Utilisation des sessions avec PHP

Qu'est ce que c'est et à quoi sa sert?

Les sessions sont un moyen de stocker des informations relatives au visiteur. Il s'agit d'une alternative aux cookies.

La différence entre les sessions et les cookies est que les cookies sont stockés sur le poste du visiteur tandis que les sessions sont dans des fichiers présents sur le serveur. Il s'agit d'un stockage temporaire mais, les fichiers de sessions ne peuvent pas être vus et modifiés, contrairement aux cookies.

Elles permettent de stocker des types de données simples (texte, nombres, …) mais pas de ressources.

Cela permet de stocker des informations sur le visiteurs donc mais de manière transparente, c'est-à-dire qu'elles n'apparaissent pas dans l'url ou dans des champs cachés et ne peuvent pas être refusées par les visiteurs (contrairement aux cookies).

Comment ça fonctionne ?

Les informations sont stockées dans des fichiers, sur le serveur, à chaque session correspond un fichier. Chaque session est désignée par un nom et un identifiant (l'identifiant est une chaîne de 32 caractères hexadécimale générée par md5).

Lorsque le visiteur accepte les cookies, l'identifiant de la session est stocké dans un cookie, dans le cas contraire, il existe un autre moyen de stocker l'identifiant.

Normalement, la session est détruite ou à la fermeture du navigateur, ou au bout de 30 minutes. Cette valeur peut être modifiée dans le fichier php.ini.

Des exemples

Nous allons voir comment utiliser les sessions dans une suite d'exemple et de bouts de code.

Commencer une session

```
session_start();
```

Ce code permet de démarrer la session. Le navigateur regarde si un cookie existe sur le poste client correspondant à ce site. Si c'est le cas, l'identifiant de la session est lu dans le cookie et envoyé au serveur. Le serveur peut lire le fichier correspondant à cet identifiant et re-créer la session. Tout ceci se passe de manière transparente dans le session_start(). Si le cookie n'existe pas sur le poste client, une nouvelle session est crée.

Notez que code est à placé tout au début de votre page, avant tout code HTML puisqu'on ne doit pas avoir écrit dans le document avant l'appel.

Créer une variable de session

Les variables de sessions sont accessibles, une fois que la sessions est démarrée, via un tableau super global : **$_SESSION** (on utilise plus session_register qui est devenu obsolète).

```
$_SESSION['variable'] = $valeur ;
```

Nous créons ici une variable de session nommée *variable* qui vaut *$valeur*

Savoir si une variable de session existe
Pour savoir si la variable de session *variable* existe, il suffit de faire :

```
if(isset($_SESSION['variable']))
  echo 'La variable "variable" existe !';
```

Remarque : Il ne faut plus utiliser session_is_registered

Utiliser la valeur d'une variable de session
Si vous voulez utiliser la valeur d'une variable de session, vous devez passez par **$_SESSION**

```
if(isset($_SESSION['variable']))
  echo 'La variable "variable" existe et vaut: ' . $_SESSION['variable'];
```

Supprimer une variable de session
Pour supprimer une variable de session, il faut utiliser **unset()** (on utilise plus session_unregister qui est devenu obsolète).

```
unset($_SESSION['variable']);
echo 'La variable de session "variable" est maintenant détruite';
```

Détruire toutes les variables de session
La fonction **session_unset()** détruit toutes les variables de session.

```
session_unset();
```

Détruire une session
La fonction **session_destroy()** permet de détruire une session.

```
session_destroy();
```

Un exemple concret

Nous allons voir ici comment fabriquer une petite zone membre. Je ne vais pas vous donner de code HTML mais juste le PHP brute.

Fichier *index.php*

```
<?php
session_start();  // démarrage de la session

// si la variable de session "pseudo" existe
if(isset($_SESSION['pseudo']))
{
  echo 'Vous êtes connecté en tant que <span style="color: 0000FF;">' .
    $_SESSION['pseudo'] . '</span><br><br>';

  echo '<a href="membre.php">Accéder à la zone membre</a><br><br>';
  echo '<a href="log.php?action=logout">Logout</a><br><br>';
}
else
{
  // si la variable erreur est dans l'url
  if(isset($_GET['erreur']))
  {
    // le compte n'existe pas
    if($_GET['erreur'] == 1)
      echo '<span style="color: FF0000;">Le compte n\'existe pas</span>';
    // mot de passe invalide
```

```
  ?>
  <form action="log.php?action=login" method="post">
  Pseudo: <input type="text" name="pseudo" maxlength="32"><br><br>
  Password: <input type="password" name="password" maxlength="32"><br><br>
  <input type="submit" value="Login">
  </form>
  <?php
}
?>
```

