PHP: SESSIONS ET REDIRECTIONS

LES REDIRECTIONS

Les redirections sont des en-têtes HTTP. Or, selon le protocole HTTP, les en-têtes HTTP doivent être envoyés avant tout autre type de contenu, ce qui signifie qu'aucun caractère ne doit être envoyé avant l'appel de la fonction header, pas même un espace !
En d'autre termes, **la fonction *header()* doit impérativement être utilisée avant tout code HTML**.
Pour plus d'informations, lire cet article : http://www.commentcamarche.net/faq/1916-php-headers-already-sent-by

Pour rediriger le visiteur vers une autre page (particulièrement utile dans une boucle conditionnelle), il suffit d'utiliser le code suivant :

```
<?php
  header('Location: autre page.php');
?>
```

Où autre*page.php* représente l'adresse de la page vers laquelle vous voulez rediriger. Cette adresse peut être absolue et peut également posséder des paramètres de la forme :

header('Location: autre*page.php?param1=val1&param2*=val2');

LES SESSIONS

Les sessions constituent un moyen de conserver des variables sur toutes les pages de votre site. Jusqu'ici, nous étions parvenus à passer des variables de page en page via les méthodes :

- `GET` (en modifiant l'URL : `page.php?variable=valeur`)
- `POST` (à l'aide d'un formulaire).

Mais imaginez maintenant que vous souhaitez transmettre des variables sur toutes les pages de votre site pendant la durée de la présence d'un visiteur. Ce ne serait pas facile avec `GET` et `POST` car ils sont plutôt faits pour transmettre les informations une seule fois, d'une page à une autre. On sait ainsi envoyer d'une page à une autre le nom et le prénom du visiteur, mais dès qu'on charge une autre page ces informations sont « oubliées ». C'est pour cela qu'on a inventé les sessions.

Fonctionnement des sessions (http://www.siteduzero.com/informatique/tutoriels/concevez-votre-site-web-avec-php-et-mysql/les-sessions)
Comment sont gérées les sessions en PHP ? Voici les trois étapes à connaître.

1. Un visiteur arrive sur votre site. On demande à créer une session pour lui. PHP génère alors un numéro unique. Ce numéro est souvent très gros et écrit en hexadécimal, par exemple : a02bbffc6198e6e0cc2715047bc3766f.
2. Une fois la session générée, on peut créer une infinité de variables de session pour nos besoins. Par exemple, on peut créer une variable `$_SESSION['nom']` qui contient le nom du visiteur, `$_SESSION['prenom']` qui contient le prénom, etc. Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée. Cela veut dire que, quelle que soit la page de votre site, vous pourrez récupérer par exemple le nom et le prénom du visiteur via la superglobale `$_SESSION` !

3. Lorsque le visiteur se déconnecte de votre site, la session est fermée et PHP « oublie » alors toutes les variables de session que vous avez créées. Il est en fait difficile de savoir précisément quand un visiteur quitte votre site. En effet, lorsqu'il ferme son navigateur ou va sur un autre site, le vôtre n'en est pas informé. Soit le visiteur clique sur un lien « Déconnexion » (que vous aurez créé) avant de s'en aller, soit on attend quelques minutes d'inactivité pour le déconnecter automatiquement .La session conserve les informations pendant quelques minutes. Cette durée dépend de la configuration du serveur mais est généralement fixée à 24 minutes par défaut. Le serveur crée des fichiers stockés dans un repertoire particulier.

Tout ceci peut vous sembler un peu compliqué, mais c'est en fait très simple à utiliser. Vous devez connaître quelques fonctions :

- `session_start()` : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui. Vous devez appeler cette fonction au tout début de chacune des pages où vous avez besoin des variables de session.
    - `$_SESSION['login'] = 'toto' ; // création d'une variable de session login`
- `session_destroy()` : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes, mais vous pouvez aussi créer une page « Déconnexion » si le visiteur souhaite se déconnecter manuellement. On peut également s'en fermer la session détruire une variable de session avec la fonction **unset**
    - `unset ($_SESSION['login']);`

ATTENTION : il faut appeler `session_start()` sur chacune de vos pages AVANT d'écrire le moindre code HTML (avant même la balise `<!DOCTYPE>`). Si vous oubliez de lancer `session_start()`, vous ne pourrez pas accéder aux variables `$_SESSION`. Si une session est déjà lancée à partir d'une autre page, celle-ci est reconduite, sinon une session est créée.

