25

Solution alternative AMFPHP

Il existe actuellement plusieurs solutions pour développer rapidement des applications communicantes entre Flash et le serveur (Flex, FlashCom, AMFPHP…). Cependant, dans le cadre de cet ouvrage, nous avons choisi de ne vous présenter que AMFPHP, qui a l'énorme avantage d'être gratuit, et qui est donc accessible à tous. L'objectif de ce chapitre est de vous initier à l'usage de cette technologie et non d'exploiter toute la puissance d'AMFPHP car il faudrait y consacrer un ouvrage complet pour faire le tour de toutes les applications possibles. Si cette introduction vous séduit et que vous désirez l'utiliser dans vos futurs projets, nous vous invitons à visiter le site officiel d'AMFPHP à l'adresse suivante : `www.amfphp.org`.

Présentation de Flash Remoting et AMFPHP

Flash Remoting pour PHP

Flash Remoting est un ensemble de composants Flash qui permet de créer facilement une passerelle de communication entre le client Flash et des ressources serveur (Programme serveur, service Web, base de données…).

Pour échanger des données avec le serveur, Flash utilise un format binaire spécifique nommé AMF (Action Message Format) qui permet d'augmenter les performances des transferts. Macromedia a développé plusieurs versions de passerelles pour communiquer avec des technologies serveur telles que ColdFusion, Java ou encore .NET. Par la suite, des versions destinées à échanger avec d'autres technologies serveur comme Perl (FLAP) ou PHP (AMFPHP) ont été développées en Open Source.

La classe AMFPHP

Comme nous venons de le voir, AMFPHP (aussi appelé « Flash Remoting pour PHP » ou encore « PHP Remoting ») est un développement Open Source (et donc gratuit) destiné à exploiter Flash Remoting avec la technologie serveur PHP.

Que le format AMF soit un format binaire permet au player Flash d'exécuter les sérialisations de données beaucoup plus rapidement que s'il s'agissait d'une chaîne de caractères classique. Avec cette technique, il n'est donc pas nécessaire de transformer les données au format XML, comme nous l'avons vu dans les chapitres précédents, car cela ralentirait considérablement le transfert entre le client et le serveur. Les composants Flash Remoting font appel à des méthodes distantes d'une classe PHP spécifique du serveur Web (AMFPHP). Ainsi couplée, la conversion d'un objet Flash sera réalisée automatiquement et permettra d'obtenir rapidement un objet de même type compatible avec la technologie PHP. Par exemple, un tableau de variables Flash sera sérialisé automatiquement côté serveur en un tableau de variables PHP en rapport. Il en est de même pour les autres types de données ActionScript (`Array`, `booleans`, `null`, `String`, `Object` et `RecordSet`), à l'exception des types `XML` et `Date` pour lesquels la conversion ne pourra pas être réalisée automatiquement (voir tableau 25-1). Il est important de noter que l'application Flash n'attend pas le résultat mais le traite dès sa réception. En effet, elle adresse les demandes à partir d'ActionScript vers le serveur et reçoit les résultats de façon asynchrone.

Tableau 25-1. Types de variables gérés par AMFPHP

Flash	Php	Commentaires	Automatique
`null`	`null`		`oui`
`boolean`	`boolean`		`oui`
`String`	`string`		`oui`
`Date`	`float`	Conversion manuelle possible par le biais de l'Unix timestamp. Dans ce cas, il faut spécifier l'attribut `"returns"` dans les attributs de la méthode `methodTable`.	`non`
`Array`	`array`		`oui`
`Object`	`associative array`		`oui`
`XML`	`string`	Conversion manuelle possible. Dans ce cas, il faut spécifier l'attribut `"returns"` dans les attributs de la méthode `methodTable`.	`non`
`Recordset`	`Resource`	Uniquement de PHP vers Flash	`oui`

Installation de Flash Remoting et d'AMFPHP

L'utilisation de cette technique de communication nécessite l'installation des composants Flash Remoting dans votre logiciel auteur Flash et celle des classes AMFPHP sur votre serveur local (ou, par la suite, sur votre serveur distant de production). Nous vous proposons de détailler ci-dessous ces deux procédures.

Installation de Flash Remoting

Pour installer les composants Flash Remoting, vous devez vous rendre sur le site d'Adobe dédié aux composants Flash. Pour trouver facilement l'adresse de cette page, saisissez les mots-clés `Flash Remoting Composant` dans votre moteur préféré, ou bien utilisez directement l'adresse suivante (sous réserve que la localisation de cette page n'ait pas changé entre-temps) : `http://www.adobe.com/fr/products/flashremoting/downloads/components/`

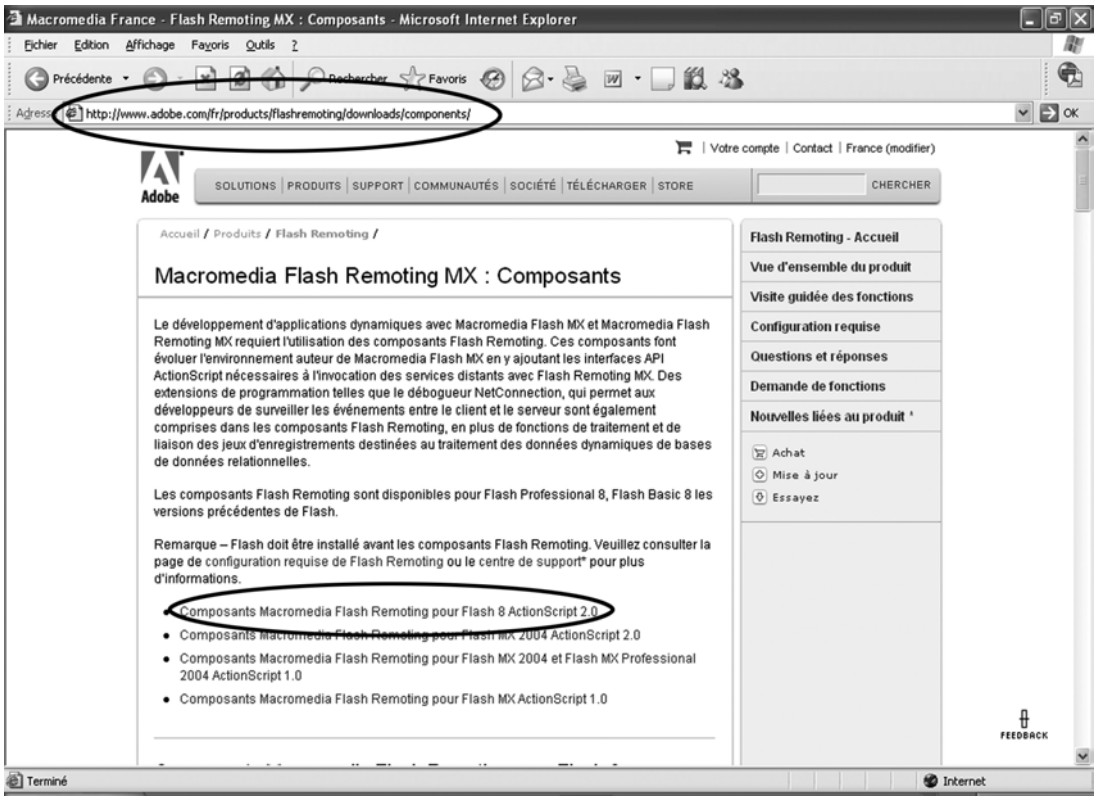


Figure 25-1

Page de téléchargement des composants Flash Remoting.

Cliquez sur le lien correspondant à la version de votre logiciel auteur (soit Flash 8 dans notre cas, voir figure 25-1). Dans la nouvelle page, choisissez ensuite la langue et le type d'OS de votre ordinateur et cliquez sur le lien de téléchargement des composants. Une fois le fichier d'installation téléchargé sur votre ordinateur, cliquez dessus pour lancer l'installation sur votre ordinateur (voir figure 25-2).

Ouvrez ensuite le logiciel auteur Flash, puis un nouveau document Flash (`Ctrl+N`). Pour vous assurer que les nouvelles classes liées aux composants Flash Remoting sont disponibles dans l'éditeur de script du panneau `Action`, cliquez sur le bouton + en haut du panneau (voir figure 25-3), puis sélectionnez l'option `Remoting` en bas de la liste déroulante. Les différentes classes Remoting disponibles doivent alors apparaître dans la seconde liste.