Fichier *log.php*

```
<?php
session_start();

// login
if($_GET['action'] == 'login')
{
  // récupération des variables
  $pseudo = $_POST['pseudo'];
  $password = $_POST['password'];

  /*
  Vérification des données
  Notez qu'il est possible de vérifier ces valeurs
  dans une base de données pour gérer plusieurs utilisateurs
  */
  if($password == 'pass' && $pseudo == 'toto')
  {
    $_SESSION['pseudo'] = 'toto'; // création d'une variable de session
    header("location: index.php"); // redirection
    exit;
  }
  // si le pseudo est faux
  else if($pseudo != 'toto')
  {
    header("location: index.php?erreur=1");
    exit;
  }
  // le mot de passe est faux
  else
  {
    header("location: index.php?erreur=2");
    exit;
  }
}
// logout
else if($_GET['action'] == 'logout')
{
  session_unset(); // suppression des variables de sessions
  session_destroy(); // destruction de la session
  header("location: index.php"); // redirection
}
?>
```

Fichier *membre.php :*

```php
<?php
session_start();

// si la variable de session "pseudo" n'existe pas, le visiteur
// n'a rien à faire ici
if(!isset($_SESSION['pseudo']))
{
  header("location: index.php"); // redirection
  exit; // arrêt du script
}

echo 'Bienvenue dans la zone membre, ' . $_SESSION['pseudo'] . '<br>';
echo '<a href="log.php?action=logout">Logout</a><br><br>';
?>
```

Oui, mais si on accepte pas les cookies ?

Comme nous l'avons dit tout à l'heure, l'identifiant de session est stocké dans un cookie. Donc si le visiteur refuse les cookie, ben la session fonctionne pas !

Nous allons alors vérifier que le client accepte les cookie (en tentant la création d'un cookie) et si ce n'est pas le cas, nous allons par l'identifiant par l'url.

Page1.php

Comme le cookie ne peut être lu qu'une fois le chargement de page suivant, nous allons utiliser une variable **step**, passée dans l'url : Si elle vaut 1, on crée le cookie, si elle vaut 2, on tente de le lire.

Si le visiteur de refuse pas les cookies, on passera l'identifiant par l'url.

```php
<?php
session_start();
// récupération de step
if(!isset($_GET['step']))
  $step = 1;
else
  $step = $_GET['step'];

// création du cookie
if($step == 1)
{
  // création d'un cookie de test
  setcookie('test', '1');
  // redirection
  header("location: page1.php?step=2");
  exit;
}
else if($step == 2)
{
  // si le cookie est défini
  if(isset($_COOKIE['test']))
  {
    // suppression du cookie
    setcookie('test');
    // url
    $url = 'page2.php';
  }
```

```
  // le cookie n'est pas défini
  else
  {
    // on passe en paramètre l'id de la session
    $url = 'page2.php?SID=' . session_id();
  }

  // on crée une variable de session
  $_SESSION['variable'] = 123;

  // lien pour la page suivante
  echo '<a href="' . $url . '">Page 2</a>';
}
?>
```

Page2.php

```
<?php
// si la variable SID est passée par l'url --> le client accepte pas les cookies
// --> on spécifie un identifiant de session
if(isset($_GET['SID']))
    session_id($_GET['SID']);

// on démarre la session
session_start();

// affichage de la valeur de la variable de session
echo 'La variable de session vaut: ' . $_SESSION['variable'];
?>
```

Si une variable **SID** a été passée par l'url, on spécifie cet identifiant de session lieu qu'il soit lu, comme par défaut, dans un cookie qui n'existe pas.

La valeur **123** est ainsi affichée même si le client n'accepte pas le cookie.

Pour la suite, si une variable **SID** a été passée par l'url, il suffit de la rajouter. On peux d'ailleurs créer une petite fonction qui le fait :

```
function url($url)
{
  if(isset($_GET['SID']) && strpos($url, 'SID=') === false)
  {
    if(strpos($url, '?') === false)
      $url .= '?SID=' . $_GET['SID'];
    else
      $url .= '&SID=' . $_GET['SID'];
  }

  return $url;
}
```

```php
<?php
// si la variable SID est passée par l'url --> le client accepte pas les cookies
// --> on spécifie un identifiant de session
if(isset($_GET['SID']))
      session_id($_GET['SID']);

// on démarre la session
session_start();

// affichage de la valeur de la variable de session
echo 'La variable de session vaut: ' . $_SESSION['variable'];

echo '<br><br><a href="' . url('page3.php') . '">Page3</a>';
?>
```

Cette fonction se charge d'ajouter à une url la variable SID si elle a été passée dans l'url précédente. Elle tient compte de la présence éventuelle d'autres variables.

Voici la fin de ce petit cours, si vous avez des questions, des remarques, des critiques, hésitez pas à m'envoyer un mail : coockiesch@yahoo.fr