Chapitre 6 :

Gestion des fichiers, des Cookies et des Sessions avec PHP

1. Téléchargement d'un fichier d'un client vers le serveur

Un formulaire permet de télécharger un fichier depuis le client vers le serveur lorsque la valeur de l'attribut TYPE de la balise <INPUT> est file.

La balise <FORM> doit alors obligatoirement contenir l'attribut ENCTYPE = "multiport/form-data ".

Le fichier HTML appelle alors un script PHP.

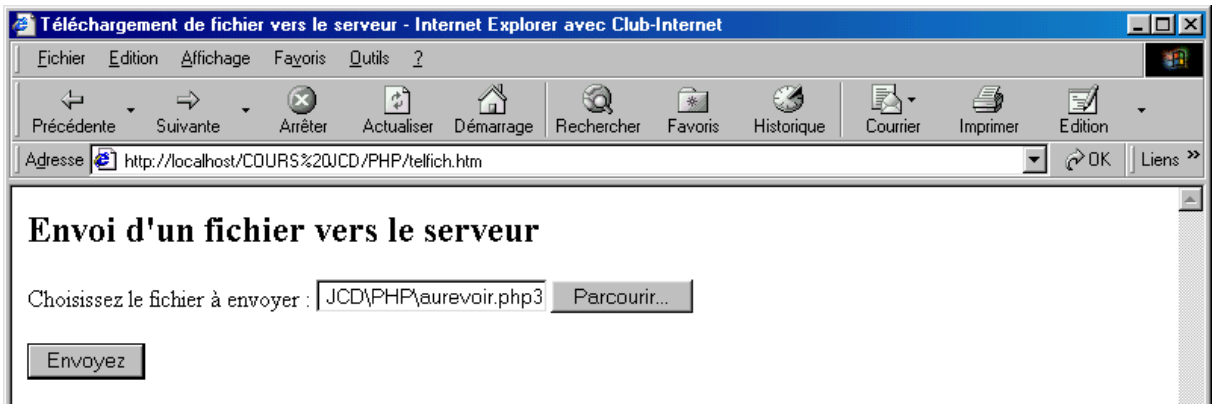
Exemple :

```
<HTML>
<!- telfich.html -!>
<HEAD>
<TITLE>Téléchargement de fichier vers le serveur
</TITLE>
</HEAD>
<BODY>
<H2>Envoi d'un fichier vers le serveur</H2>
<FORM METHOD="post" ACTION="telfich.php3"
      ENCTYPE="multipart/form-data">
Choisissez le fichier à envoyer :
<INPUT TYPE="file" NAME="fichier">
<p>
<INPUT TYPE="submit" VALUE="Envoyez">
</FORM>
</BODY>
</HTML>
```

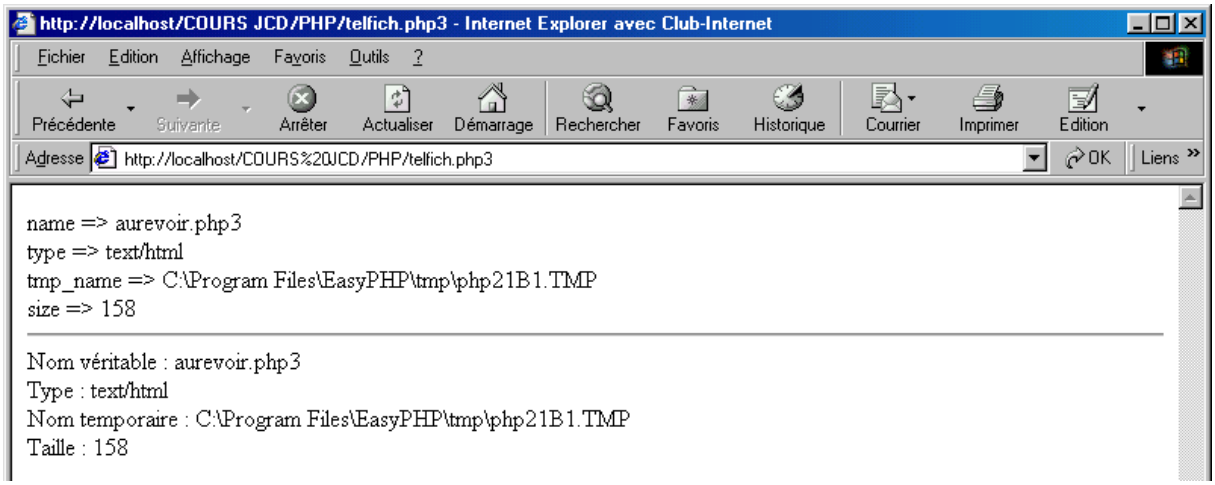
```
<? // telfich.php3
if ($fichier)
{ while (list($k, $v) = each($HTTP_POST_FILES))
  { while (list($kk, $vv) = each ($v))
      echo "$kk => $vv<br>";
  }
  echo "<hr>";

  $a = $HTTP_POST_FILES['fichier']['name'];
  echo "Nom véritable : $a<br>";
  $a = $HTTP_POST_FILES['fichier']['type'];
  echo "Type : $a<br>";
  $a = $HTTP_POST_FILES['fichier']['tmp_name'];
  echo "Nom temporaire : $a<br>";
  $a = $HTTP_POST_FILES['fichier']['size'];
  echo "Taille : $a<br>";
```

```
}
?>
```

