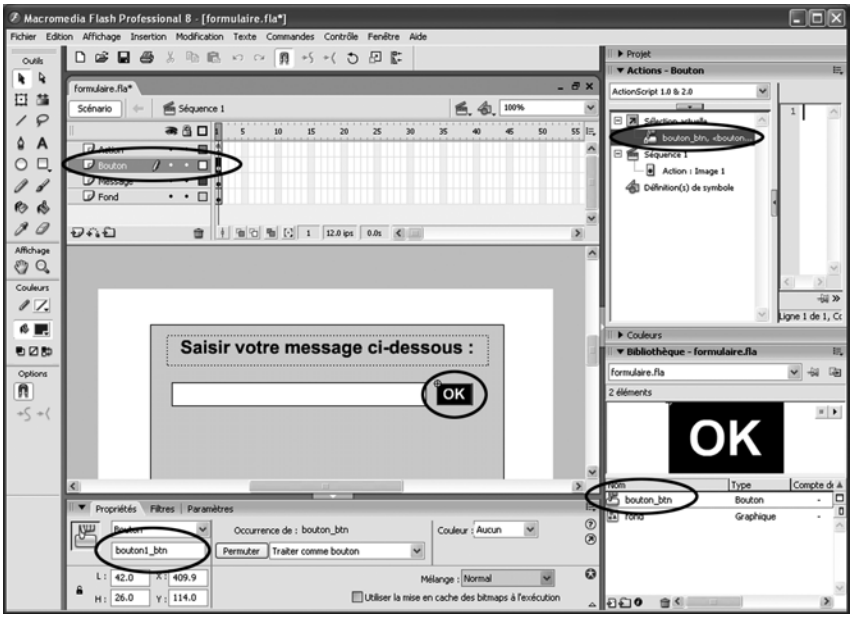


Figure 11-4

Création du bouton d'envoi du message. L'occurrence de ce bouton doit être nommée envoi1_btn.

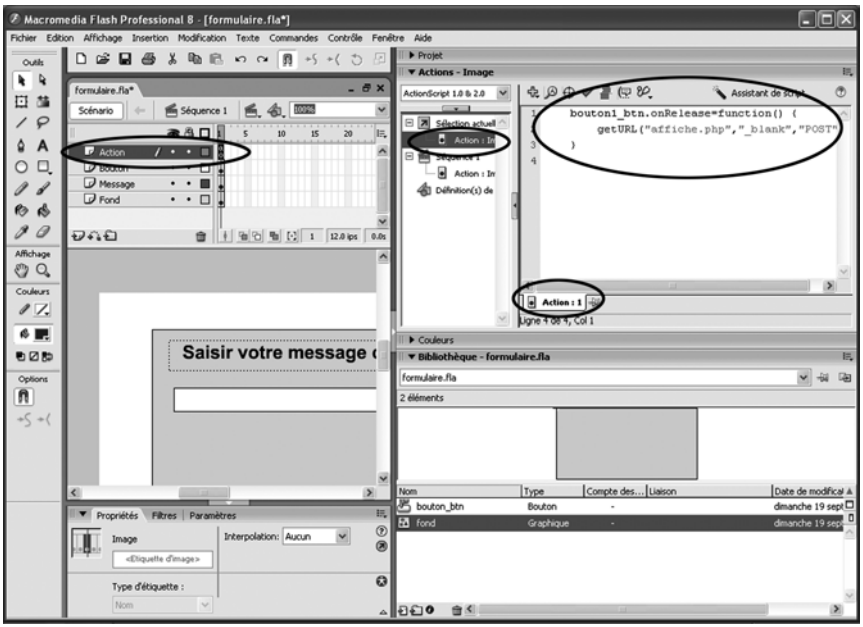


Figure 11-5
Script de gestion du bouton

7. Enregistrez votre fichier et publiez-le dans une page HTML sous le même nom que la source, soit `formulaire.htm`.

Le document PHP

Côté PHP, le script est très simple puisqu'il contient uniquement un test destiné à s'assurer de l'existence de la variable `$message`, suivi d'une instruction `echo` qui affiche la valeur de cette dernière.

1. Créez un nouveau document PHP (Fichier>Nouveau>Page dynamique>PHP) et sauvegardez-le sous le nom `affiche.php` dans le même répertoire que le document Flash.
2. Passez en mode Code et saisissez le script ci-dessous dans la fenêtre du document (voir figure 11-6).

```
<?
// Initialisation des variables
      if (!isset($_POST['message']))  $_POST['message']="";
      else
            {
                  $message=utf8_decode($_POST['message']);
                  echo "Vous avez un message : <br><h1> ".$message."</h1>";
                  //------Affichage de la variable "$message" à l'écran
             }
?>
```

3. Enregistrez le document PHP (`Ctrl + S`).

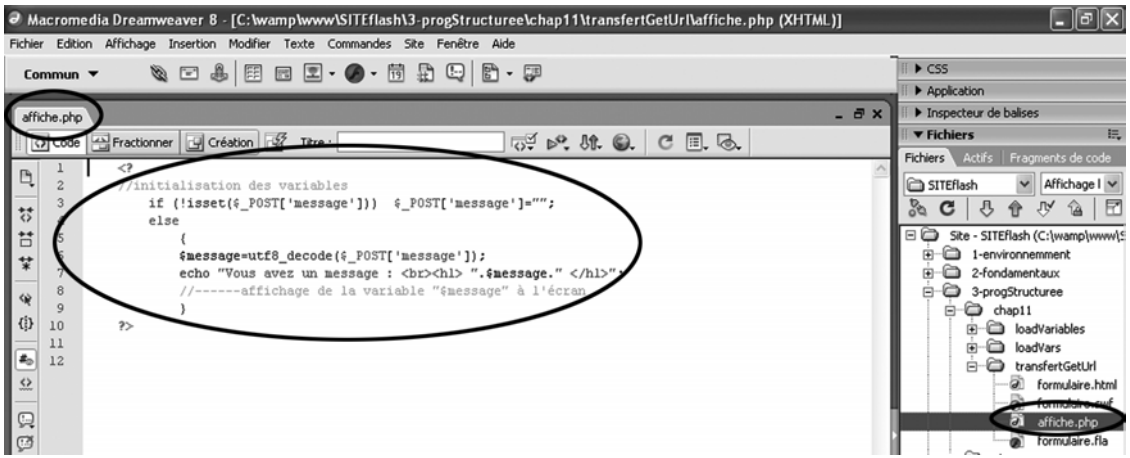


Figure 11-6

Le code PHP ci-dessus permet d'afficher le message envoyé par le formulaire Flash dans une nouvelle fenêtre du navigateur.

Test dans le WebLocal

Pour le test, il suffit d'appeler le fichier formulaire.htm depuis votre Web local (sélectionnez l'option `localhost` depuis le menu du manager de Wamp. Cliquez ensuite successivement sur les différents dossiers afin d'afficher le fichier `formulaire.htm`). Une fois le formulaire affiché à l'écran, saisissez un texte dans la zone message et cliquez sur le bouton OK. Le message doit alors s'afficher dans une nouvelle fenêtre (voir figure 11-7).

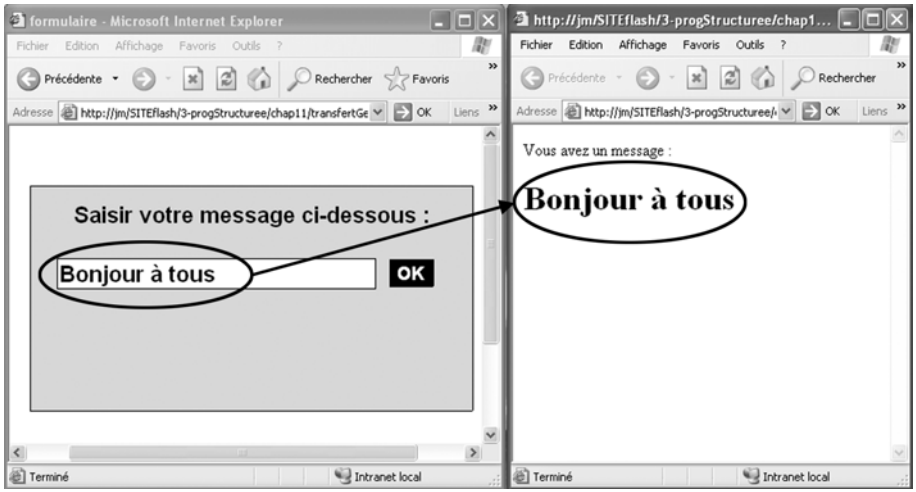


Figure 11-7

Test de l'envoi de données de Flash vers PHP avec la fonction getURL().

Chargement d'un fichier texte avec loadVariables() ou loadVariablesNum()

Ces deux méthodes se différencient principalement par l'élément dans lequel les données seront chargées. Pour `loadVariables()`, il s'agit d'un clip (identifié par son nom d'occurrence) alors que pour `loadVariablesNum()`, il s'agit d'un niveau (identifié par son numéro, exemple : 2 pour `_level2`), d'où la présence du suffixe `Num`.

Tableau 11-8. Syntaxe de la méthode loadVariables()

`loadVariables()`	
Charge les variables d'une source de données externe (simple fichier texte ou texte généré par un script PHP) et définit les variables en rapport dans le clip cible.	
Syntaxe de la méthode : `loadVariables("url" , "occurrence" [, "variables"] )`	
Légende	`url` : URL absolue ou relative de la source de données (fichier texte ou script PHP générant dynamiquement les variables). `occurrence` : chemin cible de l'occurrence du clip qui recevra les variables. `variables` : permet de spécifier la méthode `HTTP` utilisée (`GET` ou `POST`). Si aucune variable ne doit être envoyée depuis l'animation Flash, ce paramètre peut être omis. [xxx] : le code xxx est facultatif. (Attention ! Vous ne devez surtout pas saisir les crochets `[` et `]` dans le code.)
Remarques	Les noms des variables des champs de texte du document Flash doivent correspondre aux noms des variables de la source de données. L'URL de la source de données doit être du même domaine que le fichier SWF lorsque l'accès se fait par un navigateur Web. Pour plus d'information sur ces restrictions, consultez la partie concernant le fichier cross-domain au début de ce chapitre.