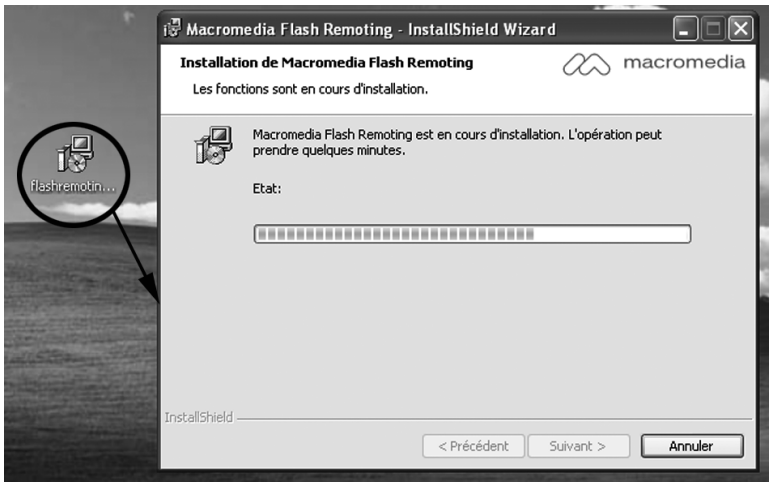


Figure 25-2
Installation des composants Flash Remoting.

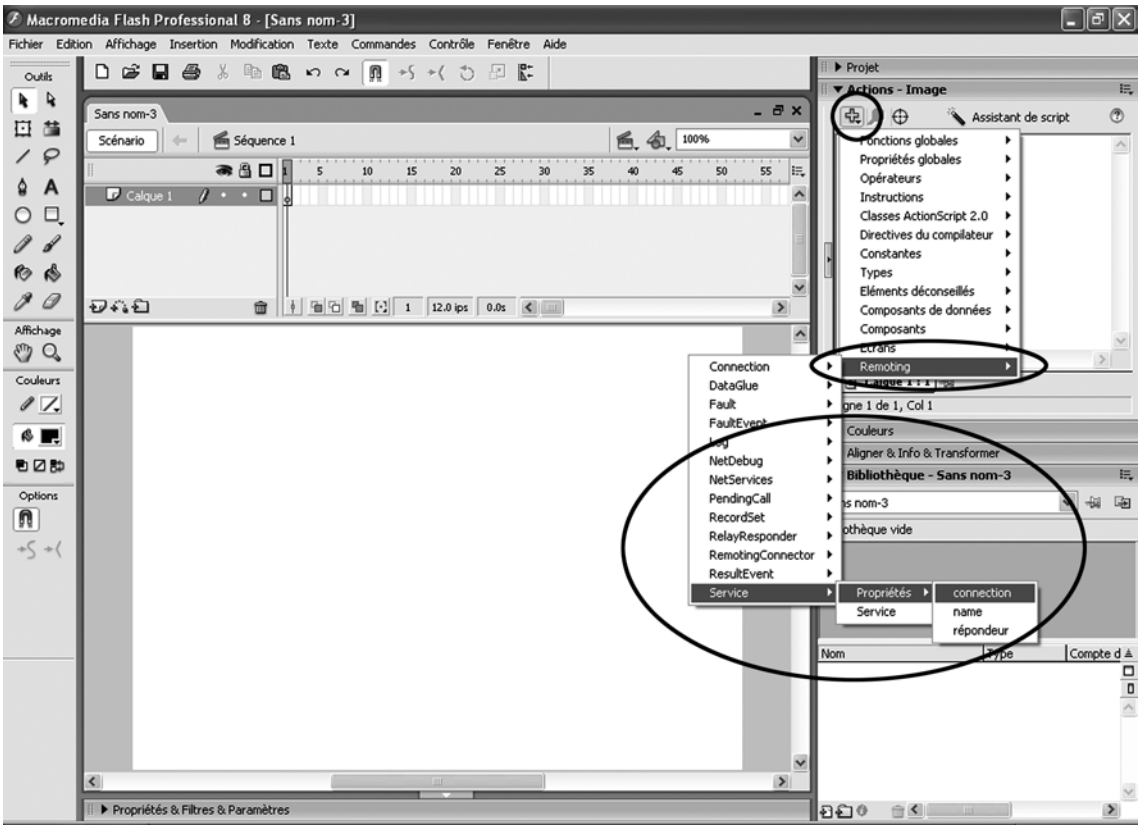


Figure 25-3
Différentes classes disponibles de Flash Remoting.

Les classes ActionScript pour Flash Remoting permettent de configurer Flash Remoting, d'interagir avec les services distants et de manipuler les données sur le client. Le tableau 25-2 présente les principales classes Flash Remoting que vous pourrez utiliser dans vos futures applications.

Tableau 25-2. Principales classes ActionScript pour Flash Remoting

Classes	Description
`Connection`	Permet de créer et d'utiliser des connexions à des services. Par exemple, la méthode `Connection.setCredentials()` fournit les justificatifs d'identité à présenter au serveur de la passerelle.
`DataGlue`	Permet de lier des objets RecordSet à des composants Flash, tels que ListBox ou ComboBox, disposant d'étiquettes ayant des données associées. Par exemple, les méthodes `DataGlue.bindFormatStrings()` et `DataGlue.bindFormatFunction()` désignent l'objet RecordSet à utiliser pour formater le composant d'interface et pour indiquer à Flash comment formater les étiquettes et les données du composant à partir de l'objet RecordSet.
`FaultEvent`	Cet objet est renvoyé en tant qu'argument de la méthode de gestion des erreurs que vous spécifiez dans l'objet Responder. Il contient les informations d'erreur renvoyées lorsqu'un appel à une fonction de service n'aboutit pas.
`NetDebug`	Obligatoire pour le Débogueur NetConnection. Gère la connexion locale entre l'application Flash à déboguer et le débogueur NetConnection.
`PendingCall`	Générée sur chaque appel à une méthode d'un objet Service. Contient la propriété `responder`, qui obtient ou définit un objet Responder pour un objet PendingCall.
`RecordSet`	Accède aux objets RecordSet renvoyés par un service et les manipule. Crée également des jeux d'enregistrements côté client. Les objets RecordSet représentent en principe les résultats de requêtes SQL et correspondent aux objets de requête.
`RelayResponder`	Objet Responder qui transmet les résultats et les erreurs aux fonctions correspondantes sur l'objet spécifié.
`ResultEvent`	Cet objet est renvoyé en tant qu'argument de la méthode de gestion des résultats que vous spécifiez dans l'objet Responder. Il contient le résultat renvoyé par une fonction de service.
`Service`	Représente une référence à un service client spécifique et aux méthodes que ce service présente. Les services distants consistent en des modules de serveur d'applications exploitant des technologies serveur telles que PHP, par exemple.

Installation d'AMFPHP

Pour installer les classes AMFPHP côté serveur, commencez par les télécharger depuis le site `www.amfphp.org`. Depuis la page d'accueil de ce site, cliquez sur le lien `Download` pour accéder à l'espace de téléchargement de SourceForge (voir figure 25-4). Cliquez sur le bouton de téléchargement de la dernière version stable de la suite AMFPHP afin de rapatrier le fichier compressé des classes sur votre ordinateur (dans nos exemples, nous avons utilisé la version 1.2.5).

Afin de tester cette technique de communication, nous allons créer au préalable un dossier SITEamfphp dans le répertoire `\www\` du serveur Wamp (soit `C:\wamp\www\SITEamfphp\`). À l'intérieur de ce nouveau répertoire, créez un autre dossier nommé `flashservices` comme le suggère la documentation du site officiel.

Une fois le fichier AMFPHP disponible sur votre ordinateur, décompressez-le, puis copiez tout le contenu du répertoire dans le dossier `SITEamfphp\flashservices\` que vous venez de créer (voir figure 25-5).

Figure 25-4

Site officiel de AMFPHP.