Le fichier HTML a l'allure suivante :



En cherchant par exemple le fichier `aurevoir.php3`, puis en cliquant sur `Envoyez` , on obtient le résultat suivant:



2. Cookies à l'aide de PHP

a. Définition et fonctionnement d'un cookie

Les cookies sont des petits fichiers texte que le serveur écrit sur le disque dur du client, afin de mémoriser certaines informations concernant la transaction en cours.

Pour manipuler les cookies, PHP ne propose qu'une seule fonction, set_cookie(), qui écrit un cookie.

La récupération des cookies s'effectue par le navigateur du client lors de la charge d'une URL.

Il recherche alors la présence de cookies relatifs à cette même URL, et, s'il en trouve, les envoie au serveur.

PHP les place alors dans une variable d'environnement nommée `HTTP_COOKIE_VARS` (voir fichier `PHP.INI`).

b. Envoi d'un cookie

Un cookie est envoyé au serveur sous forme d'entête HTTP par un appel à la fonction setcookie. Il est donc nécessaire que cet appel soit la première instruction éxécutée créant des sorties vers le navigateur.

La syntaxe générale d'un cookie est la suivante :

```
set_cookie(nom, <valeur> , <date d'expiration> , <chemin d'accès> ,
                <domaine> , <sécurité>) ;
```

Les champs placés entre < > sont facultatifs. Pour qu'un cookie persiste sur le disque dur, il faut préciser sa durée de vie à partir de l'instant présent (exprimé en secondes) .

Généralement, on ne conserve que les 2 premières options , *<valeur>*et *<date d'expiration>*

Exemples de création de cookies :

```
<? //moncookie.php3
if (empty($MonCookie))
{ setcookie("MonCookie", time(), time()+ 36);
  $msg = "Vous n'avez jamais visité cette page<br>";
}
else
{ $msg = "Votre dernière visite date du ".
         date("d/m/Y", $MonCookie)." à ".
         date("H \h i \m\n s \s", $MonCookie)."<br>";
}
?>
<html>
<head>
<title>Exemple de cookie
</title>
</head>
<body>
<h2>Exemple d'utilisation d'un cookie</h2>
<?PHP echo $msg; ?>
</body>
</html>
```

Le programme affiche le résultat suivant :

Exemple d'utilisation d'un cookie

Vous n'avez jamais visité cette page

Exemple d'utilisation d'un cookie

Votre dernière visite date du 29/03/2003 à 19 h 28 m 10 s

c. Lecture d'un cookie

Le mécanisme peut sembler facile à prori, mais il s'avère compliqué si l'on veut s 'assurer que le client accepte les cookies. Le mieux est alors de recharger un second script juste après l'envoi du cookie.

Exemple :

```
<?//cookie2.php3
```

```
SetCookie("MonTest", time()+10);
Header("Location: testcookie.php3");
?>

<? //testcookie.php3
if(isset($MonTest))
 echo "Le client accepte les cookies";
else
 echo "Le client n'accepte pas les cookies";
?>
```

Le programme affiche le résultat suivant :

Le client accepte les cookies

3. Gestion des sessions avec PHP

a. Utilité des sessions

Une session permet à un navigateur de se "rappeler"des variables utilisées lorsqu'il passe d'une page à une autre (Le mécanisme HTTP ne permet pas cette mémorisation.)

Une session possède une durée de vie.