Exemples	`loadVariables("data.txt" , "_root.monClip_mc" )`
	Chargement d'un simple fichier texte `data.txt` dans le clip `monClip_mc` placé sur le scénario principal.
	`loadVariables("process/chargeur.php" , "_root.monClip_mc", "GET")`
	Chargement dans un clip `monClip_mc` d'une source de données générée dynamiquement par le script `chargeur.php` (placé dans le répertoire processeur) et transfert de toutes les variables du clip vers le même script en méthode `GET`.

Tableau 11-9. Syntaxe de la méthode loadVariablesNum()

`loadVariablesNum()`
Charge les variables d'une source de données externe (simple fichier texte ou texte généré par un script PHP) et définit les variables en rapport dans le niveau cible.
Syntaxe de la méthode : `loadVariables("url" , niveau [, "variables"] )`

Tableau 11-9. Syntaxe de la méthode loadVariablesNum() *(suite)*

Légende	`url` : URL absolue ou relative de la source de données (fichier texte ou script PHP générant dynamiquement les variables). `niveau` : numéro du niveau qui recevra les variables (par exemple : 2 pour le `_level2`). `variables` : permet de spécifier la méthode `HTTP` utilisée (`GET` ou `POST`). Si aucune variable ne doit être envoyée depuis l'animation Flash, ce paramètre peut être omis. [xxx] : le code xxx est facultatif. (Attention ! Vous ne devez surtout pas saisir les crochets `[` et `]` dans le code.)
Remarques	Les noms des variables des champs de texte du document Flash doivent correspondre aux noms des variables de la source de données. L'URL de la source de données doit être du même domaine que le fichier SWF lorsque l'accès se fait par un navigateur Web. Pour plus d'information sur ces restrictions, consultez la partie concernant le fichier `cross-domain` au début de ce chapitre.

Exemples	`loadVariablesNum("http://www.eyrolles.com/chargeur.php", 0)`
	Chargement d'une source de données générée dynamiquement par un script `chargeur.php` placé à la racine du site `www.eyrolles.com` dans le scénario principal de l'animation (`_level0`). Rappel : Dans ce cas, l'animation Flash doit elle aussi se trouver dans le domaine `www.eyrolles.com`.
	`loadVariablesNum("chargeur.php", 2 , "POST")`
	Chargement dans le niveau 2 (`_level2`) de l'animation d'une source de données générée dynamiquement par l'appel du script `chargeur.php`. Le troisième paramètre (variables) étant spécifié, toutes les variables du niveau concerné seront également envoyées au script `chargeur.php` par la méthode `POST`.

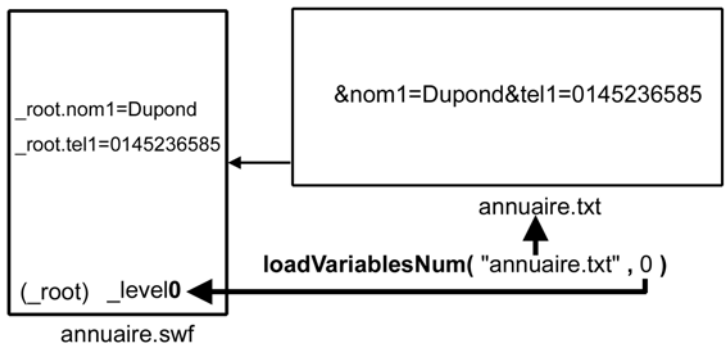


Figure 11-8
Utilisation de la méthode loadVariablesNum pour charger un fichier texte dans une animation Flash

Un annuaire Flash-Txt

Pour illustrer le chargement d'un simple fichier texte dans une animation Flash avec la méthode `loadVariables()`, nous vous proposons de créer un petit annuaire téléphonique. L'interface sera réalisée par une animation Flash (`annuaire.fla`) qui affichera les données (nom et téléphone des contacts) dans un champ texte. Les données utilisées seront issues d'un simple fichier texte (`annuaire.txt`) que nous réaliserons au préalable.

Le fichier texte

Avant de créer l'animation Flash, nous allons définir la structure et les valeurs du fichier de données au format `.txt`. Les informations transmises pour chaque contact de l'annuaire sont composées d'une variable `nom` suivi d'un numéro (exemple : `nom1`) et d'une variable `tel` suivi du même numéro (exemple : `tel1`). Pour illustrer le fonctionnement de l'animation, nous saisirons dix contacts séparés par un caractère `&` :

```
nom1=Dupond&tel1=0145235620&nom2=Vermont&tel2=0145683689&…
```

Après cette suite de couples variables/valeurs constituant les données à exploiter dans l'annuaire, nous ajouterons deux autres variables. La première indique le nombre de contacts envoyés, soit dix dans notre exemple (`nbre=10`). La deuxième clôture la chaîne de données et sera testée dans l'animation lors du chargement afin de s'assurer que toutes les variables ont bien été chargées (`fin=ok`). La fin de la chaîne de données ressemble à l'exemple ci-dessous :

```
…&nom10=Bidaut&tel10=0459932204&nbre=10&fin=ok&
```

Création du fichier `annuaire.txt` avec Dreamweaver :

1. Créez un nouveau document Texte à partir de Dreamweaver (menu Fichier>Nouveau>Catégorie Autre>Texte).
2. Saisissez les données selon l'exemple de la figure 11-9.
3. Enregistrez le fichier au format texte sous le nom `annuaire.txt` dans un sous-répertoire du dossier `www/SITEflash/` de votre serveur local (prendre par exemple le répertoire `www/SITEflash/3-progStructuree/chap11/loadVariables/`).

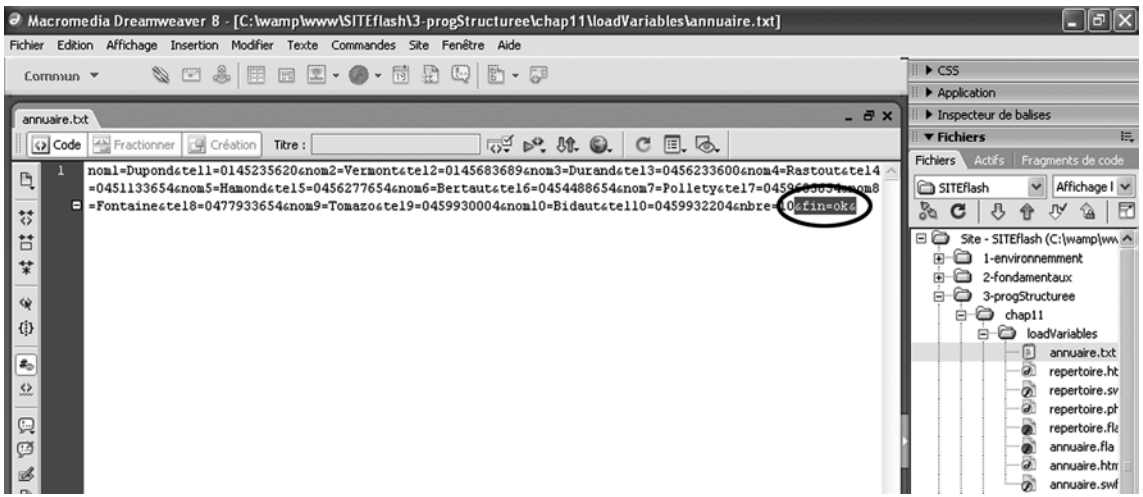


Figure 11-9
Création du fichier de données annuaire.txt avec Dreamweaver

Le document Flash

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `annuaire.fla` dans le même sous-répertoire du dossier `www/SITEflash/` que le fichier `annuaire.txt`.
2. Créez six calques : Fond, Chargement, Liste, Boutons, Action et Label.
3. Personnalisez le calque Fond en ajoutant un cadre qui contiendra les éléments de l'application et insérez une image clé dans l'image 17.
4. Dans le calque Chargement, ajoutez un texte pour indiquer le chargement en cours. Insérez ensuite une image clé vide dans l'image 10.
5. Dans le calque Liste, insérez une image clé dans l'image 10. Dans l'image 10, ajoutez sur la scène une zone de texte dynamique multiligne puis nommez son occurrence `liste_txt`. Saisissez le nom `liste` dans son champ variable.
6. Dans le calque Boutons, insérez une image clé dans l'image 10. Dans l'image 10, ajoutez sur la scène deux boutons. Nommez-les haut1_btn et bas1_btn. Ils serviront à faire défiler les contacts dans le champ texte `liste`.
7. Dans le calque Action, ajoutez l'instruction `loadVariablesNum` dans l'image clé 1 (voir figure 11-10). Le fichier `annuaire.txt` sera ainsi chargé dans le niveau 0 de l'animation (les variables de la source de données seront disponibles depuis le scénario principal de l'animation). Vous remarquerez que la source de données étant un simple fichier texte, nous n'avons pas indiqué de méthode (`POST` ou `GET`) afin que le chargement de données soit unidirectionnel et qu'aucune information ne transite de l'animation Flash vers la source.