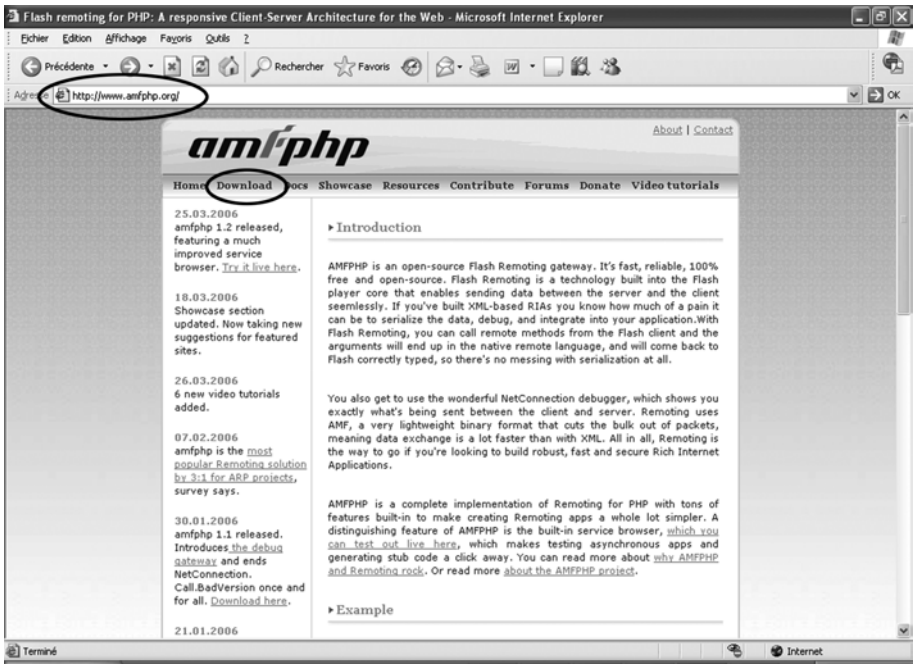
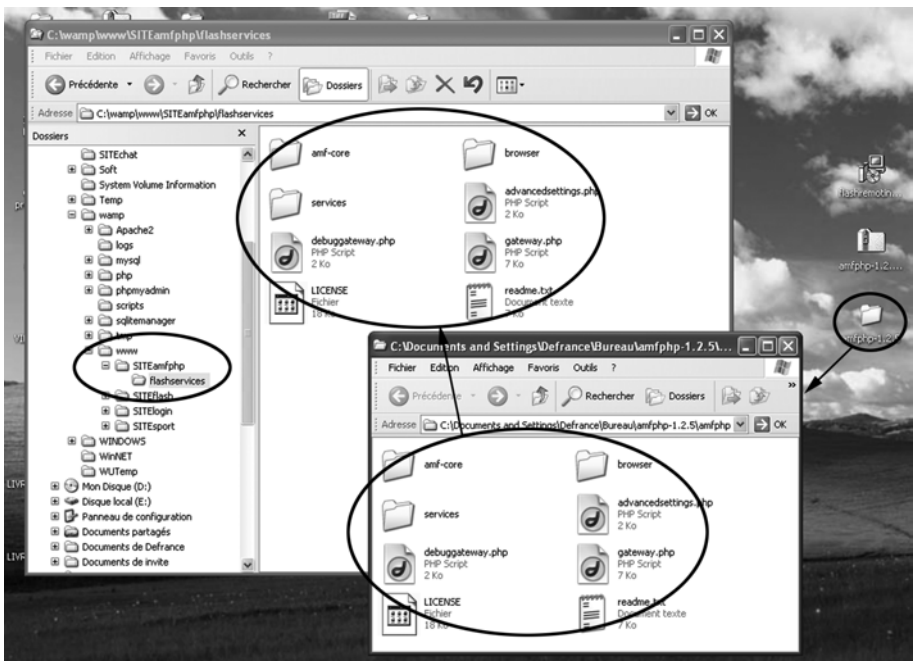


Figure 25-5

Copier le contenu de la suite AMFPHP dans un nouveau répertoire SITEamfphp/ flashservices/ situé dans le dossier / wamp/www/ .



Démarrez Dreamweaver et créez un nouveau site nommé SITEamfphp dont le dossier racine local sera configuré avec le répertoire `C:\wamp\www\SITEamfphp\` nouvellement créé (voir figure 25-6).

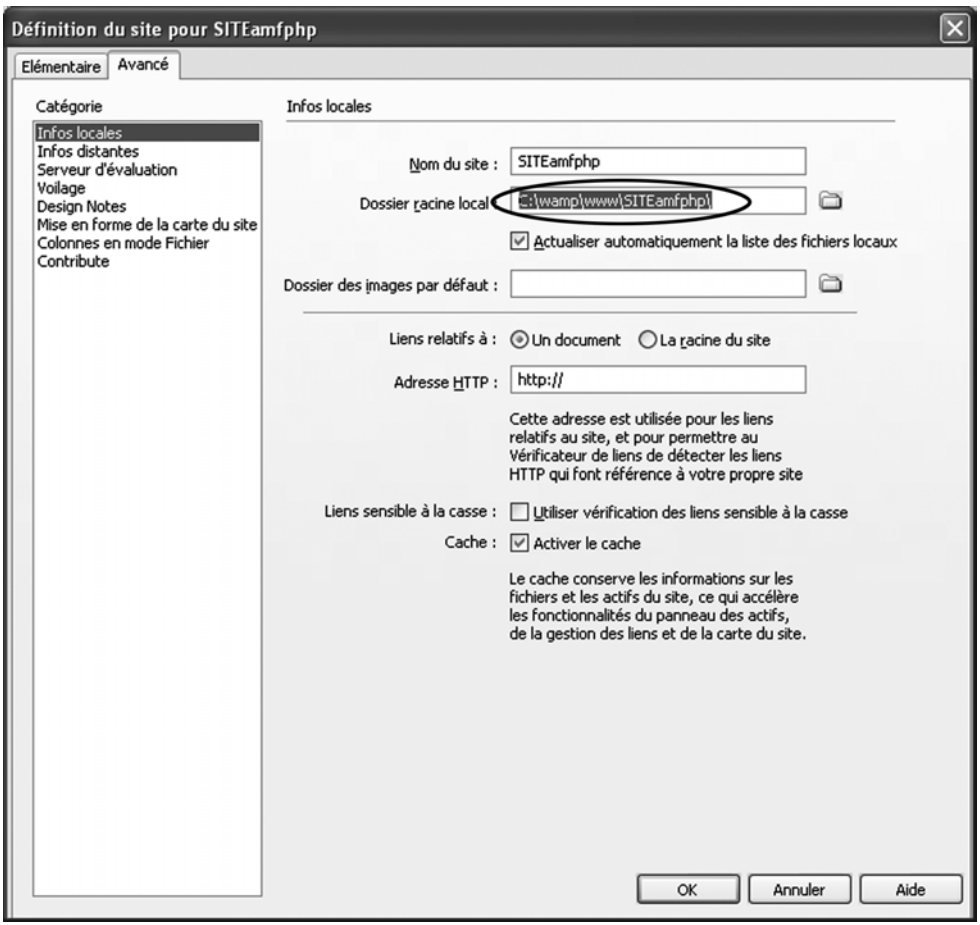


Figure 25-6

Configuration d'un nouveau site SITEamfphp dans Dreamweaver.

Ouvrez ensuite le fichier passerelle `gateway.php` et lancez une recherche dans le code de cette page avec le mot-clé `setCharsetHandler`. Supprimez le point-virgule placé devant l'instruction concernée afin de la « décommenter » (attention, il s'agit de l'instruction située à la ligne 121 et non des différentes instructions semblables placées dans les commentaires en début du code). Enregistrez ensuite ce fichier pour mémoriser votre configuration. Le fait d'avoir validé cette instruction permettra d'activer l'encodage automatique des caractères spéciaux français (ISO-8859-1) et de bénéficier ainsi du support pour la langue française.

```
$gateway->setCharsetHandler("utf8_decode", "ISO-8859-1", "ISO-8859-1");
```

