Fascicule de TP N°8 : les sessions

Une session c'est quoi ?

Une session est un mécanisme technique permettant de sauvegarder **temporairement sur le serveur** des informations relatives à un internaute. Ce système a notamment été inventé pour palier au fait que le protocole HTTP agit en mode **non connecté**. A chaque ouverture de nouvelle session, l'internaute se voit attribué un identifiant de session. Cet identifiant peut être transmis soit en GET, POST ou Cookie (cookie sur poste client) selon la configuration du serveur. Les informations seront quant à elles transférées de page en page par le serveur et non par le client. Ainsi, la sécurité et l'intégrité des données s'envoient améliorées ainsi que leur disponibilité tout au long de la session. Une session peut contenir tout type de données : nombre, chaîne de caractères et même un tableau.

Contrairement à une base de données ou un système de fichiers, la session conserve les informations pendant quelques minutes. Cette durée dépend de la configuration du serveur mais est généralement fixée à 24minutes par défaut. Le serveur crée des fichiers stockés dans un répertoire temporaire.

Parmi les utilisations les plus courantes des sessions, on trouve :

- Les espaces membres et accès sécurisés avec authentification
- Gestion d'un caddie sur un site de vente en ligne
- Formulaires éclatés sur plusieurs pages
- Stockage d'informations relatives à la navigation de l'utilisateur (thème préféré, langues...)

La théorie, c'est bien beau mais en pratique comment ça se passe ? Le chapitre suivant explique l'initialisation et la restauration d'une session ouverte.

Initialisation (et restauration) d'une session

PHP introduit nativement une unique fonction permettant de démarrer ou de continuer une session. Il s'agit de ***session_start().*** Cette fonction ne prend pas de paramètre et renvoie toujours **true**. Elle vérifie l'état de la session courante. Si elle est inexistante, alors le serveur la crée sinon elle la poursuit.

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

Lecture et écriture d'une session

Le tableau $_SESSION

Lorsqu'une session est créée, elle est par défaut vide. Elle n'a donc aucun intérêt. Il faut donc lui attribuer des valeurs à sauvegarder temporairement. Pour cela, le langage PHP met en place le tableau super global **$_SESSION**. Le terme *super global* signifie que le tableau a une visibilité maximale dans les scripts. C'est à dire que l'on peut y faire référence de manière globale comme locale dans une fonction utilisateur sans avoir à le passer en paramètre. Le tableau **$_SESSION** peut-être indexé numériquement mais aussi associative ment. En règle générale, on préfère la seconde afin de pouvoir donner des noms de variables de sessions claires et porteuses de sens.

L'écriture de session

Pour enregistrer une nouvelle variable de session, c'est tout simple. Il suffit juste d'ajouter un couple clé / valeur au tableau $_SESSION comme l'illustre l'exemple suivant.

```
<?php
// Démarrage ou restauration de la session
session_start();
// Ecriture d'une nouvelle valeur dans le tableau de session
$_SESSION['login'] = 'Salah';
?>
```

Le tableau $_SESSION, qui était vide jusqu'à présent, s'est agrandit dynamiquement et contient maintenant une valeur (**Salah**) à la clé associative **login**. Une variable de session est alors créée.

Note de rappel : à place de la chaîne de caractères «Salah», nous aurions pu mettre un nombre, un booléen ou encore un tableau par exemple.

Lecture d'une variable de session

Après l'écriture, c'est au tour de la lecture. Il n'y a rien de plus simple. Pour lire la valeur d'une variable de session, il faut tout simplement appeler le tableau de session avec la clé concernée.

L'exemple ci-dessous illustre tout ça.

```
<?php
// Démarrage ou restauration de la session
session_start();
// Lecture d'une valeur du tableau de session
echo $_SESSION['login'];
?>
```

Cette instruction aura pour effet d'afficher à l'écran la chaîne de caractères **Salah**.

Destruction d'une session

Comme cela a été évoqué plus haut, le serveur détruit lui même la session au bout d'un certain temps si la session n'a pas été renouvelée. En revanche, il est possible de forcer sa destruction grâce à la fonction **session_destroy().** Cela permet par exemple aux web masters de proposer une page de déconnexion aux membres logués à leur espace personnel. Cependant, l'utilisation de session_destroy() seule n'est pas très "propre". Le code suivant présente une manière plus correcte de mettre fin à une session.

```
<?php
// Démarrage ou restauration de la session
session_start();
// Réinitialisation du tableau de session
// On le vide intégralement
$_SESSION = array();
// Destruction de la session
session_destroy()
// Destruction du tableau de session
unset($_SESSION);
?>
```

Pour être convaincu de la destruction de la session, il suffit juste d'essayer d'afficher le contenu du tableau de session au moyen de la fonction **print_r().**

Présentation du cas pratique

Le présent paragraphe introduit un cas concret d'utilisation des sessions. Il s'agit d'un accès restreint basique. Seul un utilisateur n'est autorisé à être logué mais cet exemple est à la base de la création d'un espace membre. C'est exactement la même chose. Pour réaliser tout ça, nous aurons besoin de 2 fichiers : le formulaire accompagné de son script de login, et la page protégée. Commençons par le formulaire de login. Le code étant commenté, il n'y aura pas plus d'explications.