   Code de l'image clé 1 du calque Action :

```
loadVariablesNum("annuaire.txt",0);
```

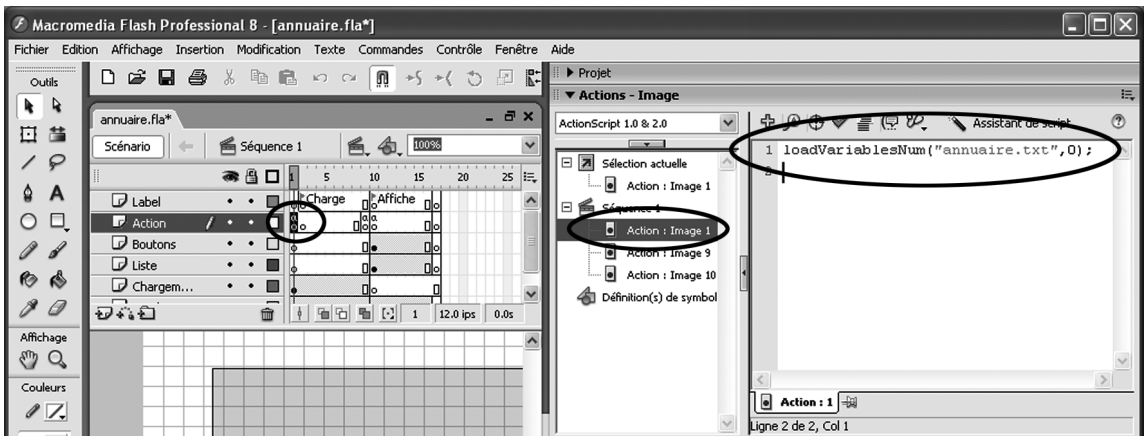


Figure 11-10

Script de chargement intégré dans l'image clé 1 du calque Action

8. Toujours dans le même calque, insérez ensuite une image clé vide dans l'image 2 puis une image clé dans l'image 9. Dans l'image 9, saisissez le code du test contrôlant la fin du chargement des données (voir figure 11-11). Tant que la valeur de la dernière variable (`fin="ok"`) ne sera pas disponible, l'animation bouclera sur l'étiquette Charge. Une fois toutes les données chargées, la tête de lecture de l'animation passera à l'étiquette Affiche.

   Code de l'image clé 9 du calque Action :

```
if(_root.fin=="ok") {
  gotoAndStop("Affiche");
}else {
  gotoAndPlay("Charge");
}
```

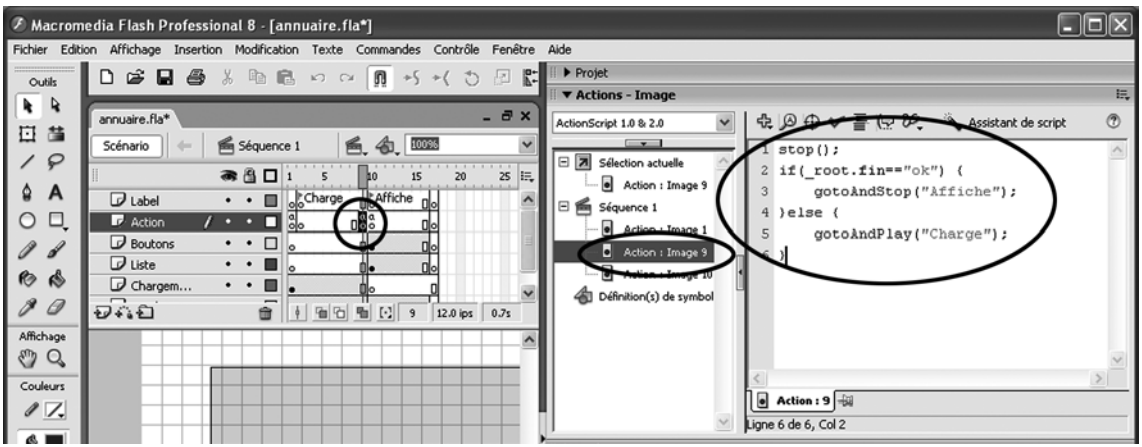


Figure 11-11

Script de test de la fin du chargement intégré dans l'image clé 9 du calque Action

9. Insérez ensuite une image clé vide dans l'image 10 et saisissez dans cette même image le code d'affectation des données à la variable `liste` ainsi que les deux scripts de gestion (utilisant des méthodes de gestionnaire d'événements) des boutons de défilement du texte (voir figure 11-12).

   Code de l'image clé 10 du calque Action :

```
//----------------------
// Affectation des valeurs à la liste
liste = "";
for (i=1; i<=_root.nbre; i++) {
      liste += _root["nom"+i]+"\t"+_root["tel"+i]+newline;
}
```

```
//-----------------------
// Gestion du bouton haut
haut1_btn.onRelease = function() {
      liste_txt.scroll -= 1;
};
//-----------------------
// Gestion du bouton bas
bas1_btn.onRelease = function() {
      liste_txt.scroll += 1;
};
```

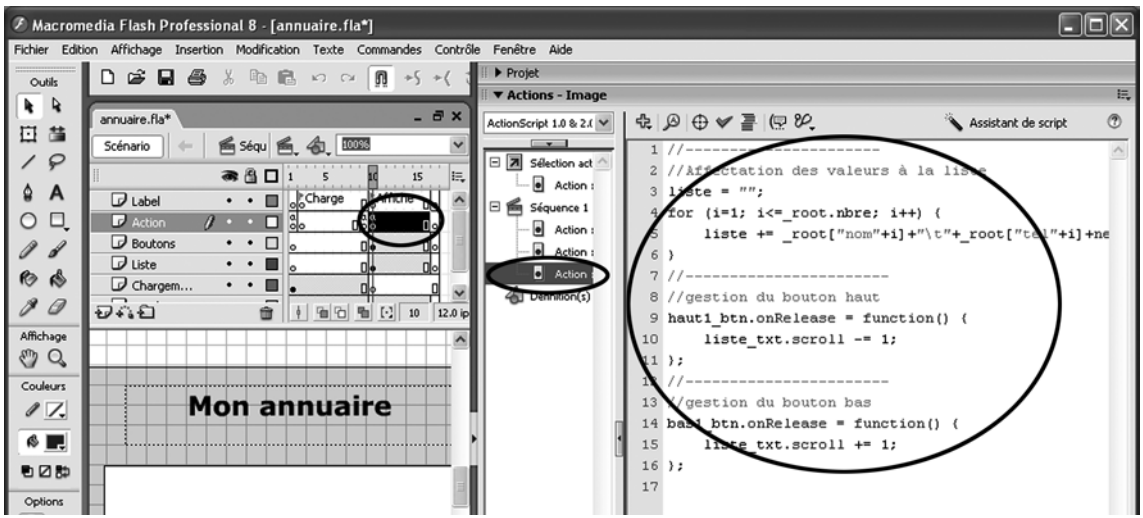


Figure 11-12

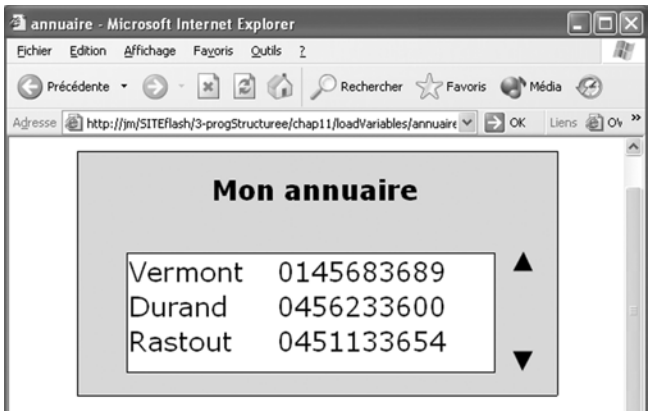
Dans l'image clé 10 du calque Action sont intégrés les scripts d'affectation des données à la variable du champ texte (liste) et de gestion des boutons de défilement du texte (bas1_btn et haut1_btn).

10. Dans le calque Label, insérez une image clé vide dans l'image clé 2 et renseignez le champ `Image` du panneau des propriétés avec l'étiquette Charge. Insérez ensuite une autre image clé vide dans l'image 10 et indiquez cette fois l'étiquette `Affiche`.

11. Enregistrez votre fichier, publiez votre animation dans une page HTML sous le même nom que le source, soit `annuaire.htm`, et testez-la depuis un navigateur (voir figure 11-13).

> **À noter**
>
> Dans cet exemple, il n'est pas indispensable de tester l'application depuis le WebLocal puisque aucun fichier PHP n'est utilisé.

Figure 11-13
Test de l'annuaire dans un navigateur

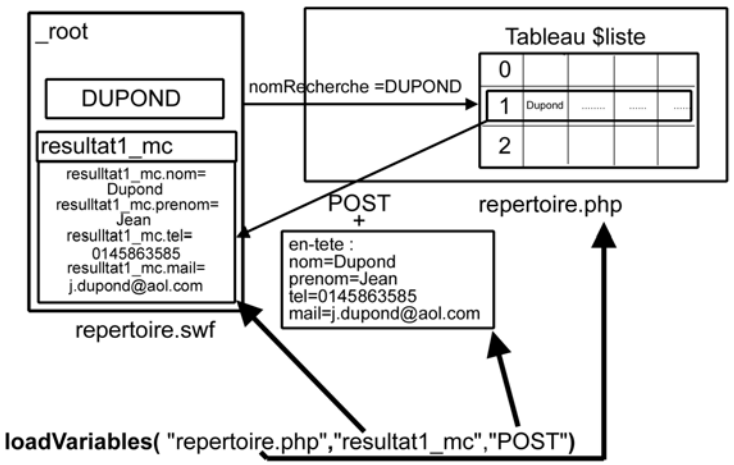


Interfaçage Flash-PHP avec loadVariables() ou loadVariablesNum()

Dans l'exemple précédent, nous avons utilisé la méthode `loadVariablesNum` sans préciser de méthode (`POST` ou `GET`). Par conséquent, seules les données du fichier ont été chargées dans l'animation.