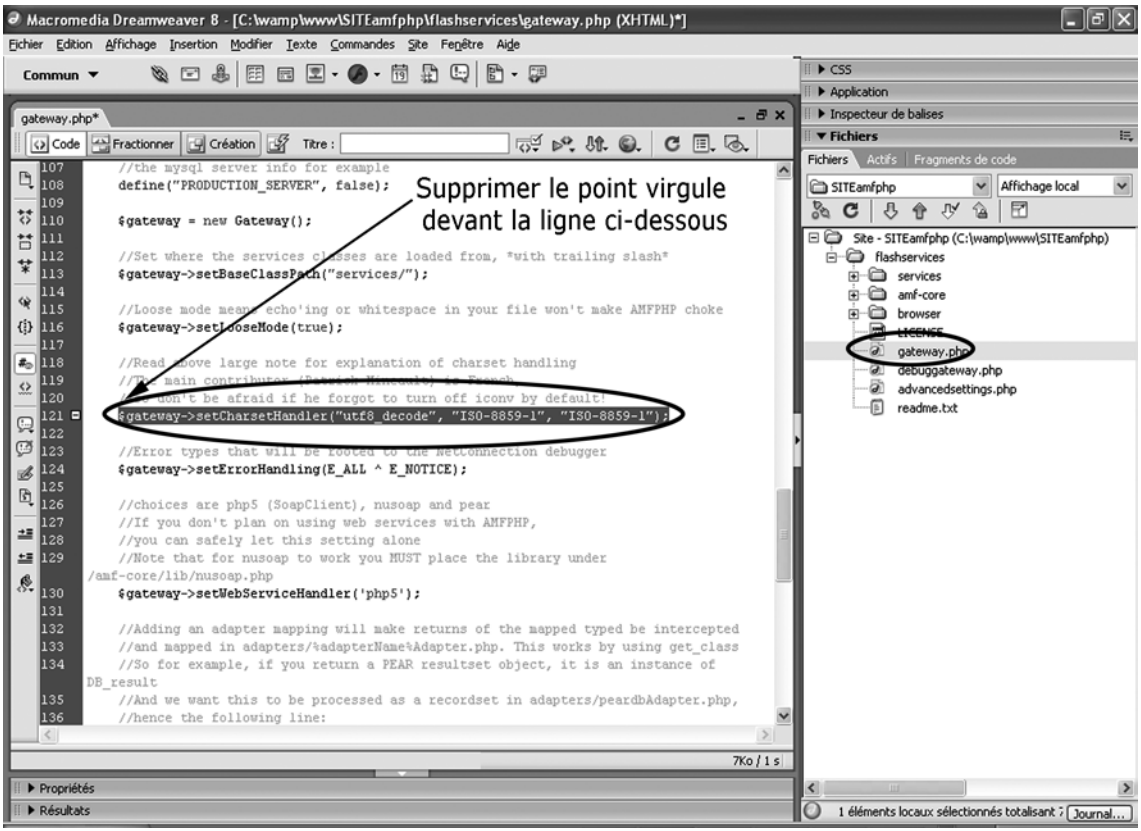


Figure 25-7

Activer le support de la langue française en décommentant la ligne $gateway->setCharsetHandler dans le fichier gateway.php.

Désactivez le débogage si vous passez en production

Lorsque votre système sera au point sur votre serveur local, pensez à modifier la valeur de la constante `PRODUCTION_SERVER` en la passant à la valeur `true` de sorte que le système de débogage ne soit pas activé en ligne :

```
define("PRODUCTION_SERVER", true);
```

Connexion de type String avec AMFPHP

Pour illustrer une première utilisation très simple d'AMFPHP, nous vous proposons de réaliser une application Flash nommée `maFacture.fla` qui utilisera une classe service PHP nommée `calculFinancier`. Dans notre exemple, l'application Flash utilisera une méthode de cette classe `calculFinancier` nommée `caculTva` qui retournera le montant de la TVA correspondant à la valeur envoyée de Flash à PHP. Évidemment, ce genre d'application ne nécessite pas l'usage d'AMFPHP car elle peut être très rapidement effectuée par une simple fonction ActionScript locale, mais la simplicité de cette application vous permettra de mieux comprendre le fonctionnement d'un transfert de données réalisé à l'aide d'AMFPHP, et par la suite de l'adapter facilement à des projets plus complexes.

Création d'une classe service AMFPHP

Commençons par créer la classe `calculFinancier` en PHP. Pour cela, ouvrez un nouveau fichier dans Dreamweaver et enregistrez-le dans le répertoire `/services/` (situé dans le dossier `/flashservices/`) en prenant soin d'utiliser le même nom que celui de la classe (donc `calculFinancier.php`).

Comme nous l'avons déjà vu au chapitre 13, la déclaration d'une classe commence par le mot-clé `class` suivi du nom de la classe (voir figure 25-8). La première méthode consiste à déclarer ce que doit être le constructeur de la classe soit, en PHP 5, `__construct()` (notez que si vous désirez conserver la compatibilté avec PHP 4, vous pouvez aussi utiliser `calculFinancier()` au lieu de `__construct()`). Dans le constructeur, nous placerons un objet particulier appelé `methodTable`. Cet objet `Array` contiendra les différentes méthodes de la classe et leurs propriétés respectives (voir figure 25-8). Dans notre exemple, nous n'aurons qu'une seule méthode nommée `calculTva` et ses propriétés. Parmi ces propriétés, nous aurons notamment la propriété `description`, qui permettra de préciser l'usage de la méthode, la propriété `access`, configurée avec la valeur `remote` afin d'autoriser Flash à accéder à la méthode (dans le cas contraire, la valeur serait `private`), la propriété `arguments`, qui précisera le nom de l'argument utilisé par la méthode (dans un `array()` afin de pouvoir déclarer plusieurs arguments si besoin), et enfin la propriété `returns` qui indiquera le type de données retournées par la méthode.

```
class calculFinancier {
  function __construct() {
         //Définition de methodTable
         $this->methodTable = array(
                 "methodTva" => array(
                         "description" => "Calcul de la TVA",
                         "access" => "remote",
                         "arguments" => array ("valeurHt"),
                         "returns" => "String"
                         )
                 );//Fin de methodTable
  }//Fin du constructeur
```

En dessous du constructeur, nous devons maintenant déclarer la méthode `calculTva`. Le fonctionnement de celle-ci sera très simple : elle recevra en argument la valeur HT (variable `$valeurHt`), puis une simple ligne de code permettra de calculer la TVA en rapport (application d'une TVA à 19,6%).

Enfin, le mot-clé `return` introduira la réponse retournée à l'application Flash, soit une phrase concaténée avec le montant de la TVA précédemment calculée.

```
//Définition de la méthode "methodTva"
  function methodTva ($valeurHt) {
         $valeurTva=$valeurHt * 0.196;
         return " Voici le montant de la TVA = ".$valeurTva;
         }
  }
```

Enregistrez ensuite votre fichier et appelez le depuis le Web Local pour vous assurer qu'aucune erreur n'est retournée par l'interpréteur PHP. L'appel de la classe dans le Web Local est uniquement effectué pour s'assurer que celle-ci ne contient pas d'erreur (auquel cas elle doit afficher une simple page blanche sans message d'erreur).

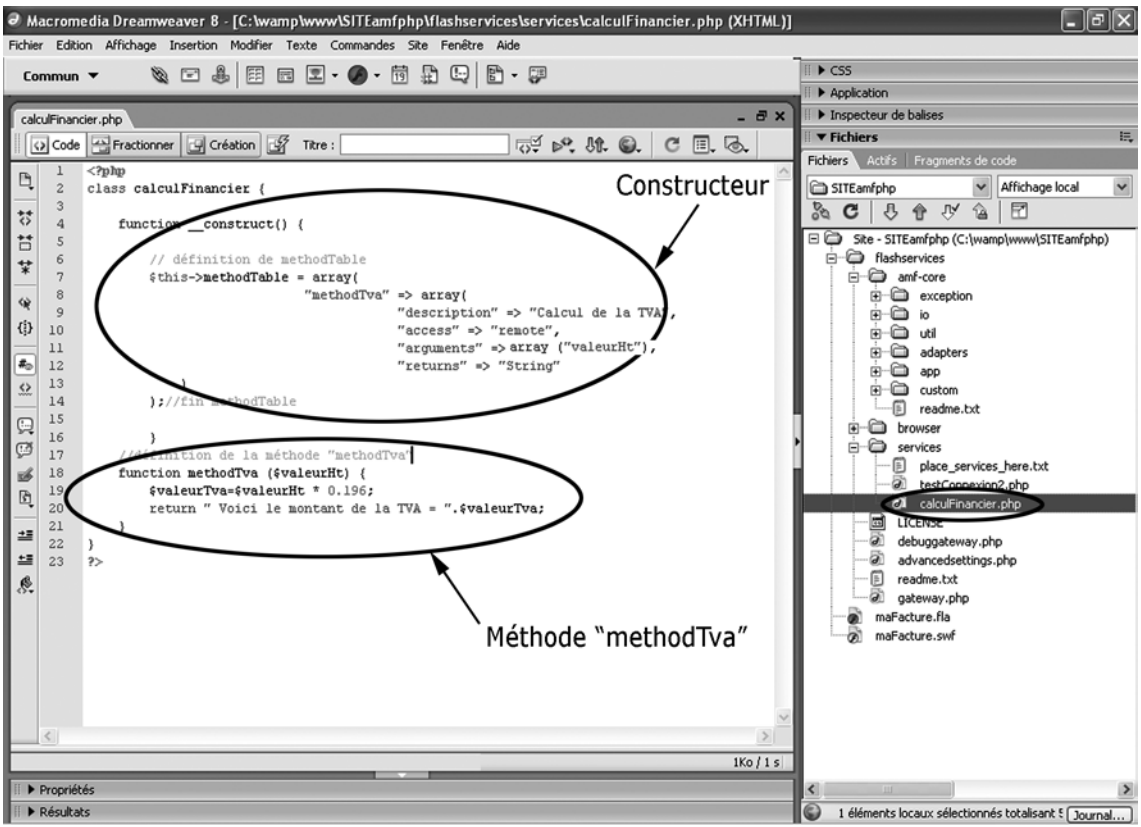


Figure 25-8

Création de la classe service PHP calculFinancier.