Formulaire d'authentification : authentification.php

```
<?php
// Definition des constantes et variables
define('LOGIN','toto');
define('PASSWORD','tata');
$errorMessage = '';
// Test de l'envoi du formulaire
if($_POST) {
// Les identifiants sont transmis ?
if(!empty($_POST['login']) && !empty($_POST['password'])) {
// Sont-ils les mêmes que les constantes ?
if($_POST['login'] !== LOGIN) {
$errorMessage = 'Mauvais login !';
}
elseif($_POST['password'] !== PASSWORD) {
$errorMessage = 'Mauvais password !';
}else{
// On ouvre la session
session_start();
// On enregistre le login en session
$_SESSION['login'] = LOGIN;
// On redirige vers le fichier admin.php
header('Location: admin.php');
exit();}}
else{// si les champs sont vides
$errorMessage = 'Veuillez inscrire vos identifiants svp !';
}}?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head><title>Formulaire d'authentification</title></head>
<body><form action="<?php echo
htmlspecialchars($_SERVER['PHP_SELF']); ?>" method="post">
<fieldset><legend>Identifiez-vous</legend>
<?php
// Rencontre-t-on une erreur ?
if(!empty($errorMessage)) {
echo htmlspecialchars($errorMessage);
}?>
<p><label for="login">Login :</label><input type="text" name="login" id="login"value=""
/></p>
```



```
<p><label for="password">Password :</label><input type="password"
name="password"id="password" value="" />
<input type="submit" name="submit"value="Se logguer" /></p>
</fieldset>
</form>
</body></html>
```

Exemple de page protégée : admin.php

```
<?php
// On prolonge la session
session_start();
// On teste si la variable de session existe et contient une valeur
if(empty($_SESSION['login'])) {
// Si inexistante ou nulle, on redirige vers le formulaire de login
header('Location: authentification.php');
exit();
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
<title>Administration</title>
</head>
<body>
<?php
// Ici on est bien loggué, on affiche un message
echo 'Bienvenue ', $_SESSION['login'];
?>
</body>
</html>
```

Travail demandé :

0-Tester l'exemple précédant

1- Créer une base de données MySQL « enseignement »

2- Créer la table Compte suivante :

Champ	Type
code_connexion	entier
login	Chaine de 25 caractères
mot_de_passe	Chaine de 35 catactères
role	Chaine de 30 caractères

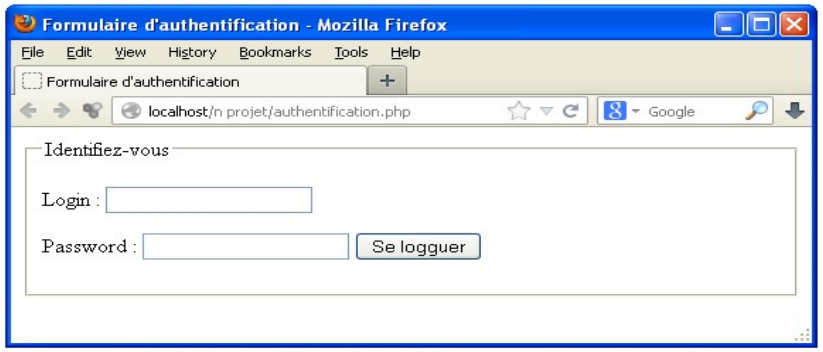
3-Remplir la table par les données suivantes:

code_connexion	login	mot_de_passe	role
1	enseignant	enseignant	ens
2	eleve	eleve	ele
3	Enseignant2	Enseignant2	ens
4	Eleve2	Eleve2	ele

4-Développer l'ensemble des pages PHP suivantes :

1)-Compte.php : c'est la classe qui contient les différents méthodes de la mise à jours dans la BD .

2)-authentification.php qui affiche la fenêtre ci-dessous et qui assure la connexion aux différentes sessions de la table connexion. La page doit contenir un bouton **Se logguer** qui permet d'ouvrir la page **accueil.php** (voir après) si le login et le mot de passe sont corrects.



3)-Page accueil.php :
Le contenu de cette page doit dépendre du type de l'utilisateur.

Pour l'élève :

4)-Page deconnexion.php : qui permet de fermer la session et de revenir à la page d'accueil