Les bases de données en PHP

1. Base de données

Nous allons nous baser ici sur une connexion à un SGBD distant de type mySQL avec une interrogation de la base en SQL.

a. Ouverture d'une base

Il existe essentiellement deux fonctions permettant l'accès à une base. La première, **mysql_connect** permet de se connecter au SGBD, la seconde **mysql_select_db** permet de sélectionner une base. **Exemple**

```
<?  $bdd= "mabase"; // Base de données
    $host= "sql.iutbayonne.univ-pau.fr";
    $user= "root";  // Utilisateur
    $pass= "12345";  //mot de passe
    mysql_connect($host,$user,$pass)  or  die  ("Impossible  de  se
     connecter à  la base de données");
    mysql_select_db($bdd);  ?>
```

L'exemple précédent va permettre de se connecter sur le SGBD mySQL situé à l'adresse sql.iutbayonne.univ-pau.fr, avec une identification personnelle. On n'oubliera pas donc de fermer l'accès à la BD lorsque l'on en aura terminé avec elle, avec **msql_close( )**.

b. Requêtes SQL

```
$query =    "SELECT num, pays, date, circuit FROM $nomtable ";
//$nomtable contient le nom de la table.
$result= mysql_query($query);
```

Bien qu'ici la requête est mise dans une variable, il est tout à fait possible de l'inclure directement dans la commande **mysql_query**.
Pour tester si tout c'est bien passé , on appelle la fonction **mysql_error** qui récupère le dernier message d'erreur retourné par une fonction mySQL.

```
if (mysql_error(){ print  "Erreur dans la BD : ".mysql_error();
    exit(); }
```

c. Récupération de données

La fonction **mysql_fetch_row( )** retourne un tableau qui représente tous les champs d'une rangée de résultat (un tuple). Chaque appel produit le tuple jusqu'à ce qu'il n'y en ait plus.

```
while  ($row=mysql_fetch_row($result))  //  $result  a  été  obtenu  par  le
msql_query précédent.
    { // récupération des informations
    $num = $row[0];
    $pays = $row[1];
    $date =  $row[2];
    $circuit =  $row[3];    …  ; }
```

A noter qu'au lieu d'utiliser les indices 0, 1, ... il est possible de donner le nom du champ ($row['pays']) à la condition d'utiliser **mysql_fetch_array()** en lieu et place de mysql_fetch_row().

d. Autres fonctions

mysql_num_rows( ) retourne le nombre de lignes d'un résultat obtenu par une requête.

2. Transmission d'informations et formulaires

Il est possible de transmettre des informations à un script PHP pour modifier son comportement. Ces informations peuvent être par exemple le nom de l'utilisateur ou les réponses d'un formulaire.

2.1. GET et POST

Dans un formulaire de type **GET**, les informations sont passées directement par **l'URL**. Avec la méthode **POST** les informations sont transmises par **l'entête http** (le conteneur de variables globales) et sont de plus cachées.

2.2. Récupération par tableau

Chaque champ de formulaire en PHP est défini par un nom et une valeur. Dans un script, PHP va récupérer ces noms et ces valeurs dans des tableaux spéciaux dit **super globaux** (accessibles depuis partout). Pour la méthode GET, le tableau est **$_GET**, pour la méthode POST le tableau est **$_POST**. Si on ne souhaite pas se soucier de la méthode, on peut utiliser le tableau **$_REQUEST**. Par exemple :

```
<form action="foo.php" method="post">
Name: <input type="text" name="username"><br>
Email: <input type="text" name="email"><br>
<input type="submit" name="submit" value="Submit me!">
</form>
```

Dans la page PHP foo.php on aura :

```
<?php   echo $_POST['username'];
        echo $_REQUEST['email'];
?>
```

$_GET ne contiendra que les variables de type GET. $_POST ne contiendra que les variables de type POST. $_REQUEST contient les variables de type POST et GET mais aussi les variables de **cookies**.

2.3. Retour à l'XHTML : les formulaires

Nous présentons ici les différentes balises XHTML permettant la création d'un formulaire.

Balise form : Cette balise permet de définir un formulaire. Toute balise définissant un champ de saisie de formulaire doit être comprise entre des balises **<form>** et **</form>**. La balise form doit obligatoirement contenir l'attribut **action** qui indique le nom de la page PHP qui est appelée lorsque l'utilisateur soumet le formulaire.

Balise input : Cette balise auto-fermante permet de créer des champs de saisie de différents types. Elle doit obligatoirement contenir l'attribut **name** indiquant le nom du paramètre. De plus, elle possède l'attribut **type** qui permet de définir le type du champ. Voici les principales valeurs que peut prendre l'attribut **type** :
– text : indique que le champ est de type texte. La valeur du paramètre sera égale au texte saisi par l'utilisateur.
– password : joue le même rôle que text mais le texte saisi par l'utilisateur est caché. Ceci est utilisé lors de la saisie de mot de passe.
– hidden : indique que le champ ne sera pas affiché ("hidden" signifie "caché" en anglais). Ce type permet de transmettre des informations invisibles pour l'utilisateur.
– radio : indique que le champ est une case à cocher. Ce type de bouton est à utiliser si l'on souhaite que l'utilisateur **coche exactement un seul bouton** pour une question. C'est le cas pour la civilité : l'utilisateur doit cocher Homme ou Femme. Dans ce cas, on crée deux balises input de type radio ayant le **même nom** de paramètre (défini par l'attribut **name**). Il faut également spécifier pour chaque balise la valeur retournée si l'utilisateur coche cette case.
Ceci se fait grâce à l'attribut **value**. Par exemple, on peut avoir :

```
<input type="radio" name="civilite" value="h" /> Homme <br />
<input type="radio" name="civilite" value="f" /> Femme
```

