Chapitre 4

Exécution de code par le serveur HTTP

L'un des inconvénients de HTML est qu'un serveur ne peut renvoyer que des pages Web dites **statiques**, c'est-à-dire ne dépendant pas d'un paramètre qui serait envoyé par le client. La seule latitude est le choix de la page. Cette restriction se voit très bien dans le cas des formulaires : nous avons vu comment envoyer les renseignements d'un formulaire par courrier électronique mais nous ne savons pas comment renvoyer 'Bonjour' suivi du nom qui se trouverait dans le formulaire. On a donc conçu des moyens d'obtenir des pages **dynamiques**.

La première méthode consiste à faire exécuter du code par le serveur HTTP et de placer le résultat de celui-ci dans la page Web avant de l'envoyer. Apparue très rapidement dans l'histoire du Web, il s'agit des **directives SSI** (pour *Server Side Includes*). Elles gardent un intérêt, même si elles sont maintenant éclipsées par PHP.

4.1 Mise en place

4.1.1 Configuration du serveur

Pour pouvoir exécuter les directives SSI, il faut que le serveur HTTP puisse les exécuter et qu'il soit paramétré pour cela. Prenons le cas de Apache pour Windows.

Module Include.- Le module SSI doit être chargé au démarrage du serveur. De façon générale, si Apache se trouve, disons dans le répertoire :

```
E:\Program Files\Apache Software Foundation\Apache 2.2\
```

que nous abrégerons désormais en « `Apache` », les binaires se trouvent de façon naturelle dans le sous-répertoire :

```
Apache\bin
```

les fichiers de module se trouvent, quant à eux, dans le répertoire :

```
Apache\modules
```

comme on peut le vérifier.

Pour que le module `include` soit chargé au démarrage du serveur Apache, il faut que dans le fichier `httpd.conf` se trouvant dans le répertoire :

```
Apache\conf
```

se trouve la ligne (décommentée, c'est-à-dire non précédée du caractère '#') :

```
LoadModule include_module modules/mod_include.so
```

Option Includes.- D'autre part, il faut ajouter `+Includes` aux options du répertoire des pages Web par défaut, c'est-à-dire `Apache2.2/htdocs`, ce qui devrait donner :

```
Options Indexes FollowSymLinks +Includes
```

de façon à pouvoir faire exécuter les codes SSI situés dans les fichiers HTML qui se trouveront dans ce répertoire.

Nom des fichiers contenant des SSI.- Enfin, à la fin de la section <`IfModule`>, il faut décommenter les deux lignes :

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

pour enjoindre au serveur d'analyser tout document HTML d'extension `.shtml` avant envoi. Ceci qui obligera à donner l'extension traditionnelle `.shtml` (évidemment pour SSI HTML) aux fichiers HTML contenant une directive SSI.

On peut se passer de cette dernière contrainte en décommentant, entre autre, la ligne suivante :

```
XBitHack on
```

ce qui est cependant à éviter car une telle pratique nuit aux performances du serveur.

4.1.2 Premier exemple

Dans l'exemple suivant, sont affichées la date et l'heure à laquelle la page Web a été envoyée :

```
<HTML>
<HEAD>
<TITLE>Page Web donnant la date et l'heure</TITLE>
</HEAD>
```

```
<BODY>
<H1>Page Web donnant la date et l'heure</H1>
<HR>
Vous avez charg&eacute cette page &agrave cette date :
 <!--#echo var="DATE_LOCAL" -->.
<HR>
</BODY>
</HTML>
```

4.2 Les directives SSI

4.2.1 Syntaxe

Comme nous l'avons vu à propos de notre premier exemple, les directives SSI sont construites en respectant la syntaxe suivante :

```
<!--#directive paramètre="valeur" -->
```

Commentaires.- 1) Il s'agit de commentaires HTML de façon à ce qu'elles ne soient pas interprétées par les serveurs ne les prenant pas en compte ou non paramétrés pour cela.

- 2) La marque de commentaire `<!--` est immédiatement suivie du caractère '`#`' de façon à indiquer qu'il s'agit d'une directive SSI aux serveurs qui les prennent en compte.

- 3) Il doit y avoir un espace entre la valeur et la marque de fin de commentaire `-->`.

