Chapitre 3

Les Sessions en PHP

Introduction

Une session est une période entre un début et une fin d'une activité.

Par exemple pour Internet j'ouvre mon navigateur sur http://www.php.com et je referme le navigateur : j'ai ainsi réalisé une session que j'ai commencée, puis terminée.

Par extension, on associe à une session un ensemble de valeurs définies de manière transparente pour le visiteur et que le serveur conserve de page en page.

Le maniement des valeurs de sessions est assez simple en PHP, on peut stocker presque tout dans une variable de session : un chiffre, un texte, voir un objet

Utilité des Sessions

Par exemple lorsqu'on désire conserver une variable entre plusieurs pages, mon visiteur rentre sur une page un identifiant et un mot de passe, je lui affiche des liens que je n'affiche pas s'il n'y a pas eu cette "connexion" préalable... Cela permet de réserver des liens à des gens, par exemple, qui ont une inscription à un site.

Autre exemple, je souhaite que le visiteur remplisse un formulaire mais le nombre de champs et leur diversité (informations personnelles, informations administratives, situation géographique...) ne me permet pas de tout afficher sur une même page.

La solution d'afficher chaque jeu de champs sur plusieurs pages se dessine alors, mais comment être sûr que, lorsque le visiteur passe à la page suivante, les valeurs qu'il a rentrées dans la page précédente soient conservées ?

La session répond à ma demande : le serveur va stocker ces valeurs "chez lui" et je peux ainsi y faire appel dès que bon me semble.

Voilà quelques exemples ... Mais comment marchent les sessions *concrètement* ?

Les sessions : généralités

Une session s'ouvre dans une balise PHP par :

```
<?php
        session_start() ;
  ?>
```

Attention à commencer la session en tête du fichier; avant tout Doctype ou autre code HTML qui sera envoyé au client.

Ceci fait, le serveur crée "chez lui" un fichier dans lequel il écrira les variables et leurs valeurs. La session est définie par un identifiant qui est unique et aléatoire, c'est la variable cachée qui est transmise de page en page de manière transparente.

La session est conservée sur le serveur tant qu'elle n'a pas reçu un ordre de destruction ou bien que le client n'a pas fermé son navigateur : elle expire alors au bout d'un certain temps (par défaut 1h).

Pour déclarer une variable de session (ici vide), il suffit d'exécuter :

```
<?php
        $_SESSION['prenom'] = '' ;
?>
```

C'est aussi simple que cela. *(Cette variable ne pourra exister en tant que variable de session que si session_start() ; est exécuté au préalable.)*

on peut faire des tests conditionnels sur ces variables :

```
<?php
        if ((isset($_SESSION['login'])) && (!empty($_SESSION['login'])))
{
        // le login a été enregistré dans la session, j'affiche le lien du profil

echo    '<a href="profil.php" title="Accédez à votre profil">Mon profil</a>';
         }
        else
        {
                // pas de login en session : proposer la connexion
echo '<a href="connexion.php" title="Accès à la page de connexion">Connexion</a>';
}
?>
```

ici, je dis : si la valeur de variable de session $_SESSION['login'] est définie et non vide (isset et !empty) alors j'affiche le lien "profil".

Attention : une variable définie et une variable vide ne sont pas du tout les mêmes choses... Je peux définir une variable sans pour autant la remplir (cf. plus haut : $_SESSION['prenom'] est définie, mais vide).

L'inverse n'est pas pareil : si je dis explicitement :

```
<?php
        $_SESSION[ ' grand-mere ' ] = 'Jeanne';
?>
```

alors PHP va me définir la variable et me la remplir. La définition sera implicite.

Gestion des variables de session

le simple fait de faire un lien a href vers une autre page d'extension .php qui contient également un session_start(); suffit à conserver les variables.

$_SESSION est un tableau (array) typé comme suit : index => valeur.

On peut donc lui faire faire les travaux des tableaux habituels : reset, next, previous, end, foreach ... et afficher sa structure grâce à :

```
<?php
        print_r($_SESSION);
?>
```

On peut supprimer une variable de session en la détruisant :

```
<?php
        // Suppression d'une variable de session
        unset($_SESSION['bidule']);
?>
```

Exemple 1 : affichage des variables de session

Exemple1.php

```
<?php
        session_start();
        $_SESSION['prenom'] = "Mounir";
        $_SESSION['age'] = "inconnu";
        $_SESSION['race'] = "humain";
        echo "<ul>\n";
        foreach($_SESSION as $cle => $valeur)
        {
        echo "
<li><strong>".ucfirst($cle)."</strong><em>".$valeur."</em></li>\n";
        }
        echo "</ul>\n";
?>
```

Explication de l'exemple 1

Je démarre ma session. J'initialise trois variables dans ma session : prenom, age, race. Chacune a sa valeur.
Je décide ensuite d'afficher les variables de session. Je vais donc lire séquentiellement le tableau $_SESSION avec une boucle foreach de la manière suivante :

Pour chaque couple clé/valeur de $_SESSION, tu m'affiches dans un <li> (élément de liste) en gras la clé et en italique sa valeur.

fonction ucfirst : cette fonction (UpperCase First) affiche la valeur qui lui est soumise avec la première lettre en majuscule.