Cependant, les méthodes `loadVariables` et `loadVariablesNum` permettent aussi de transférer des données d'une manière bidirectionnelle (de la source de données vers Flash et de Flash vers la source de données). Il suffit pour cela de définir la méthode à utiliser (`GET` ou `POST`) dans les paramètres de la fonction. Évidemment, dans ce cas, la source du chargement ne peut être un simple fichier texte car le même document devra générer dynamiquement les informations envoyées à l'application Flash et récupérer d'autres données retournées par Flash en méthode `GET` ou `POST`. Afin d'assurer cette double fonction, nous utiliserons un script PHP (voir le fichier `repertoire.php` dans l'exemple de la figure 11-14).

Figure 11-14
Utilisation de la méthode loadVariables pour échanger des données entre une animation Flash et un script PHP



Un répertoire Flash-PHP

Pour illustrer cet interfaçage Flash-PHP, nous vous proposons de créer un petit répertoire réalisé à l'aide d'une animation Flash (`repertoire.fla`) couplée à un script PHP (`repertoire.php`).

L'interface Flash est constituée d'un champ de saisie dans lequel l'utilisateur peut saisir le nom de la personne recherchée. Un bouton de validation permet d'appeler la méthode `loadVariables` afin d'envoyer en `POST` la valeur de ce champ au script PHP. En retour, le script PHP renvoie dans un clip `resultat1_mc` les coordonnées de la personne. Dans le clip, une boucle (semblable à celle utilisé dans l'exemple précédent) s'effectue tant que la dernière valeur à charger n'est pas disponible (une valeur spécifique, `fin="ok"`, sera envoyée pour indiquer la fin du chargement des données). Dès que toutes les données sont chargées dans le clip, l'animation est dirigée vers l'étiquette Affiche qui permet d'afficher les coordonnées du contact. Si aucun nom de contact ne correspond au nom recherché, le script renvoie la valeur `Inconnu`.

Le document Flash

L'interface Flash est composée d'un formulaire contenant un champ texte de saisie (nom de variable : `nomRecherche`) et d'un bouton qui permet d'appeler la méthode `loadVariables` (nom d'occurrence du bouton : bouton1_btn). Un clip, dont l'occurrence est nommée `resultat1_mc`, est dédié à la récupération des données retournées par le script PHP. Ce clip comporte une étiquette `Recherche` sur laquelle l'animation boucle tant que toutes les données ne sont pas complètement chargées. Une autre étiquette nommée `Affiche` correspond à l'affichage du tableau des coordonnées du contact. Ce tableau contient autant de champs texte dynamiques que de valeurs retournées par le script. Les noms de variable de chaque champ sont les mêmes que les noms des variables générées par le script PHP (`nom`, `prenom`, `tel`, `mail`).

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `repertoire.fla` dans un sous-répertoire du dossier `www/SITEflash/`. Vous pouvez prendre par exemple le répertoire `www/SITEflash/3-progStructuree/chap11/loadVariables/`.
2. Créez quatre calques : Fond, Résultat, Formulaire et Action.
3. Personnalisez le calque Fond en ajoutant un cadre qui contiendra les éléments de l'application.
4. Dans l'image 1 du calque Résultat, créez un clip `resultat_mc` et nommez son occurrence `resultat1_mc`. Ajoutez dans l'image clé 1 du clip `resultat1_mc` quatre champs dynamiques et nommez respectivement leur champ `Var :` `nom`, `prenom`, `tel` et `mail`.
5. Dans l'image 1 du calque Formulaire, insérez un champ texte de saisie et nommez son champ `Var : nomRecherche`.
6. Dans l'image 1 du calque Action, ajoutez une méthode de gestionnaire d'événements afin d'appeler la méthode `loadVariables` et une méthode `gotoAnPlay` (cette méthode est appliquée au clip et permet de démarrer la boucle de chargement) dès que l'utilisateur clique sur le bouton bonton1_btn (voir figure 11-15).

Code à ajouter dans l'image 1 du scénario principal :

```
bouton1_btn.onRelease= function() {
     loadVariables("repertoire.php","resultat1_mc","POST");
     resultat1_mc.gotoAndPlay("Recherche");
     }
```

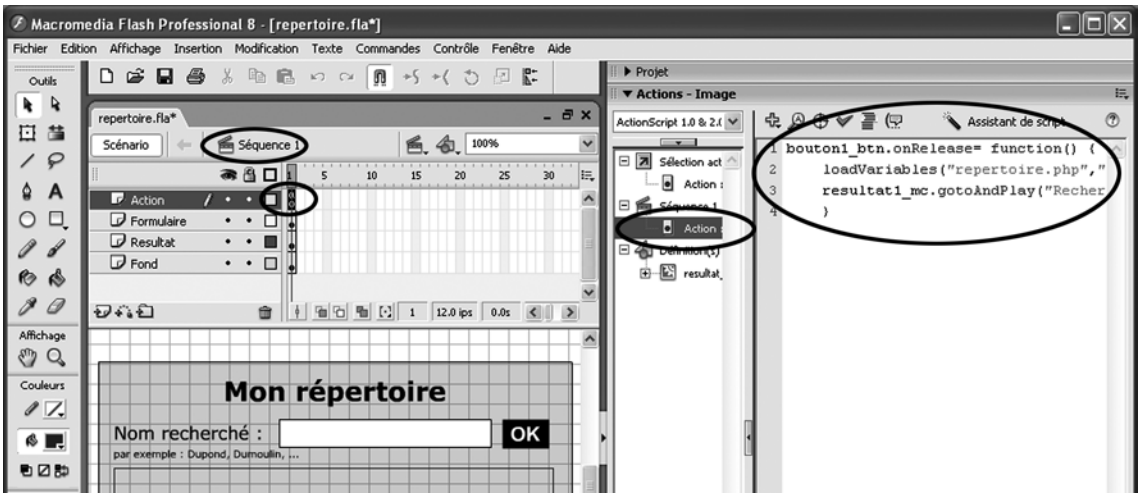


Figure 11-15

Création des quatre calques du scénario principal et intégration du code dans l'image clé 1 du scénario principal

7. Ouvrez le scénario du clip `resultat1_mc` et créez trois calques : Fond, Action et Label (voir figure 11-16).

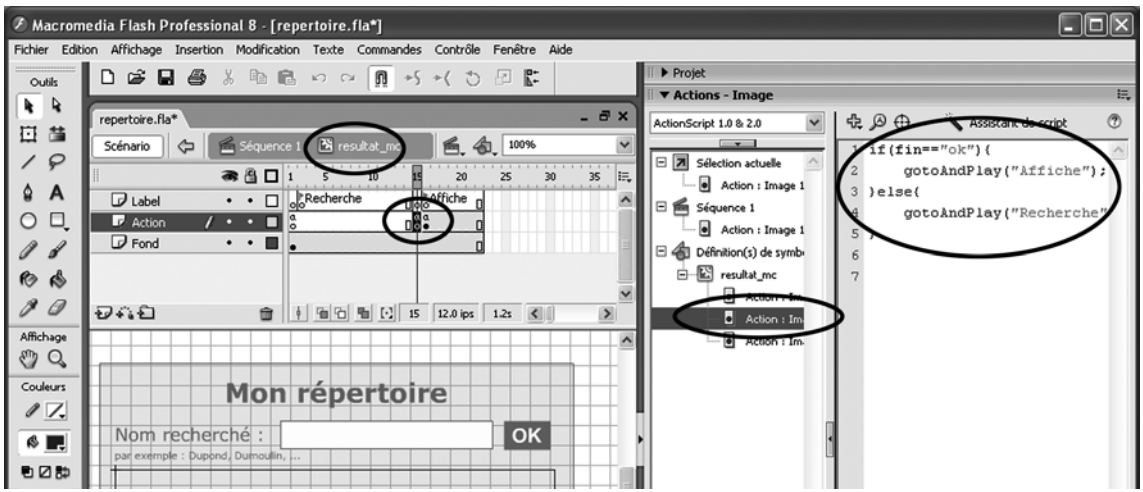


Figure 11-16

Intégration du script de test de la boucle de chargement dans l'image clé 15 du scénario du clip resultat1_mc

8. Dans le calque Fond, ajoutez sur la scène un cadre destiné à délimiter la zone de résultat et insérez une image clé dans l'image 22.