MISE EN OEUVRE DES SESSIONS ET REDIRECTIONS

Les sessions et redirections sont particulièrement utilisées pour les accès sécurisés avec authentification.

authentification.php

```
<form action="login.php" method="post">
        Votre login : <input type="text" name="login"><br />
        Votre mot de passe : <input type="password" name="pwd"><br />
        <input type="submit" value="Connexion">
</form>
```

Les données du formulaire sont envoyées à la page login.php, ils seront récupérés via la méthode POST :

- $_POST['login']
- $_POST['pwd']

login.php

```
<?php
    require_once('fonctionsPdo.php');
    // on teste si nos variables sont définies et remplies
   if (isset($_POST['login']) && isset($_POST['pwd']) && !empty($_POST['login'])&& !
empty($_POST['login'])) {

      // on appele la fonction getAuthentification en lui passant en paramètre le login et password
      //la fonction retourne les caractéristiques du salaries si il est connu sinon elle retourne false
      $result = getAuthentification($_POST['login'],$_POST['pwd']);
      print_r($result);
      // si le résulat n'est pas false
      if($result){
           // on la démarre la session
           session_start ();
           // on enregistre les paramètres de notre visiteur comme variables de session
           $_SESSION['nom'] = $result->nom;
           $_SESSION['identifiant'] = $result->idsalaries;
           $_SESSION['role'] = $result->role;
           // on redirige notre visiteur vers une page de notre section membre
           header ('location: listeSalariesPdo.php');

      }
      //si le résultat est false on redirige vers la page d'authentification
      else{
            header ('location: authentification.php');
      }
    }

    //si nos variables ne sont pas renseignées on redirige vers la page d'authentification
    else {
         header ('location: authentification.php');
    }
 ?>
```

```
function getAuthentification($login,$pass){
  global $pdo;
  $query = "SELECT * FROM salaries where login=:login and password=:pass";
  $prep = $pdo->prepare($query);
  $prep->bindValue(':login', $login);
  $prep->bindValue(':pass', $pass);
  $prep->execute();

  // on vérifie que la requête ne retourne qu'une seule ligne
  if($prep->rowCount() == 1){
    $result = $prep->fetch();
    return $result;
  }
  else
  return false;
}
```

listeSalariesPdo.php

```
<?php
  require_once('fonctionsPdo.php');
   // On récupère la session
  session_start ();
  // On teste pour voir si nos variables de session ont bien été enregistrées
  if (isset($_SESSION['nom']) && isset($_SESSION['role'])) {
    echo "<p style=text-align:right;>Bienvenue : ".$_SESSION['nom']."(".$_SESSION['role'].")";
    echo '<br><a href="./logout.php">Deconnexion</a></p>';
    }
  else
     header ('location: authentification.php');
```

Dans le script listeSalariesPdo.php, on teste si les variables de session ont bien été enregistrées. Si c'est le cas on affiche les variables de session nom et role, ainsi qu'un lien vers une page de déconnexion : logout.php. Dans le cas contraire on redirige le visiteur vers la page d'authentification.

logout.php

```
<?php
   // On récupère la session
   session_start ();
    // On détruit les variables de notre session
   session_unset ();
    // On détruit notre session
   session_destroy ();
    // On redirige le visiteur vers la page d'accueil
   header ('location: authentification.php');
?>
```

TRAVAIL À FAIRE :

1 – Télécharger le nouveau script SQL : salaries2.sql
2 – Modifier vos pages PHP pour respecter les cas d'utilisation suivants. Vous pouvez télécharger la correction du TP précédent (cor-pdo.tar.gz)
3 – Pour modifier une page en fonction du rôle de la personne (user ou admin), vous pouvez tester la variable de session : rôle.

```
<?php if($_SESSION['role']=='admin'): ?>
        <th>delete</th>
       <th>update</th>
<?php endif; ?>
```