La propagation des variables de session peut s'effectuer soit par cookies, soit par URL.

La méthode par URL est la plus sure .C'est celle qui est décrite ci-dessous.

Elle s'efectue de la façon suivante :

- Création de la session (éventuellement précédée par la définition d'un nom de session)
- Enregistrement / déclaration des variables de session.
- Utilisation des variables de session : lecture, modification, suppression
- Fermeture de la session avec annulation de l'enregistrement des variables de session.

- ***session_start***

Cette fonction initialise une session ou reprend la session courante d'après son identificateur.

Elle ne reçoit aucun argument et renvoie toujours la valeur TRUE.Elle doit toujours etre appelée avant toute autre sortie vers le navigateur.

- ***session_register***

Cette fonction enregistre les variables dont la liste est donnée en argument:

```
session_register("nom_chaine","mon_numero");
```

- ***session_name***

Cette fonction reçoit en argument (facultatif) un nouveau nom de session et renvoie l'ancien nom sous forme de chaîne de caractères.

Si aucun argument n'est spécifié, l'ancien nom est conservé.

Elle doit etre appelée avant `session_start` ou `session_register`.Un nom de session ne peut contenir que des caractères alphanumériques.

C'est sous ce nom que sera enregistré le cookie de session (si l'on a choisi cette méthode d'enregistrement).

Exemple :

```
$nouveau_nom="Toto";
$ancien_nom=session_name($nouveau_nom);
```

- ***session_unregister***

Cette fonction annule l'enregistrement des variables dont les noms figurent en argument.

Exemple :

```
session_unregister("mon_numero");
```

- ***session_destroy***

Cette fonction détruit toutes les données associées à la session.

Ces données continuent d'exister tant que l'éxécution du script se poursuit.

Elle renvoie `TRUE` si elle est correctement éxécutée et `FALSE` sinon. Elle ne reçoit pas d'argument.

- ***session_write_close***

Met fin à la session en cours et sauvegarde les variables de session.

Elle ne reçoit pas d'argument et ne renvoie rien.

Elle permet de mettre fin au mécanisme de session avant la fin normale d'un script.

Exemple :

```
<? // session1.php3
//$maSession = "toto";
session_start();
$maSession = "toto";
session_register("maSession");
echo "La variable de session \$maSession est enregistrée avec la
valeur \"$maSession\"<br>";
?>
<p>Allez à la <A HREF="session2.php3">page 2</A>.
```

<? //**session2.php3**

```
session_start();
echo "Dans la page 2 nous retrouvons $maSession ".
     "qui vaut \"$maSession\"<br>";
$maSession = "c'est fini";
session_register("maSession");
echo "On donne maintenant à \$maSession la valeur".
     " \"$maSession\"<br>";
     ?>
<p>Allez à la <A HREF="session3.php3">page 3</A>.
```

<? //**session3.php3**

```
session_start();
```

```
echo "Dans la page 3 nous retrouvons \$maSession qui vaut
:",$maSession;
session_unregister("maSession");
echo "<br>L'enregistrement de $maSession a été annulé<br>";
?>
<p>Allez à la <A HREF="session4.php3">page 4</A>.
```

<? // **session4.php3**

```
session_start();
echo "\$maSession vaut : \"$maSession\"<br>";
echo "Dans la page 4 nous ne retrouvons plus  \$maSession<br>";
echo "Clôture de la session<br>";
session_destroy();
?>
```

b. Fonctions des sessions

`session_cache_expire` : Retourne la configuration actuelle du cache expire

`session_cache_limiter` : Lit et/ou modifie le limiteur de cache

`session_decode` : Décode les données de session

`session_destroy` : Détruit une session

`session_encode` : Encode les données de session

`session_get_cookie_params` : Lit la configuration du cookie de session

`session_id` : Let et/ou modifie l'identifiant courant de session

`session_is_registered` : Vérifie si une variable est enregistrée dans la session

`session_module_name` : Lit et/ou modifie le module de session courant

`session_name` : Lit et/ou modifie le nom de la session

`session_readonly` : Initialise une session en mode lecture

`session_register` : Enregistre une variable dans une session

`session_save_path` : Lit et/ou modifie le chemin de sauvegarde des sessions

`session_set_cookie_params` : Modifie les paramètres du cookie de session

`session_set_save_handler` : Configure les fonctions de stockage de sessions

`session_start` : Initialise une session

`session_unregister` : Supprime une variable de la session

`session_unset` : Détruit toutes les variables de session

`session_write_close` : Ecrit les données de session et ferme la session

Les sessions et PHP

On entend régulièrement que http est un protocole sans état ou 'statless'. Cela veut dire que le protocole HTTP ne comprend pas de mécanisme de maintien des états entre deux transactions. Lorsqu'un utilisateur demande une page et puis une autre, HTTP n'offre aucun moyen de spécifier que les deux requêtes émanent du même utilisateur.