9. Dans l'image 1 du calque Action, ajoutez une instruction `stop()`.

10. Dans l'image 15 du calque Action, insérez une image clé vide et ajoutez les lignes de code suivantes afin de tester la fin du chargement (voir figure 11-17) :

```
if(fin=="ok"){
      gotoAndPlay("Affiche");
}else{
      gotoAndPlay("Recherche");
}
```

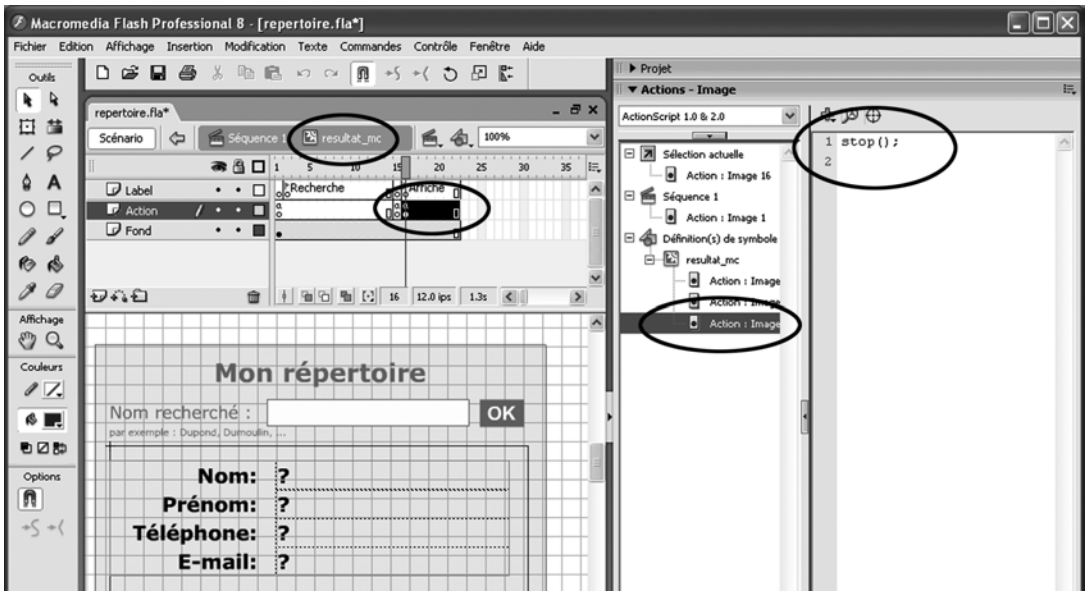


Figure 11-17

Création des quatre champs de texte dynamiques dans l'image clé 16 du scénario du clip resultat1_mc

11. Dans l'image 16 du calque Action, insérez une image clé vide et ajoutez une instruction `stop()` puis ajoutez sur la scène quatre champs de texte dynamiques respectivement nommés : `nom`, `prenom`, `tel` et `mail` (voir figure 11-18).

12. Dans le calque Label, définissez une zone nommée Recherche de l'image 2 à 14 et une deuxième zone nommée Affiche de l'image 16 à 22.

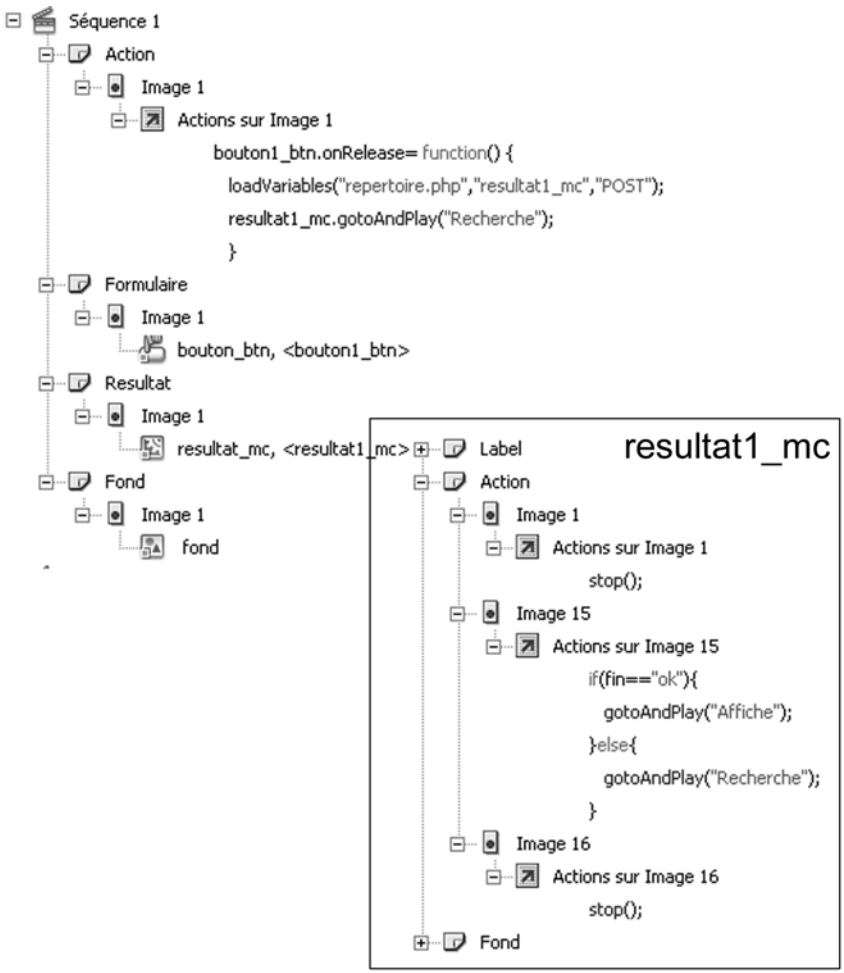


Figure 11-18
Structure de l'animation complète affichée avec l'explorateur d'animation

Le document PHP

Le fichier `repertoire.php` a une double fonction :

- recueillir le nom du contact recherché (`$nomRecherche`) puis réaliser une recherche dans un tableau de variables contenant tous les contacts du répertoire ;
- mettre en forme les différentes coordonnées du contact trouvé afin de renvoyer les données vers l'interface Flash.

La mémorisation des contacts sera réalisée à l'aide d'un tableau de variables (`$liste`) à deux dimensions. Le premier indice correspond au numéro du contact. Le second permet de distinguer les quatre types de coordonnées d'un contact (nom, prenom, tel et mail). Les valeurs du tableau sont affectées directement dans le script, mais il est facile de construire un tableau de données similaire à partir d'une source de données externe (fichier texte, XML ou base de données).

La valeur de la variable `$nomRecherche` (envoyée par le formulaire Flash) est utilisée dans la boucle de recherche afin de tester la correspondance de ce nom avec l'un des noms du tableau `$liste`. Afin d'éviter que la saisie d'un nom en majuscules ou en minuscules ne perturbe la recherche, la variable `$nomRecherche` est automatiquement transformée en majuscules avant d'être utilisée dans la boucle de recherche (instruction PHP `strtoupper($nomRecherche)`). Dès que le contact est trouvé, son numéro (premier indice du tableau `$liste`) est mémorisé dans une variable `$iDresultat`. Lors de la mise en forme des données des résultats, cette même variable est utilisée pour récupérer les quatre coordonnées issues du tableau `$liste`.

À noter

Pour la mise en forme des données, une fonction `envoi()` est utilisée afin d'éviter de répéter le script de mise en forme pour chaque couple de variable/valeur.

1. Créez un nouveau document PHP (Fichier>Nouveau>Page dynamique>PHP) et sauvegardez-le sous le nom `repertoire.php` dans le même répertoire que le document Flash.
2. Passez en mode Code et saisissez la première ligne de script ci-dessous dans la fenêtre du document (voir figure 11-19).

```
<?php
// Initialisation des variables (paramètre de recherche : nomRecherche envoyé en POST
➥par Flash)
if(isset($_POST['nomRecherche'])) $nomRecherche=$_POST['nomRecherche'];
else $nomRecherche="";
```

3. Ajoutez à la suite le script suivant, destiné à déclarer la fonction `envoi()` utilisée pour la mise en forme des données renvoyées à Flash (voir figure 11-19) :

```
// envoi() Fonction de mise en forme des données retournées vers Flash
// ENTREE : deux arguments : $var = le nom de la variable et $val
➥=  la valeur correspondante
// SORTIE : affiche le couple &variable=valeur conforme au format de chargement
➥de données Flash
function envoi($var, $val){
echo "&".$var."=".utf8_encode($val);
}
```

4. Ajoutez ensuite le script suivant, destiné à initialiser le contenu du tableau `$liste` contenant les différents contacts du répertoire :

```
//------Création d'un tableau de contacts $liste pour les tests
$contact0=array("Inconnu","--","--","--");
$contact1=array("DUPOND","Jean","0145863585","j.dupond@aol.com");
$contact2=array("VERMONT","Patrice","0145568749","p.vermont@wanadoo.fr");
$contact3=array("DUMOULIN","Maurice","0460258936","m.dumoulin@free.fr");
$liste=array($contact0,$contact1,$contact2,$contact3);
```

5. Ajoutez ensuite le script suivant, qui permet de générer la boucle de recherche et de parcourir ainsi tous les noms du tableau `$liste` (voir figure 11-19) :

```
//------Recherche des coordonnées du contact
$nomRecherche=strtoupper($nomRecherche);
```

```
// Transforme le nom recherché en Majuscule
$iDresultat=0;
// Initialisation de l'ID du contact correspondant à la recherche
for($i=1;$i<=count($liste);$i++){
if($liste[$i][0]==$nomRecherche) $iDresultat=$i;
}
```

6. Ajoutez enfin ce dernier script afin de mettre au format URL les données à renvoyer à Flash en appelant la fonction `envoi()` déclarée ci-dessus :

```
//------Mise en forme de la réponse envoyée à Flash
envoi("nom",$liste[$iDresultat][0]);
envoi("prenom",$liste[$iDresultat][1]);
envoi("tel",$liste[$iDresultat][2]);
envoi("mail",$liste[$iDresultat][3]);
envoi("fin","ok");//drapeau de fin de données
```