- 4) Le tableau suivant donne la liste exhaustive des directives SSI :

Directive	Paramètre	Description
`echo`	*var*	Affiche la valeur de la variable spécifiée, d'environnement ou propre à SSI
`include`		Insère les données textuelles d'un autre document dans le fichier en cours
	file	Chemin d'accès relatif du document
	virtual	Chemin d'accès virtuel du document
`fsize`	*file*	Affiche la taille du fichier spécifié
`flastmod`	*file*	Insère la date et l'heure de la dernière modification du fichier spécifié
`exec`		Lance un programme externe et en insère la sortie dans le document en cours
	cmd	Toute applcation disponible sur le serveur
	cgi	Un programme de CGI
`config`		Configuration de l'outil SSI
	errmsg	Message d'erreur par défaut
	sizefmt	Format d'affichage de la taille du fichier
	timefmt	Format de présentation des reprères temporels

4.2.2 Directive d'affichage

Notre premier exemple a montré l'utilisation de la directive `echo` d'affichage. Les variables propres à SSI sont les suivantes :

Nom	Description
DOCUMENT_NAME	Nom du fichier en cours
DOCUMENT_URI	Chemin virtuel du fichier en cours
QUERY_STRING_UNESCAPED	Demande de recherche non décodée, où le caractère d'échappement « \ » remplace les métacaractères du shell
DATE_LOCAL	Date et heure locales
DATE_GMT	Temps universel
LAST_MODIFIED	Date et heure de la dernière modification du fichier en cours

L'exemple suivant illustre l'utilisation de celle-ci :

```
<HTML>
<HEAD>
<TITLE>Bienvenue</TITLE>
</HEAD>
<BODY>
<H1>Bienvenue sur le serveur <!--#echo var="SERVER_NAME" --></H1>
<HR>
Ce document a pour titre : <!--#echo var="DOCUMENT_NAME" -->.
<BR>
Pour y acc&eacute;der ult&eacute;rieurement, composez l'adresse :
<!--#echo var="DOCUMENT_URI" -->. Pensez &agrave l'inclure dans vos signets !
<P>
Vous avez charg&eacute cette page &agrave cette date :
 <!--#echo var="DATE_LOCAL" -->
<P>
Date de la derni&egrave;re mise &agrave jour :
 <!--#echo var="LAST_MODIFIED" -->.
<HR>
</BODY>
</HTML>
```

4.2.3 Directive d'inclusion de fichiers

Certaines portions de fichiers HTML se retrouvent dans beaucoup de nos pages. Il est peut-être inutile de les réécrire à chaque fois ; il suffit de les écrire très proprement et de les inclure lorsque nécessaire.

Considérons le fichier suivant, nommé `adresse.shtml` :

```
<HR>
<ADRESS>
<PRE>
Patrick C&eacute;gielski
```

```
Universit&eacute Paris-Est Cr&eacute;teil
cegielski@u-pec.fr

Mis &agrave jour le : <!--#echo var="LAST_MODIFIED" -->.
</PRE>
</ADRESS>
```

Insérons-le dans le fichier précédent pour obtenir le fichier `inclusion.shtml` :

```
<HTML>
<HEAD>
<TITLE>Bienvenue</TITLE>
</HEAD>
<BODY>
<H1>Bienvenue sur le serveur <!--#echo var="SERVER_NAME" --></H1>
<HR>
Ce document a pour titre : <!--#echo var="DOCUMENT_NAME" -->.
<BR>
Pour y acc&eacute;der ult&eacute;rieurement, composez l'adresse :
<!--#echo var="DOCUMENT_URI" -->. Pensez &agrave l'inclure dans vos signets !
<P>
Vous avez charg&eacute cette page &agrave cette date :
 <!--#echo var="DATE_LOCAL" -->.

<!--#include file="adresse.shtml" -->
<HR>
</BODY>
</HTML>
```

Son appel affichera l'adresse et la date de dernère mise à jour du fichier `inclusion.shtml`.

4.3 Inconvénient des SSI

L'inconvénient essentiel des SSI est que le serveur HTTP doit lire ligne à ligne le code HTML de toute page à renvoyer (tout au moins celles d'un suffixe déterminé tel que `.html`) pour voir si du code est à exécuter, exécuter celui-ci (et en placer le résultat là où il faut, mais cette dernière étape n'est pas la plus lente). Ceci risque de ralentir fortement un serveur HTTP très sollicité.