Le contrôle de session vise à permettre le suivi d'un utilisateur tout au long d'une session sur un site. Les cookies nous ont déjà permis de faire cela mais avec une certaine limitation au niveau des tailles et avec le risque de contrôle par l'utilisateur. Pour la sauvegarde de données confidentielles ou importantes les sessions sont plus adaptées que les cookies.

Leurs applications sont diverses : authentification d'un visiteur, gérer le panier d'achat d'un client sur une boutique en ligne, etc. On peut ainsi envisager la mise en place de formulaires en plusieurs étapes (page par page) donc les données ne seront pas postées d'une page à une autre mais gardées en session, jusqu'à la validation finale. Dans ce chapitre nous allons voir une à une les principales étapes d'un contrôle de session.

Démarrage d'une session

De base, chaque session est caractérisée par un numéro d'identification ID qui permet d'enregistrer des variables de session. Les contenus des variables de session sont stockés sur le serveur.

L'ID de session est la seule information visible du coté du client. Sous forme d'un nombre aléatoire crypté, un ID de session est générer par PHP lui même. Ce numéro peut être stocké sur l'ordinateur comme un cookie ou bien passé en paramètre par l'URL. Avant d'exploiter la fonctionnalité de PHP relative aux sessions, il faut commencer par ouvrir une session. Trois possibilités nous sont offertes:

La première possibilité est la plus simple de tous. **Elle consiste à appeler la fontion session_start() tout au début du code**. Cette fonction vérifie s'il y a déjà un ID pour la session courante. S'il y en a déjà un, elle charge les variables de session enregistrées pour qu'elles puissent être utilisées dans le corps du programme. Sinon, elle crée un ID de session.

La deuxième possibilité serait d'utiliser la fontion session_register(). Cette option est souvent utilisée dans le cas où la directive register_globals est activée dans le fichier de configuration php.ini.

Une session est automatiquement ouverte lorsque l'on essaie d'enregistrer une variable de session avec cette fonction.

Enregistrement des variables de session

Comme les cookies, les variables de session sont aussi stockées dans un tableau super global. Celui qui correspond aux sessions est $_SESSION. Pour créer une variable de session, on définit un élément dans un de ces tableaux comme suit :

```
$_SESSION['myvar']=5 ;
```

On peut aussi procéder autrement en utilisant la fonction session_register.
Exemple

```
<?php
$var=8 ;
session_register('var') ;
?>
```

Le nom de la variable à enregistrer dont être passée en paramètre de la fonction, et sans le préfixe dollar '$'. La variable var peut alors être utilisée d'un script à un autre jusqu'à la fin de la session.

Utilisation de la variable session

Après avoir ouvert la session, on peut accéder à la variable via le tableau global $_SESSION. Il faut quand même s'assurer que les variables de session ont bien été définies. Vous pouvez par exemple vérifier cela à l'aide de la fonction isset() ou empty(). **On peut aussi vérifier que la variable en question est bien une variable de session**. La fonction session_is_registered('var') retourne true si la variable qui lui est passée en argument est de type session. Elle retourne false dans le cas contraire. Toutefois, si on utilise les tableaux super globaux, on n'a pas à utiliser cette fonction. On peut accéder directement au tableau de la manière suivante :

```
if(isset($_SESSION['var']))...
```

Suppression de la session : unset

Une fois qu'on en a terminé avec une variable de session, on peut la supprimer comme suit en cas d'utilisation des tableaux superglobaux :

```
unset($_SESSION['var']);
```

L'idéal étant de vider cette variable auparavant :

```
$_SESSION['var'] = "";
```

Puis

```
unset($_SESSION['var']);
```

Sinon, il est toujours possible de supprimer cette variable à l'aide de la fonction session_unregister();

Après avoir supprimé toute les variables, il ne reste plus qu'à détruire la session complètement en appelant la fonctions session_destroy().