7. Enregistrez le document PHP.

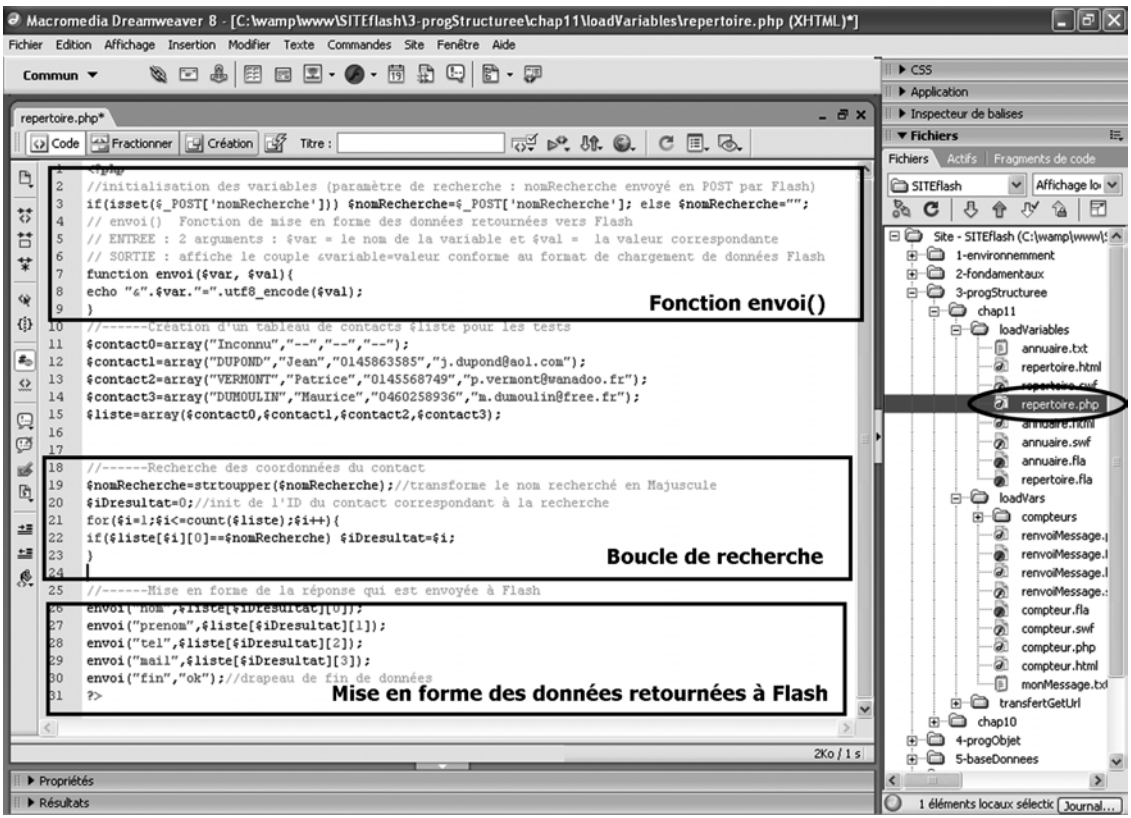


Figure 11-19

Saisie et enregistrement avec Dreamweaver des scripts du fichier repertoire.php

Test dans le WebLocal

Pour le test, il suffit d'appeler le fichier repertoire.htm depuis votre Web local (sélectionnez l'option `localhost` depuis le menu du manager de Wamp. Cliquez ensuite successivement sur les différents dossiers afin d'afficher le fichier repertoire.htm). Une fois le formulaire de recherche affiché à l'écran, saisissez un nom dans le champ (Dupond, par exemple) et cliquez sur le bouton OK. La recherche est lancée et les résultats doivent s'afficher dans l'interface (voir figure 11-20).

Figure 11-20

Test du répertoire dans le WebLocal

Interfaçage Flash-PHP avec la classe loadVars

L'utilisation de la classe `LoadVars` est une alternative à l'utilisation des méthodes `loadVariables()` et `loadVariablesNum()` pour l'échange des données entre Flash et une source de données ou une application serveur (script PHP, par exemple).

Une fois l'instance de la classe créée (objet LV créé avec `new LoadVars()`), il suffit ensuite d'utiliser ses méthodes (`load()`, `send()` et `sendAnLoad()`) pour gérer des transferts de données depuis la source vers Flash (chargement), de Flash vers une application serveur (envoi) ou d'une manière bidirectionnelle (interfaçage). La méthode `toString()` permet de coder les couples variable/valeur de l'objet LV en une chaîne au format URL (`var1=valeur1var2=valeur2…`). De même, les classes `getBytesLoaded()` et `getBytesTotal()` permettent, lors de l'utilisation des méthodes `load()` ou `sendAnLoad()`, de connaître le nombre d'octets téléchargés (dans le cas d'un chargement en cours) et le nombre total d'octets téléchargés (à la fin du chargement).

La classe `LoadVars` met aussi à votre disposition deux propriétés, `contentType` et `loaded`, qui permettent de modifier le type MIME des données avant un envoi (le type par défaut est `application/x-www-urlform-encoded`) et de savoir si un chargement est terminé (dans ce cas, la propriété `loaded` est à l'état `true`).

Enfin, deux gestionnaires d'événements, `onData` et `onLoad`, permettent de contrôler la fin du chargement avant ou après l'analyse des données par Flash Player lors de l'usage d'une méthode `load()` ou `sendAndLoad()`.

Tableau 11-10. Syntaxe du constructeur de la classe loadVars

Classe `loadVars`
La classe `loadVars` permet de créer des objets (`monObjet_lv`) et d'utiliser ensuite leurs méthodes pour envoyer ou charger des données.
Création d'un objet à l'aide du constructeur : `var monObjet_lv = new loadVars()`

Tableau 11-11. Méthodes d'envoi et de chargement de la classe loadVars

Méthodes	Définitions des paramètres
`monObjet_lv.load("url")` Charge des variables depuis l'URL spécifiée, analyse les données et place les variables résultantes dans l'objet `monObjet_lv`.	`url` : URL absolue ou relative de la source de données. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur Web, l'URL doit être du même domaine que le fichier SWF.
`monObjet_lv.send("url" [,"cible" ,"methode" ])` Envoie les variables de l'objet `monObjet_lv` à l'URL spécifiée. Toutes les variables énumérables dans `monObjet_lv` sont concaténées dans une chaîne au format `application/x-www-form-urlencoded` par défaut (ce format MIME peut être modifié au préalable à l'aide de la propriété `contentType`).	`cible` : fenêtre du navigateur dans laquelle les réponses seront affichées. Par défaut, la cible est `_self` (fenêtre courante) mais vous pouvez utiliser `_blank` (nouvelle fenêtre), `_parent` (fenêtre du cadre parent) ou `_top` (fenêtre du cadre principal).
`monObjet_lv.sendAndLoad("url", objetCible [,"methode"])` Envoie les variables de l'objet `monObjet_lv` à l'URL spécifiée et charge les variables issues de la réponse du serveur dans l'objet `objetCible`.	`methode` : permet de spécifier la méthode `HTTP` utilisée (`GET` ou `POST`). Si aucune méthode n'est précisée, la méthode `POST` sera utilisée par défaut.
	`objetCible` : objet `LoadVars` qui reçoit les valeurs chargées (Attention ! Le nom de cet objet ne doit pas être encadré par des guillemets).
	[xxx] : le code xxx est facultatif.
	(Attention ! Vous ne devez surtout pas saisir les crochets `[` et `]` dans le code.)

Tableau 11-12. Autres méthodes de la classe loadVars

Méthodes	Définitions des paramètres

`monObjet_lv.getBytesLoaded()`
Renvoie le nombre d'octets déjà téléchargé lors d'un chargement de données initié par l'appel de la `méthode load()` ou `sendAndLoad()`.
 	
`monObjet_lv. getBytesTotal()`
Renvoie le nombre total d'octets téléchargés après un chargement de données initié par l'appel de la méthode `load()` ou `sendAndLoad()`. Cette méthode renvoie `undefined` si aucune opération `load` n'est en cours ou si une opération `load` n'a pas encore été initiée.
 	
`monObjet_Rv.addRequestHeader(*`
 	
Première utilisation :
`monObjet_lv.addRequestHeader(nomDentête, valeurDentête)`
 	
Seconde utilisation :
`tabEnTete = ["nomDentête_1", "valeurDentête_1" ... "nomDentête_n", "valeurDentête_n"];`
`monObjet_lv.addRequestHeader(tabEnTete);`
 	
Ajoute ou modifie les en-têtes de requête HTTP (telles que `Content-type` ou `SOAPAction`) envoyés avec les actions `POST`.
 	
Dans la première utilisation, vous transmettez deux chaînes à la méthode : `nomDentête` et `valeurDentête`.
 	
Dans la seconde utilisation, vous transmettez un tableau de chaînes*, alternant noms et valeurs d'en-têtes.
	`nomDentête` : nom d'en-tête de requête HTTP
	
	`valeurDentête` : valeur associée à `nomDentête`
	
	`tabEnTete` : tableau de variables contenant les couples `nomDentête_n` et `valeurDentête_n`. Ce tableau est ensuite transmis en paramètre lors de l'appel de la méthode `addRequestHeader()`.
	
	(*) Attention !
	Dans la syntaxe de la méthode `addRequestHeader()`, vous devez saisir les crochets `[` et `]` dans le code car il s'agit dans ce cas de délimiteurs de tableau de variables.

Tableau 11-13. Propriétés de la classe loadVars

Descriptions des propriétés

`monObjet_lv.loaded`
La propriété `loaded` indique si une opération de chargement de données initiée par l'appel de la méthode `load()` ou `sendAndLoad()` est terminée. Elle est affectée de la valeur `true` lorsque l'opération est terminée et de la valeur false dans le cas contraire. Si une opération de chargement échoue avec une erreur, la propriété `loaded` reste définie sur la valeur `false`. À noter : Si la méthode `load()` ou `sendAndLoad()` n'a pas encore été initiée, cette propriété renvoie la valeur `undefined`.
 	
`monObjet_lv.contentType`
La propriété `contentType` est utilisée pour modifier le type `MIME` ajouté à l'en-tête HTTP lors de l'envoi de données initié par l'appel de la méthode `send()` ou `sendAndLoad()`. Par défaut, cette propriété est affectée avec la valeur `application/x-www-urlform-encoded`.