Afin que le bouton soit cliquable même si l'on clique sur le texte à côté, il faut encadrer la balise input et le texte associé par les balises **<label>** et **</label>**.
– checkbox : indique que le champ est une case à cocher. Contrairement au bouton radio, la case checkbox permet de cocher autant de cases que l'on souhaite. Dans le cas où l'utilisateur peut **cocher plusieurs cases** (par exemple : quels langages de programmation utilisez-vous pour le Web ?), il existe deux possibilités de programmer ceci. À chaque case à cocher est associé un nom de paramètre (**name**) différent. De plus, aucune valeur (value) n'est définie.

– Balise select : Cette balise permet de créer un **menu déroulant**. Ceci est utile si l'utilisateur doit saisir une réponse parmi un ensemble prédéfini. C'est le cas par exemple lorsque l'utilisateur doit saisir le pays où il réside. L'utilisateur sélectionne alors le nom du pays parmi une liste donnée. Voici un exemple d'utilisation de la balise select :

```
< select name="pays">
<option value="en"> England </option>
<option value="fr"> France </option>
<option value="it"> Italia </option>
</ select >
```

– Balise textarea : Cette balise permet **la saisie d'un texte plus long** que la balise input de type text. Les attributs obligatoires sont : name correspondant au nom du paramètre (la valeur du paramètre est le texte saisi par l'utilisateur), **cols** et **rows** qui donnent respectivement le nombre de colonnes et de lignes utilisées pour l'affichage de la zone de saisie du texte.

3. Variable $_SESSION

Si l'on demande à un utilisateur de saisir son nom en arrivant sur le site, il peut être intéressant de **garder ce nom** pour toutes les pages du site visitées par l'utilisateur. Or, on ne peut pas

demander à l'utilisateur de saisir à chaque fois son nom dans un formulaire. De plus, par souci de simplicité et pour des raisons de sécurité évidentes (surtout si c'est un mot de passe !), il n'est pas pratique de transmettre ces informations directement dans l'url. On va alors utiliser la variable **$_SESSION**. Cette variable est une **variable globale** de type tableau, comme les variables $_GET et $_POST. Elle permet de stocker des informations durant le temps de visite de l'utilisateur.

Pour que la variable $_SESSION fonctionne, il faut impérativement ajouter au tout début (première ligne) des fichiers PHP où vous souhaitez utiliser cette variable l'instruction :

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

Cette instruction active les sessions en PHP. Concrètement, cela signifie que lorsqu'un utilisateur arrive sur cette page, une variable $_SESSION est créée sur le serveur pour cet utilisateur si elle n'existait pas déjà. De cette manière, il y a une variable $_SESSION par utilisateur. Cette variable s'utilise ensuite comme un tableau classique.

Les informations stockées dans la variable $_SESSION sont automatiquement détruites lorsque l'utilisateur n'accède plus au serveur pendant un certain temps. Cette durée dépend de la configuration du serveur. Elle est de l'ordre d'une demi-heure. Il est néanmoins possible de supprimer directement les informations à l'aide de l'instruction

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

Exemple

```
<?php //Code PHP exécuté lors de la soumission du formulaire
//On teste si l'utilisateur a correctement saisi son nom et son prénom
if( isset ($_POST['nom']) and isset($_POST['prenom'])
and trim($_POST['nom'])!="" and trim($_POST['prenom'])!="" )
{echo '<p> Connexion réussie </p>';
$_SESSION['nom'] = $_POST['nom'];
$_SESSION['prenom'] = $_POST['prenom'];
$_SESSION['connecte'] = true;}
```

Les informations stockées dans la variable $_SESSION sont relativement sûres. (Le risque zéro n'existe pas !) En effet, comme ces informations sont stockées sur le serveur, il est très difficile de pirater ces informations si le site est correctement configuré.

4. Utilisation des cookies

Il est possible de stocker des informations sur le client dans des fichiers appelés cookies. L'intérêt de ces fichiers est qu'ils peuvent être stockés autant de temps que nécessaire. C'est au moment de la création du cookie que l'on spécifie sa date d'expiration.

Comme il est possible de créer plusieurs cookies pour un même utilisateur, on stockera une unique information dans un cookie. La création d'un cookie se fait à l'aide de la fonction

```
<?php setcookie (nom du cookie, valeur du cookie, date d'expiration,
null, null, false, true); ?>
```

qui doit être appelée avant tout envoi de code XHTML, c'est-à-dire, avant tout code XHTML et avant toute instruction **echo**. Ainsi, si je souhaite créer un cookie contenant le pseudo d'un utilisateur, disons Kiti, et que je souhaite que ce cookie soit automatiquement supprimé au bout d'une semaine, je dois alors appeler la fonction **setcookie()** de la manière suivante :

```
<?php setcookie ('pseudo', 'Kiti', time() + 7 * 24 * 60 * 60, null,
null, false, true); ?>
```