Création d'une application Flash Remoting

Il faut maintenant créer l'application Flash Remoting. Pour cela, ouvrez un nouveau document Flash puis enregistrez-le sous le nom `maFacture.fla` dans le répertoire `/SITEamfphp/`. Depuis le menu de Flash, ouvrez la bibliothèque commune de Remoting (`Fenêtre>Bibliothèques communes>Remoting` : voir figure 25-9). Ouvrez aussi la bibliothèque du document actif (`Ctrl+L`), puis sélectionnez les composants dans la bibliothèque Remoting et faites-les glisser dans la bibliothèque du document actif (figure 25-9).

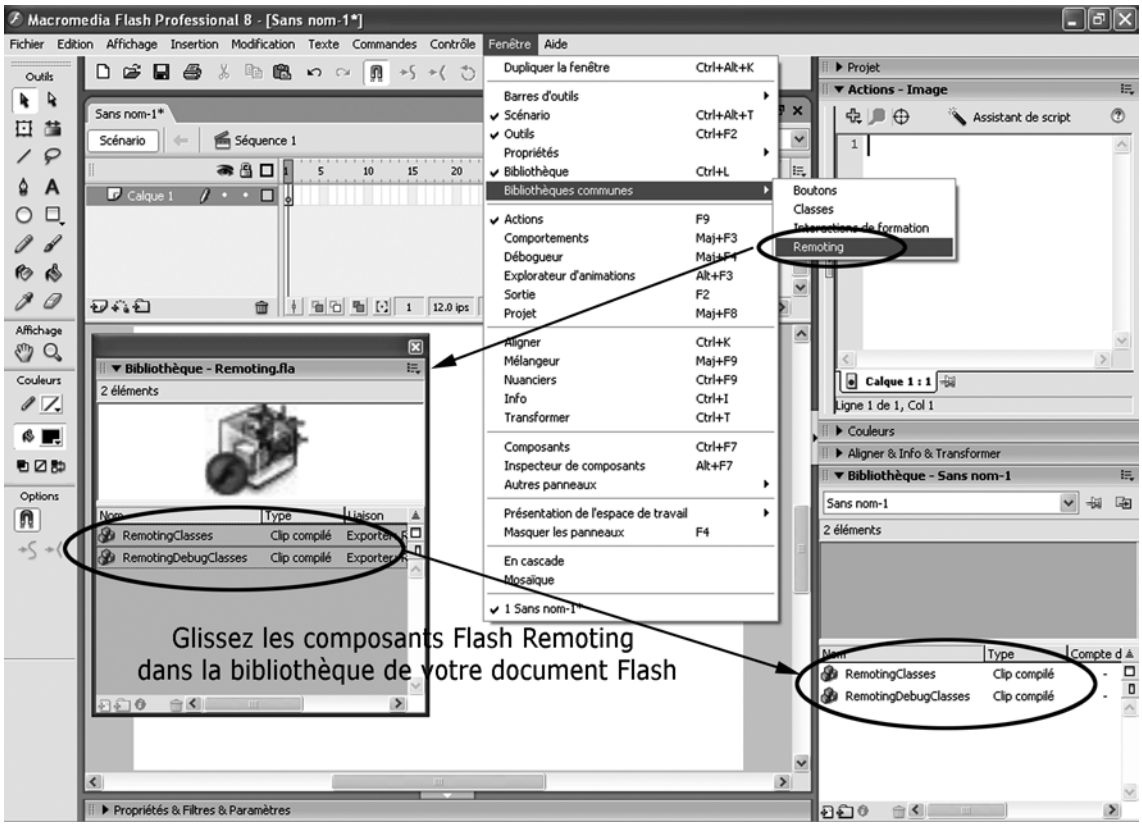


Figure 25-9

Chargement des composants Remoting dans la bibliothèque du document actif.

Placez-vous sur la première image du scénario principal puis ouvrez l'éditeur de script pour y copier les instructions d'importation des différentes classes nécessaires au fonctionnement de Flash Remoting (voir code ci-dessous ; pour plus de précision sur ces classes, reportez-vous au tableau 25-2).

```
// Import des Classes Remoting
import mx.remoting.Service; //Importer la classe Service
import mx.remoting.PendingCall //Importer la classe PendingCall
```

```
import mx.rpc.RelayResponder //Importer la classe RelayResponder
import mx.rpc.ResultEvent //Importer la classe ResultEvent
import mx.rpc.FaultEvent; //Importer la classe FaultEvent
```

Afin de disposer des fonctionnalités du débogeur NetDebug, nous allons importer aussi la classe en rapport et l'initialiser pour qu'il soit actif (voir code ci-dessous).

```
//Import de la classe NetDebug et initialisation du debogeur
import mx.remoting.debug.NetDebug;
mx.remoting.debug.NetDebug.initialize();
```

Pour créer la connexion avec le service AMFPHP, nous allons créer un objet `Service` nommé `maConnexion`. Le premier paramètre correspond à la localisation du fichier passerelle `gateway.php`, le second paramètre n'est pas utilisé dans notre exemple et nous le remplacerons donc par la valeur `null`, enfin, le troisième paramètre correspond au nom de la classe service PHP ciblée, que nous avons créée dans la partie précédente (`calculFinancier`).

```
// ----------------------Création de la connexion avec la classe PHP calculFinancier
urlPasserelle="http://localhost/SITEamfphp/flashservices/gateway.php";
var maConnexion:Service=new Service(urlPasserelle,null,"calculFinancier");_
```

La ligne de code qui suit permet d'appeler la méthode `calculTva` de la classe `calculFinancier` en lui passant comme paramètre la valeur HT de la facture (soit, dans l'exemple, la valeur numérique 100). Notez que le résultat sera retourné dans un objet `PendingCall` nommé `monRetour`.

```
//-------Appel de la méthode PHP methodTva avec passage du parametre
var monRetour:PendingCall=maConnexion.methodTva(100);
```

Il nous faut maintenant définir les deux gestionnaires dans lesquels nous allons définir les traitements à effectuer lors de la réception de la réponse du serveur. Deux traitements devront être prévus. Le premier, `traitementResultat`, permettra de gérer dans Flash le résultat retourné par le serveur, alors que le second, nommé `traitementErreur`, devra gérer les erreurs si le transfert échoue. Avant de définir ces deux gestionnaires, nous devons créer un objet `RelayResponder` afin de définir les noms des deux gestionnaires présentés précédemment. Pour cela, le constructeur de l'objet `RelayResponder` utilise comme second et troisième paramètres le nom des méthodes gérant les objets `ResultEvent` et `FaultEvent` renvoyés par la méthode de service. En effet, lorsqu'un appel à une fonction de service aboutit, un objet `ResultEvent` est renvoyé sous forme d'argument à la méthode de gestion des résultats. L'objet `ResultEvent` dispose d'une propriété `result`, qui stocke l'objet résultat renvoyé par la fonction de service. Dans notre exemple, le résultat sera matérialisé par une simple chaîne de caractères (récupérable avec `resultat.result`), mais, dans d'autres cas, celui-ci pourra contenir un résultat d'un type différent. Pour cette raison, la gestion de l'objet résultat exige une connaissance de la fonction de service PHP concernée afin de bien définir le type de résultat retourné.

```
//-------------------Gestion du résultat asynchrone et définition
//-------------------des méthodes de traitement des résultats et des erreurs
monRetour.responder = new RelayResponder(this, "traitementResultat", "traitementErreur");
```

```
//----------------Méthode de gestion du résultat
function traitementResultat( resultat:ResultEvent ):Void {
//Recevoir le résultat
trace( "Resultat reçu cote Flash :"+resultat.result  );
}
//----------------Méthode de gestion des erreurs
function traitementErreur( erreur:FaultEvent ):Void {
//Recevoir l'erreur
trace("Erreur connexion :"+erreur.fault.faultstring);
}
```