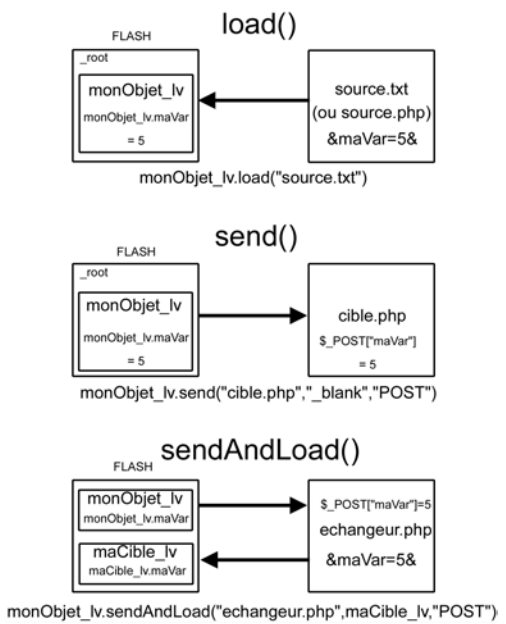
Tableau 11-14. Gestionnaires d'événements de la classe loadVars

Gestionnaires d'événements	Définitions des paramètres

`monObjet_lv.onData = function (src) {`
`// Mettre ici vos instructions }`
Le gestionnaire d'événements `onData` est invoqué si les données ont été complètement téléchargées lors d'un chargement initié par l'appel de la méthode `load()` ou `sendAndLoad()`. Ce gestionnaire étant invoqué avant l'analyse des données, il peut être utilisé pour appeler une routine d'analyse personnalisée au lieu d'une routine intégrée dans Flash Player. Par défaut, la routine appelée est `monObjet_lv.onLoad` (voir ci-dessous) mais si vous affectez une fonction personnalisée à `monObjet_lv.onData`, la routine `monObjet.onLoad` ne sera plus appelée, à moins que vous ne l'ajoutiez dans les instructions de la fonction personnalisée.
`monObjet_lv.onLoad = function (success) {`
`// Mettre ici vos instructions }`
Si aucune fonction personnalisée n'est affectée à l'objet `monObjet_lv.onData` (voir ci-dessus), le gestionnaire d'événements onLoad est invoqué par défaut dès que le chargement initié par l'appel de la méthode `load()` ou `sendAndLoad()` est terminé.
Si l'opération de chargement est réussie, l'objet `monObjet_lv` est alors renseigné par les valeurs téléchargées. Ces variables sont disponibles depuis l'objet `monObjet_lv` (ex : `monObjet_lv.maVar`) lorsque le gestionnaire `onLoad` est invoqué.
	`src` : chaîne contenant les paires nom-valeur encodées URL téléchargées depuis le serveur.
	À noter : Si une erreur se produit lors du chargement, le paramètre scr est alors affecté avec la valeur undefined.
	`success` : ce paramètre indique si l'opération de chargement s'est déroulée avec succès (`true`) ou non (`false`).
	Remarque : En pratique dans la plupart des applications, on utilise uniquement le gestionnaire d'événements `onLoad` pour gérer les variables téléchargées (comme dans l'exemple ci-dessous).
	Exemple :
	Script placé dans l'image 1 du scénario principal de l'animation `loadVars.fla` :
	`monObjet_lv= new LoadVars();`
	`monObjet_lv.load("fichier.txt");`
	`monObjet_lv.onLoad=function(success) {`
	`if(success){`
	`_root.nom= monObjet_lv.maVar1;`
	`_root.prenom= monObjet_lv.maVar2;`
	`} else {`
	`gotoAndStop("Erreur");`
	`}`
	`}`
	`----------`
	Contenu du document `fichier.txt` :
	`&maVar1=Defrance&maVar2=Jean-Marie`

Figure 11-21

Utilisation des différentes méthodes de la classe loadVars pour gérer le chargement ou l'envoi de données entre une animation Flash et une source de données ou une application serveur PHP.



Exemples de scripts utilisant la classe loadVars

Chargement dans un tableau avec load()

Ce script permet de charger dans une animation Flash sept variables (les sept jours de la semaine : `jour1=lundi`, `jour2=mardi`, etc.) depuis un fichier texte `fichierJour.txt`. Dès que le chargement est terminé, la tête de lecture de l'animation est envoyée vers une image Affiche et ces variables sont affectées à un tableau de variables nommé `monTableau_array`. Si un problème survient lors du chargement, la tête de lecture de l'animation est envoyée vers une image Erreur.

Script à ajouter dans l'image clé 1 du scénario principal :

```
stop();
// Création du tableau récepteur
monTableau_array=new Array();
// Création de l'objet LV
monObjet_lv= new LoadVars();
// Chargement du fichier
monObjet_lv.load("fichierJours.txt");
// Gestionnaire d'événement onLoad
monObjet_lv.onLoad=function(success) {
if (success) {
            for (var n = 1; n<=monObjet_lv.nbre; n++) {
                    monTableau_array.push(monObjet_lv["jour"+n]);
                    gotoAndStop("Affiche");
            }
     } else {
            gotoAndStop("Erreur");
     }
}
```

Un champ texte dynamique dont l'occurrence sera nommée `afficheTableau_txt` doit être créé sur la scène de l'image Affiche.

Script à ajouter dans l'image nommée Affiche :

```
stop();
afficheTableau_txt.text=monTableau_array;
```

Contenu du document `fichierJour.txt` utilisé dans le script précédent :

```
&jour1=Lundi&jour2=Mardi&jour3=Mercredi&jour4=Jeudi&jour5=Vendredi&jour6=Samedi&jour7
➥=Dimanche&nbre=7&
```

Envoi vers un script avec send()

Ce script permet d'envoyer des variables (propriétés ajoutées à l'objet LV : `monObjet_lv.monNom` et `monObjet_lv.monMessage`) au script PHP afficheMessage.php. Le message reçu par le script sera ensuite affiché dans une nouvelle fenêtre (option `_blank`) (voir figure 11-22).

Figure 11-22

Message affiché dans le navigateur après l'envoi des données

Script à ajouter dans l'image clé 1 du scénario principal :

```
stop();
// Création de l'objet LV
monObjet_lv= new LoadVars();
// Initialisation des variables
monObjet_lv.monNom="Jean";
monObjet_lv.monMessage="bonjour à tous";
// Envoi des variables vers le script
monObjet_lv.send("afficheMessage.php", "_blank", "GET");
```

Script PHP du fichier afficheMessage.php :

```
<?php
if(isset($_GET['monNom'])) $monNom= utf8_decode($_GET['monNom']);
      else $monNom="inconnu";
if(isset($_GET['monMessage'])) $monMessage= utf8_decode($_GET['monMessage']);
      else $monMessage="inconnu";
echo "Voici le nouveau message de <b>".$monNom."</b> : <b>".$monMessage."</b>" ;
?>
```

Remarque

Lors de nos tests de la méthode `send()`, nous avons relevé des problèmes de fonctionnement avec la méthode `POST` et Flash Player 6 et 7 sur les plates-formes PC (voir TechNote sur le site de Macromedia pour plus de détails).

Envoi et chargement avec sendAndLoad()

Ce script permet d'envoyer un nom et un message depuis un formulaire Flash (animation `renvoiMessage.fla`) vers un script PHP (`renvoiMessage.php`). Le script PHP archive les différents messages reçus dans un fichier texte (`monMessage.txt`) et renvoie à Flash un message attestant la bonne réception. Flash charge ensuite le message de retour et l'affiche dans un champ texte dynamique placé sur la scène (`retour_txt`) (voir figure 11-23).

Sur la scène de l'animation, vous devrez au préalable créer deux champs de saisie d'occurrence `nom_txt` et `message_txt`, un bouton de validation d'occurrence `bouton1_btn` et un champ dynamique d'occurrence `retour_txt`. Dans l'image clé 1 du scénario principal, créez deux objets LV (`monEnvoi_lv` et `monChargement_lv`) afin de gérer l'envoi et le chargement dans des objets différents. Le chargement et l'envoi seront déclenchés par une action sur le bouton `bouton1_btn`.

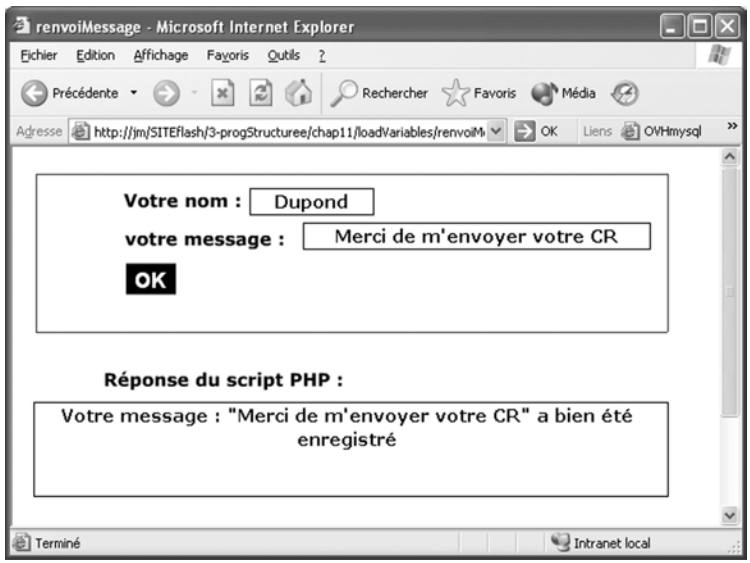


Figure 11-23

Interface de l'animation renvoiMessage.fla utilisée pour illustrer le fonctionnement de la méthode senAndLoad()