On a ainsi de quoi afficher toutes les variables si besoin, à n'importe quel moment je peux faire appel à $_SESSION['race'] par exemple. Par print_r($_SESSION); on affiche ainsi dans le code source :

Array

([prenom] => Mounir
[age] => inconnu
[race] => humain)

Exemple 2 : cas d'un formulaire de connexion

L'exemple décrit permet par une page de s'identifier, et l'accès à toutes les autres pages sera tributaire de cette identification.

Le principe est, somme toute, assez simple :

- J'initialise ma session
- J'initialise mes variables
- J'affiche mon formulaire, ou j'effectue son traitement : dans un cas, mes variables de session seront vides, dans l'autre, elles ne seront remplies que si le traitement retourne un résultat correct.
- Enfin, si le formulaire de connexion est validé, j'affiche un lien pour passer à la page suivante en "mode connecté". Autrement, pas d'accès aux autres pages.

Code de la page index.php

```
<?php
        session_start();
        $_SESSION['login'] = '';
        $_SESSION['password'] = '';

        if (isset($_POST['submit']))
```

```
{
        // bouton submit pressé, je traite le formulaire
        $login = (isset($_POST['login'])) ? $_POST['login'] : '';
        $pass  = (isset($_POST['pass']))  ? $_POST['pass']  : '';

        if (($login == "mounir") && ($pass == "tata"))
        {
                $_SESSION['login'] = "mounir";
                $_SESSION['password'] = "tata";
echo '<a href="accueil.php" title="Accueil de la section membre">Accueil</a>';
        }
        else
        {
                // une erreur de saisie ...?
                echo '<p style="color:#FF0000; font-weight:bold;">Erreur
de connexion.</p>';
        }
}; // fin if (isset($_POST['submit']))

if (!isset($_POST['submit']))
{
        // Bouton submit non pressé j'affiche le formulaire
        echo '
        <form id="conn" method="post" action="">
                <p><label for="login">Login :</label><input type="text"
id="login" name="login" /></p>
                <p><label for="pass">Mot de Passe :</label><input
type="password" id="pass" name="pass" /></p>
                <p><input type="submit" id="submit" name="submit"
value="Connexion" /></p>
        </form>';
}; // fin if (!isset($_POST['submit']))?>
```

Explication de l'exemple 2

On a choisi d'effectuer le traitement du formulaire avant son affichage pour pouvoir afficher un message d'erreur puis réafficher le formulaire ensuite.

Ainsi lorsque le lien Accueil apparaît, je sais :

- que la personne est correctement identifiée,
- que mes variables de session sont non vides.

Autrement, mes variables restent vides. Mais qu'en est-il de la page **accueil.php** ?

Si un visiteur mal intentionné rentre son adresse directement dans le navigateur pour contourner le formulaire ?

Heureusement, il y a moyen de se protéger. Voici le code de

accueil.php :

```
<?php
        session_start();
        if ((!isset($_SESSION['login'])) || (empty($_SESSION['login'])))
        {
                // la variable 'login' de session est non declare ou vide
                echo '  <p>Petit curieux... <a href="index.php"
title="Connexion">Connexion d abord !</a></p>';
                exit();
        }
?>
```

Dans ce code :

- J'initialise la session (session récupérée si déjà existante)
- Je teste si une des variables de session (dans mon exemple, login) est définie et remplie ... Si non définie ou non remplie, dehors :-) (*Notez la présence de* **exit();** *qui coupe net l'exécution de PHP, ainsi la suite du code source de accueil.php ne sera pas exécutée.*)
-

- Je mets ce test tout en haut de page, pour que ça soit le premier élément vérifié par le serveur

Si je mets ces quelques lignes dans toutes les pages que je veux protéger, alors ma "section membre" est presque finie ...

1. Principe du log-out, fin de session

Explication

La fin d'une session est la destruction du tableau de variables $_SESSION.

Je préfère, à titre personnel, réécrire ce tableau puis le détruire. Voyez ci-dessous :

```
<?php
        // Destruction de la session : manière simple
        session_destroy();

        // Destruction de session : (utilisez l'une ou l'autre)
        $_SESSION = array(); // on réécrit le tableau
        session_destroy(); // on détruit le tableau réécrit
?>
```