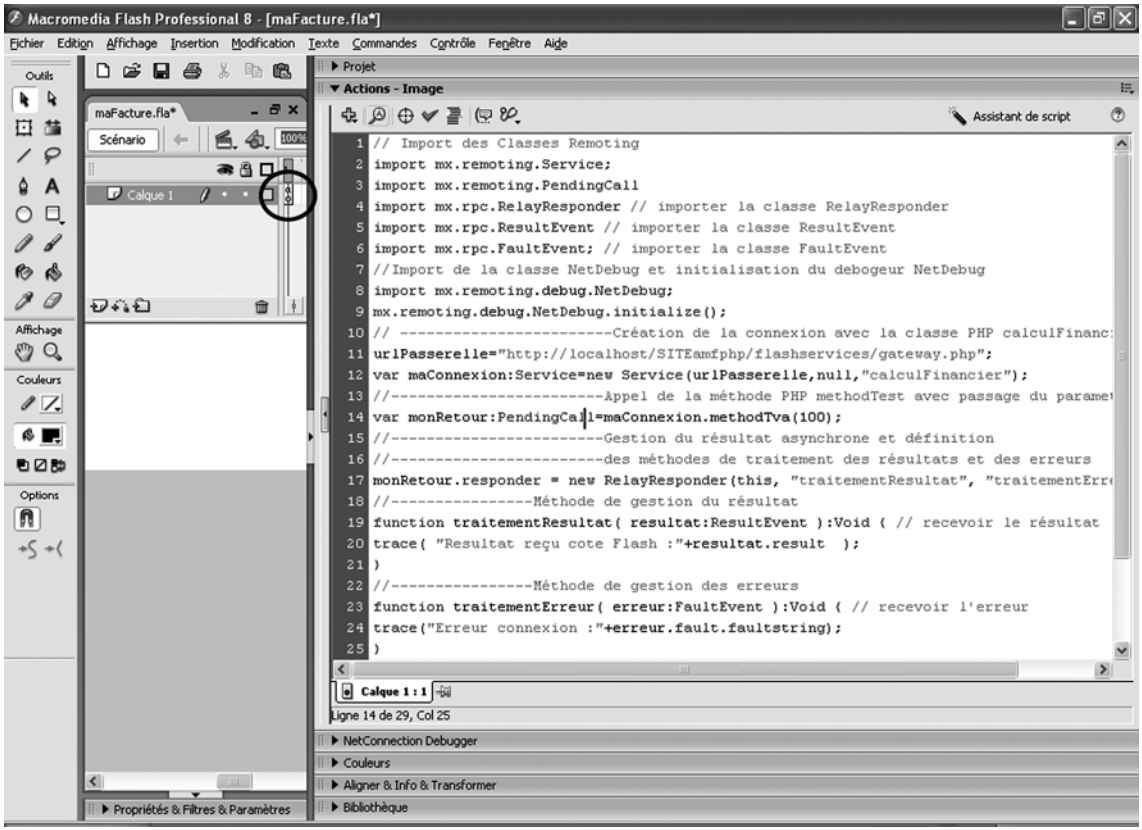


Figure 25-10

Configuration de l'application Flash maFacture.fla .

Test de l'application avec NetDebug

Maintenant que la classe de service PHP est définie et que l'application Flash a été configurée correctement, nous pouvons passer à la phase de test du système de communication. Pour cela, nous allons mettre en œuvre le débogueur NetConnection Debugger. Depuis le menu de Flash activez le panneau du débogueur : `Fenêtre>Autres panneaux>NetConnection Debugger`, puis accrochez le panneau du débogeur en dessous du panneau `Action` (voirfigure 25-11).

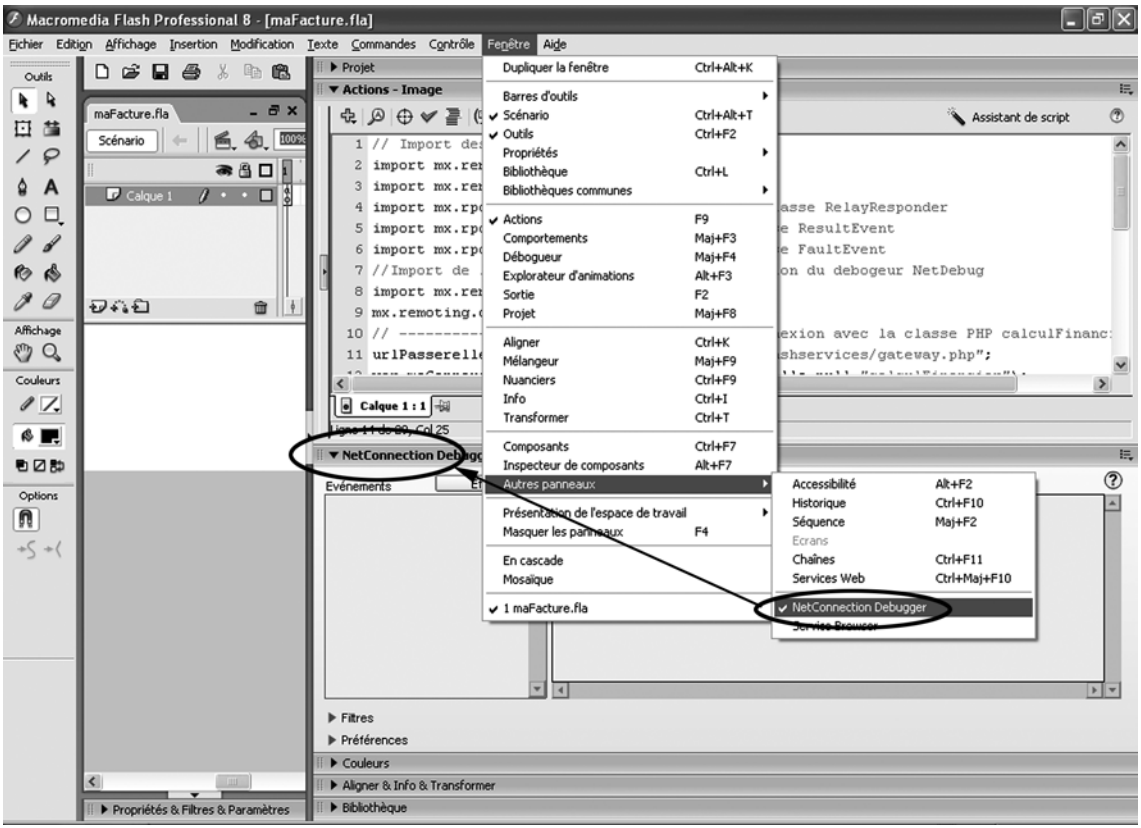


Figure 25-11

Activation du panneau NetConnection Debugger.

Enfin, testez l'application Flash (`Ctrl+Entrée`) : la fenêtre de l'application `maFacture.swf` doit apparaître, ainsi que le panneau de sortie dans lequel doit figurer le résultat renvoyé par le serveur (voir figure 25-12) :

```
Resultat reçu cote Flash : Voici le montant de la TVA = 19.6
```

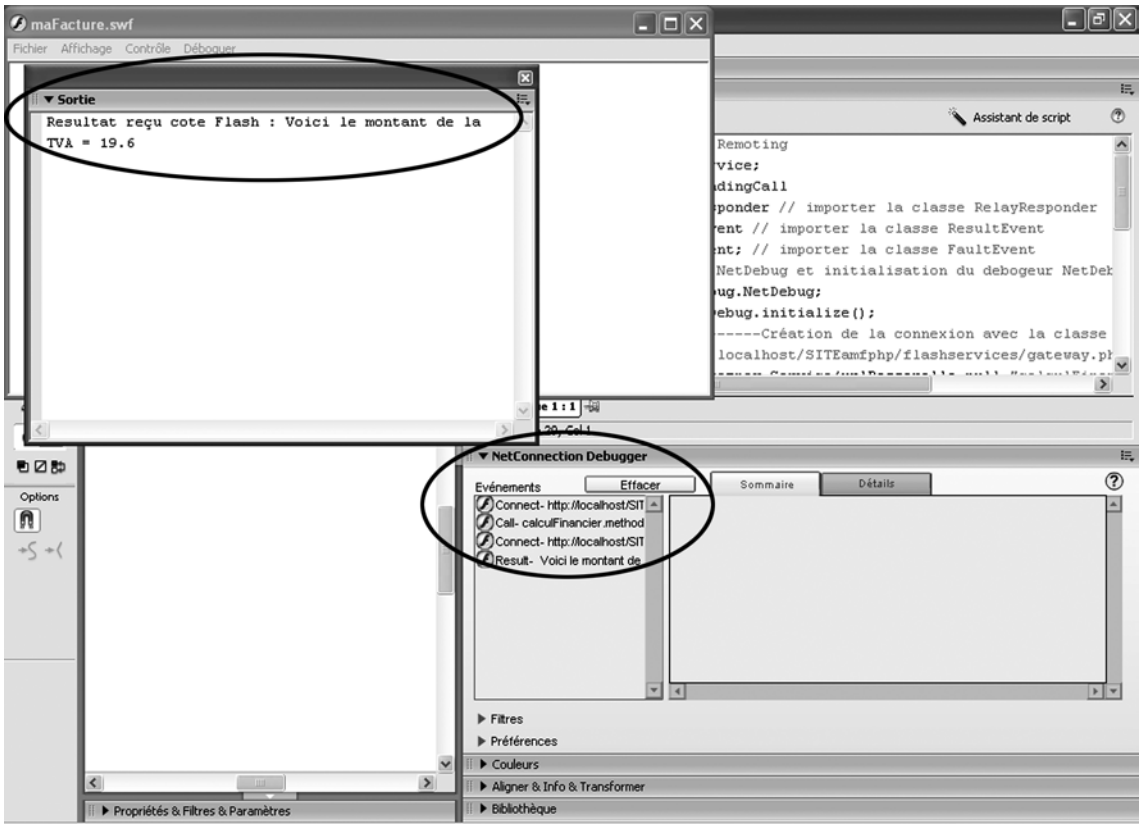


Figure 25-12

Test de l'application maFacture.swf.