Script à ajouter dans l'image clé 1 du scénario principal :

```
stop();
// Création des objets LV
monEnvoi_lv = new LoadVars();
monChargement_lv = new LoadVars();
bouton1_btn.onRelease = function() {
      // Initialisation des variables
      monEnvoi_lv.monNom = nom_txt.text;
```

```
        monEnvoi_lv.monMessage = message_txt.text;
        // Envoi des variables vers le script
        monEnvoi_lv.sendAndLoad("renvoiMessage.php", monChargement_lv, "POST");
        // Affiche résultat chargement
        monChargement_lv.onLoad = function(success) {
                if (success) {
                        _root.retour_txt.text = monChargement_lv.retour;
                 } else {
                       _root.retour_txt.text = "problème de chargement";
                 }
        };
};
```

Script PHP du fichier `renvoiMessage.php` :

```
<?php
// Récupération des variables envoyées par Flash
if(isset($_POST['monNom'])) $monNom= utf8_decode($_POST['monNom']); else $monNom="inconnu";
if(isset($_POST['monMessage'])) $monMessage= utf8_decode($_POST['monMessage']);
➥else $monMessage="inconnu";
// Préparation du message de retour
$messageRetour="Votre message : \"".$monMessage."\" a bien été enregistré ";
$messageRetour=utf8_encode($messageRetour);
echo "&retour=".$messageRetour."&" ;
// Préparation du texte à sauvegarder dans le fichier
$monTexte="---Emetteur : ".$monNom." --- Message : ".$monMessage." \n";
$fichier=fopen("monMessage.txt","a");
fputs($fichier,$monTexte);
fclose($fichier);
?>
Contenu du fichier monMessage.txt après quatre envois de message :
---Émetteur : Dupond --- Message : Merci de m'envoyer votre CR
---Émetteur : Hamond --- Message : Mon CR a déjà été envoyé hier
---Émetteur : Dupond --- Message : Désolé mais je n'ai rien reçu !
---Émetteur : Hamond --- Message : Ok, je vous envoie une copie
```

Affichage du taux de chargement

Ce script montre comment utiliser les méthodes `getBytesLoaded()` et `getBytesTotal()` pour afficher l'état du chargement.

Script à ajouter dans l'image clé 1 de l'animation :

```
monObjet_lv = new LoadVars();
monObjet_lv.load("grosFichier.txt");
_root.onEnterFrame = function() {
      var chargePourcent:Number;
      if (monObjet_lv.loaded == false) {
```

```
        chargePourcent=Math.ceil((monObjet_lv.getBytesLoaded()
        ➥/monObjet_lv.getBytesTotal())*100);
                chargePourcent=(isNaN(chargePourcent))?1:chargePourcent;
                trace("Téléchargement effectué : "+monObjet_lv.getBytesLoaded()+" octets");
                trace("Taux de chargement :"+chargePourcent+"%");
        } else {
                trace("Le chargement est terminé");
               trace("Total téléchargé : "+Math.floor(monObjet_lv.getBytesTotal()/1024)+"Ko");
                delete this.onEnterFrame;
        }
};
```

Un compteur Flash-PHP-Txt

Pour illustrer cet interfaçage Flash-PHP-Txt, nous vous proposons de créer un système de comptage des pages visitées dans une animation Flash. Ce système sera réalisé à l'aide d'une animation Flash (`compteur.fla`) couplée à un script PHP (`compteur.php`). Différents fichiers texte (autant que de pages d'animation à gérer) seront regroupés dans un répertoire spécifique nommé `compteurs/`.

L'interface Flash est constituée d'un ensemble de trois boutons permettant d'accéder à des étiquettes différentes (`page1`, `page2` et `page3`). Un champ de texte dynamique placé au centre de la scène permet d'afficher l'état du compteur de la page active. L'image 1 du scénario principal regroupe les gestionnaires des boutons, le script de création de l'objet `LoadVars` et son gestionnaire `onLoad` ainsi qu'une fonction `compte(n)`. Lorsque l'on clique sur l'un des trois boutons, la tête de lecture se positionne sur la page désirée. Chaque page contient un appel à la fonction `compte()` qui est configurée pour passer en paramètre le numéro de la page visualisée (exemple : `compte(2)` pour la page 2). Cette fonction envoie un ordre de chargement au script `compteur.php` en lui passant le numéro de la page en paramètre d'URL. Dès réception de cette valeur, le script PHP lit le fichier texte correspondant, incrémente l'ancienne valeur et renvoie le nouvel état du compteur au fichier Flash pour affichage. Cet état sera ensuite mémorisé dans le fichier texte afin de comptabiliser les visites de la page concernée.

Le document Flash

Le document Flash est organisé en cinq calques (Label, Action, Menu, Message et Fond). Le scénario est découpé en quatre zones. La première correspond à l'image 1 et regroupe les différents scripts de l'animation ; les trois autres portent les étiquettes page1, page2 et page3 et simuleront les pages de l'animation pour lesquelles le nombre de visites devra être comptabilisé.

1. Créez un nouveau document Flash et sauvegardez-le sous le nom `compteur.fla` dans un sous-répertoire du dossier `www/SITEflash/`. Vous pouvez prendre par exemple le répertoire `www/SITEflash/3-progStructuree/chap11/loadVars/`.
2. Créez cinq calques : Label, Action, Menu, Message et Fond.
3. Personnalisez le calque Fond en ajoutant un cadre qui contiendra les éléments de l'application puis insérez une image clé dans l'image 31 du scénario.

4. Dans l'image 1 du calque Message, créez un champ de texte dynamique et nommez son occurrence `visite_txt` puis insérez une image clé dans l'image 31 du scénario.

5. Dans l'image 1 du calque Menu, insérez trois boutons et nommez leurs occurrences respectives : `page1_btn, page2_btn` et `page3_btn` puis insérez une image clé dans l'image 31 du scénario.

6. Dans l'image 1 du calque Action, ajoutez trois méthodes de gestionnaire d'événements `onRelease` pour les boutons des pages afin d'envoyer la tête de lecture sur l'étiquette correspondante.

7. Dans la même image clé, ajoutez le constructeur de la classe `loadVars` et son gestionnaire d'événements `onLoad`. Définissez à la suite la fonction `compte(n)` afin qu'elle appelle la méthode `load()` tout en passant le numéro de la page (`n`) en paramètre d'URL à la suite du nom du script (voir figure 11-24).

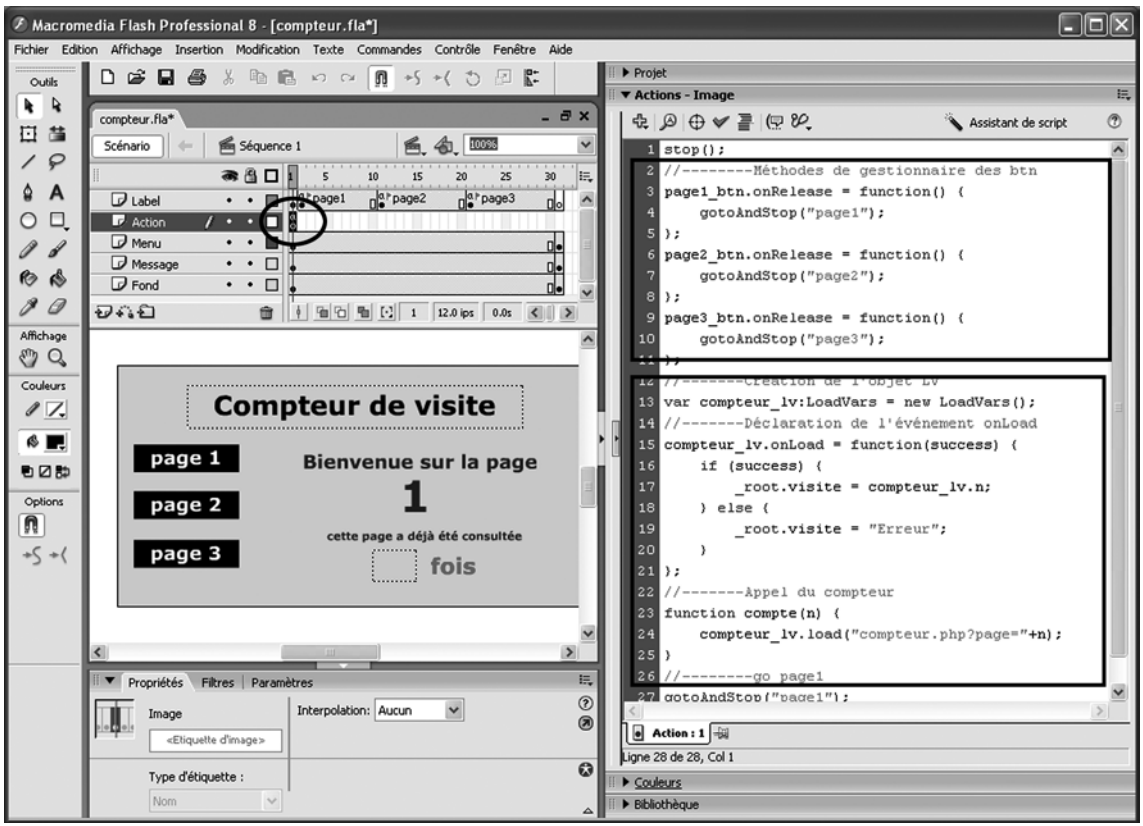


Figure 11-24

Création des cinq calques du scénario principal et intégration du code dans l'image clé 1 de ce même scénario

8. Ajoutez en bas de cette image un `gotoAndStop("page1")` afin que l'animation démarre automatiquement sur la première page. Enregistrez puis publiez l'animation.