Dans le panneau `NetConnection Debugger`, vous devez voir apparaître 4 événements (pour consulter le détail d'un événement, cliquez sur le nom de l'événement concerné dans la zone `Événements`, voir figure 25-13). Le premier correspond à la connexion établie entre Flash et le serveur (initiée par `var maConnexion:Service=new Service()`), le second correspond à l'envoi du paramètre (soit la valeur numérique 100 dans notre exemple), le troisième indique la connexion de la réponse asynchrone du serveur, et le quatrième correspond à la réponse retournée par le serveur (soit, dans notre exemple, le montant de la TVA correspondant à la valeur 100 : `Voici le montant de la TVA = 19.6`)

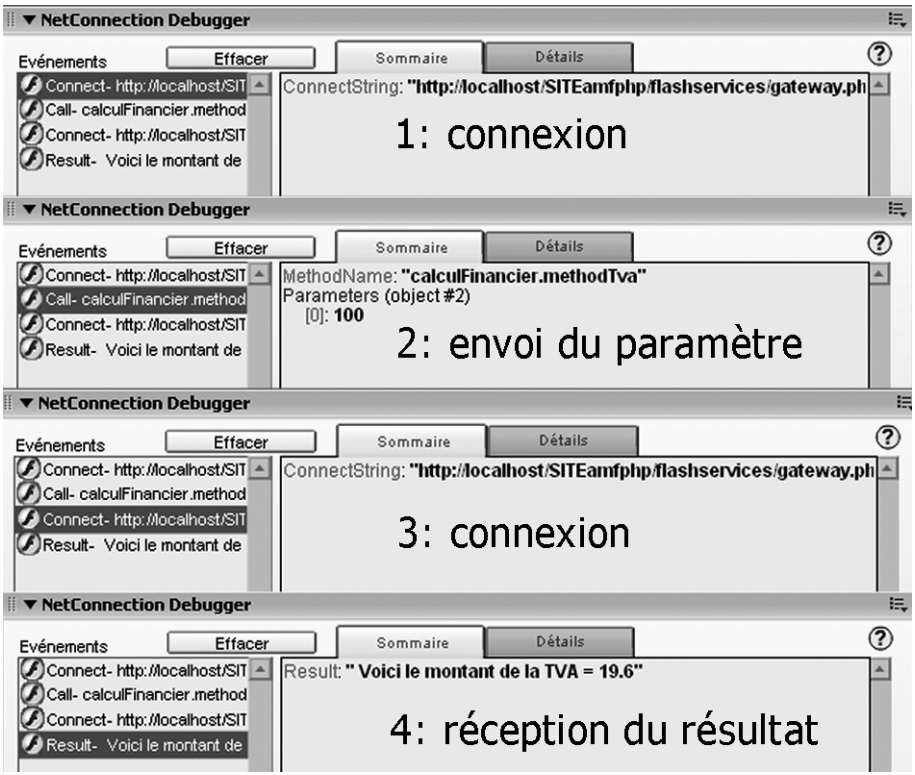


Figure 25-13

Événements générés par NetConnection Debugger.

Connexion de type RecordSet avec AMFPHP

Afin d'illustrer l'utilisation d'AMFPHP et de RecordSet qui permettent d'interfacer une base de données MySQL, nous allons réaliser une application qui aura pour fonction d'afficher la liste des différents adhérents de la base de données `sport_db`, créée au chapitre 16 de cet ouvrage.

Création des classes service AMFPHP

Pour ce second exemple, nous allons créer deux classes PHP. La première sera une classe dédiée à la connexion MySQL et nous la nommerons `connexionMysql` (ainsi, si besoin, elle pourra être utilisée par d'autres classes de services par la suite) ; la seconde sera la classe service proprement dite, elle se nommera `baseSport`, assurera le traitement de la demande client, et renverra le jeu d'enregistrements en retour à l'application Falsh.

Classe connexionMysql

Commençons par créer la classe `connexionMysql` en PHP. Pour cela, ouvrez un nouveau fichier dans Dreamweaver et enregistrez-le dans le répertoire `/services/` (situé dans le dossier `/flashservices/`) en prenant soin d'utiliser le même nom que celui de la classe (donc `connexionMysql.php`).

Le début du fichier de classe commence, comme d'habitude, par le mot-clé `class` suivi du nom de la classe. Suit une série d'initialisations de 4 variables qui contiendra les différents paramètres de connexion à la base de données (revoir si besoin la figure 18-12 pour vous remémorer le script de connexion PHP à une base de données). Notez que les paramètres de connexion utilisés sont les mêmes que ceux du compte utilisateur `sport` , créé dans les chapitres précédents. Si toutefois vous n'avez pas encore créé cet utilisateur, sachez qu'il est aussi possible de se connecter à la base `sport_db` en utilisant le compte `root` pas défaut et en ne mettant pas de mot de passe (attention, ce compte `root` ne doit être utilisé que sur votre serveur local pour des raisons de sécurité évidentes).

```
class connexionMysql
{
  var $hostname = "localhost";
  var $database = "sport_db";
  var $username = "sport";
  var $password = "eyrolles";
```

La première méthode consiste à déclarer ce que doit être le constructeur de la classe, soit `__construct()` en PHP 5(notez que si vous désirez conserver la compatibilté avec PHP 4, vous pouvez aussi utiliser `connexionMysql()` à la place de `__construct()`). Dans ce constructeur, nous placerons deux instructions. La première permettra d'initialiser un identifiant de connexion appelé `$connexion` en se référant à trois des paramètres déclarés au début de la classe (`$hostname`, `$username` et `$password`). La seconde instruction permettra de sélectionner la base de données à utiliser se référant à l'identifiant de connexion précédemment défini (`$connexion`) et au quatrième paramètre contenant le nom de la base de données (`$database`).

```
//Constructeur de la classe
  function __construct()
  {
         $connexion = mysql_pconnect($this->hostname, $this->username, $this->password);
         mysql_select_db($this->database, $connexion) ;
  }
```

En dessous du constructeur, nous devons maintenant déclarer la méthode qui sera utilisée pour soumettre la requête passée en paramètre (`$sql`) à la base de données. Cette méthode renverra le jeu d'enregistrements correspondant en retour (`$rs`).

```
function requete ($sql)
    {
         $rs = mysql_query($sql);
         return $rs;
    }
```

Terminez ensuite la classe en ajoutant une accolade fermante (voir le fichier complet de la figure 25-14). Enregistrez votre fichier et testez-le si besoin en l'appelant depuis le Web Local (l'appel de cette classe de façon isolée doit afficher une page blanche, mais aucun message d'erreur ne doit apparaître. Revoyez le code pour corriger l'erreur dans le cas contraire).

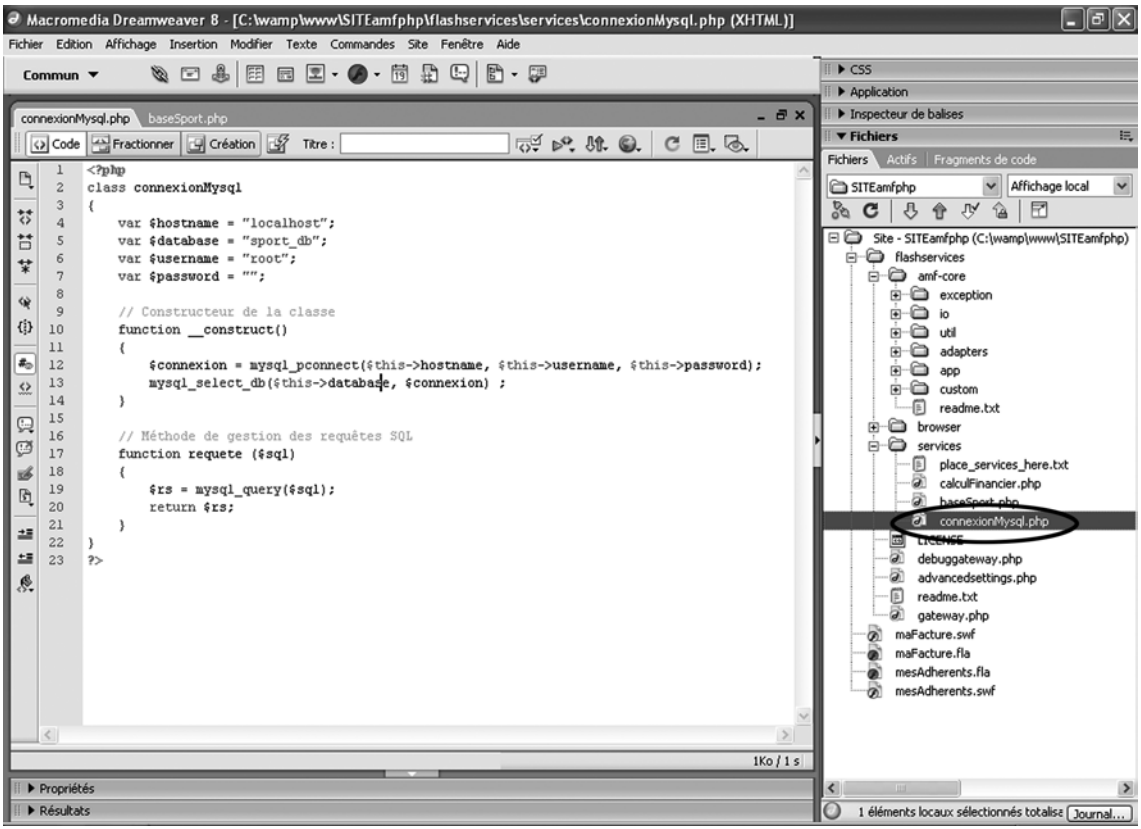


Figure 25-14
Création de la classe service PHP connexionMysql.

Classe baseSport

Avant de commencer l'écriture de la classe, nous ajouterons une instruction au début du fichier pour inclure la classe `connexionMysql` précédemment créée, afin de pouvoir disposer de ses méthodes dans cette nouvelle classe.

```
include_once("connexionMysql.php");
```

Le début du fichier de classe commence comme d'habitude par le mot-clé `class` suivi du nom de la classe, soit `baseSport` dans notre cas. Une fois encore, la première méthode à déclarer dans une classe doit être le constructeur de la classe soit `__construct()` en PHP 5 (notez que si vous désirez conserver

la compatibilité avec PHP 4, vous pouvez aussi utiliser `baseSport()` au lieu de `__construct()`). Comme dans la classe `calculFinancier` de l'exemple précédent, nous placerons dans le constructeur un objet particulier appelé `methodTable`. Cet objet contiendra les différentes méthodes de la classe et leur propriétés respectives (voir figure 25-15). Dans notre exemple, nous n'aurons qu'une seule méthode nommée `listeAdherents` et ses propriétés. Parmi ces propriétés, nous aurons notamment la propriété `description`, qui permettra de préciser l'usage de la méthode, la propriété `access`, configurée avec la valeur `remote` afin d'autoriser Flash à accéder à la méthode, et enfin la propriété `returns`, qui indiquera le type de données retournées par la méthode (soit `RecordSet` dans notre cas).

```
class baseSport
{
  var $connexionSport;
  function __construct() {

        // définition de methodTable
        $this->methodTable = array(
              "listeAdherents" => array(
                    "description" => "renvoi la liste des adhérents",
                    "access" => "remote",
                    "returns" => "RecordSet"
              )
        );//fin de methodTable
        $this->connexionSport = new connexionMysql();
  }
```

En dessous du constructeur, nous devons maintenant déclarer la méthode qui définira la requête SQL à utiliser pour générer le jeu d'enregistrements contenant la liste des adhérents (`listeAdherents()`). Dans cet exemple, la méthode n'a pas d'argument. Par la suite, nous verrons qu'il peut être intéressant de passer un paramètre à cette méthode afin de créer des filtres selon une information envoyée par le client Flash. La première ligne de cette méthode permet de mémoriser la requête SQL dans une variable que nous nommerons `$sql`. La seconde ligne appellera la méthode `requete()` de la classe `connexionMysql` afin de soumettre la requête SQL passée en paramètre au serveur de base de données. Le jeu d'enregistrements ainsi récupéré (`$resultat`) sera ensuite retourné par la méthode.

```
function listeAdherents()
  {
        $sql = "SELECT id,nom,prenom,anneeNaissance FROM adherents";
              $resultat = $this->connexionSport->requete($sql);
        return $resultat;
  }
```

Terminez ensuite la classe en ajoutant une accolade fermante (voir le fichier complet de la figure 25-15). Enregistrez votre fichier et testez-le si besoin en l'appelant depuis le Web Local (l'appel de cette classe de façon isolée doit afficher une page blanche et aucun message d'erreur ne doit apparaître. Revoyez le code pour corriger l'erreur dans le cas contraire).

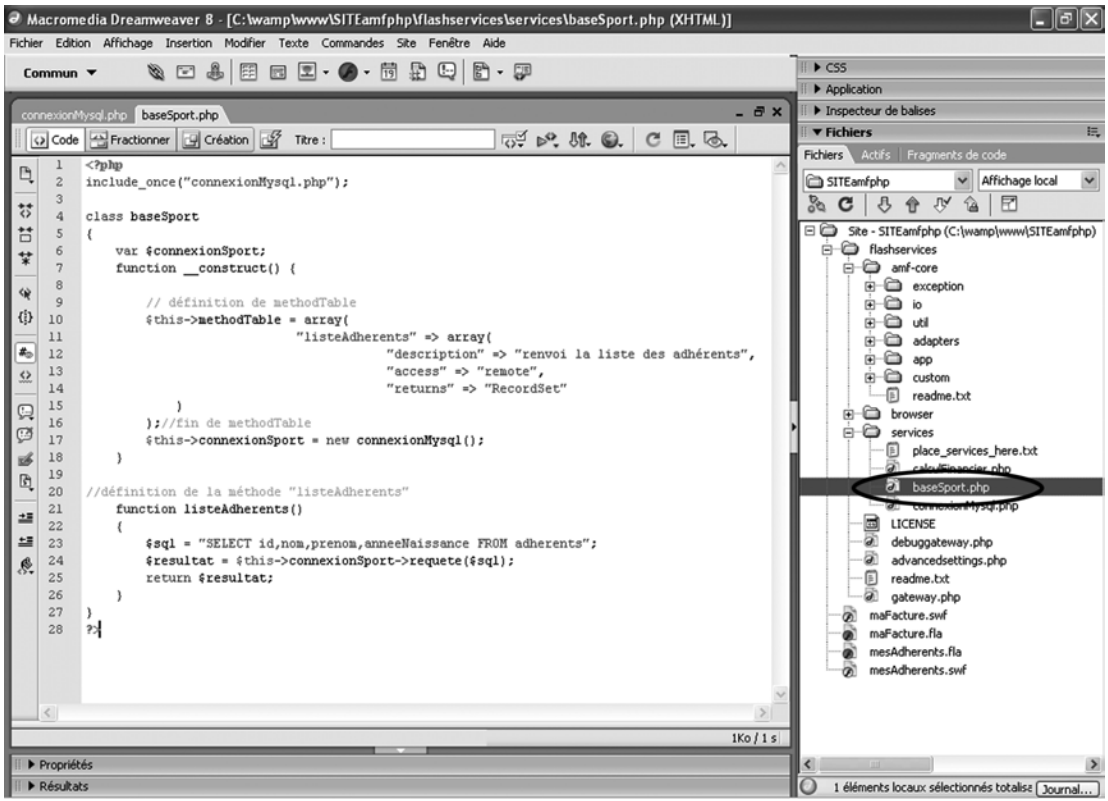


Figure 25-15

Création de la classe service PHP baseSport.

Création d'une application Flash Remoting avec RecordSet

Il faut maintenant créer l'application Flash Remoting. Pour cela, ouvrez un nouveau document Flash, puis enregistrez-le sous le nom `mesAdherents.fla` dans le répertoire `/SITEamfphp/`. Depuis le menu de Flash, ouvrez la bibliothèque commune de `Remoting` (`Fenêtre>Bibliothèques communes>Remoting` : revoir si besoin la figure 25-9). Ouvrez aussi la bibliothèque du document actif (`Ctrl+L`), puis sélectionnez les composants dans la bibliothèque Remoting et faites-les glisser dans la bibliothèque du document actif.

Placez-vous sur la première image du scénario principal et ouvrez ensuite l'éditeur de script pour y copier les instructions d'importation des différentes classes nécessaires au fonctionnement de Flash Remoting (voir code ci-dessous ; pour plus de précision sur ces classes, reportez-vous au tableau 25-2).

```
// Import des Classes Remoting
import mx.remoting.Service; //Importer la classe Service
import mx.remoting.PendingCall //Importer la classe PendingCall
import mx.rpc.RelayResponder //Importer la classe RelayResponder
import mx.rpc.ResultEvent //Importer la classe ResultEvent
import mx.rpc.FaultEvent; //Importer la classe FaultEvent
```