SPRINT 1 : Se connecter et voir le menu

Lorsque vous devez insérer un formulaire dans *votre page HTML*, vous devez pour commencer écrire une balise <form> </form> . C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

```
<p>Texte avant le formulaire</p>
<form>
  <p>Texte à l'intérieur du formulaire</p>
</form>
<p>Texte après le formulaire</p>
```

On va donc maintenant compléter la balise <form> avec les deux attributs :
<form action="verification.php" method="POST">

- method="post" : c'est la méthode la plus utilisée pour les formulaires car elle permet d'envoyer un grand nombre d'informations. Les données saisies dans le formulaire ne transitent pas par la barre d'adresse ;
- Pour action , je vais taper le nom d'une page fictive en PHP (verification.php). C'est cette page qui sera appelée lorsque le visiteur cliquera sur le bouton d'envoi du formulaire.

Pour créer une zone de texte à une ligne, il faut donner un nom à votre zone de texte. Ce nom n'apparaît pas sur la page, mais il vous sera indispensable par la suite. En effet, cela vous permettra (en PHP par exemple) de reconnaître d'où viennent les informations : vous saurez que tel texte est le pseudo du visiteur, tel texte est le mot de passe qu'il a choisi, etc.
Pour donner un nom à un élément de formulaire, on utilise l'attribut name. Ici, on va supposer qu'on demande au visiteur de rentrer son username:
<input type="text" name="username">
Ici, on va supposer qu'on demande au visiteur de rentrer son mdp:
<input type="password" name="password">

Les libellés

Cette zone de texte est bien jolie mais si votre visiteur tombe dessus, il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise <label>:

```
<form method="post" action="traitement.php">
    <label>Votre username</label> : <input type="text" name="username" />
</form>
```

Ce code donne exactement le résultat que vous avez pu observer à la figure précédente.
Mais cela ne suffit pas. Il faut lier le label à la zone de texte.
Pour ce faire, on doit donner un nom à la zone de texte, non pas avec l'attribut name mais avec l'attribut id (que l'on peut utiliser sur toutes les balises).

Le name , lui, réfère à la variable du formulaire que l'élément concerné. Ici, il n'y a qu'un seul élément qui pourra référer à la variable username; name et id auront donc la même valeur. Mais lorsque nous utiliserons des checkbox ou des radio , plusieurs éléments correspondront à la même variable. Par exemple, la variable couleur avec un élément pour rouge, un pour bleu et un pour vert. Ils auront donc le même name , mais pas le même id . Pour lier le label au champ, il faut lui donner un attribut FOR qui a la même valeur que l' id du champ… Le mieux est de le voir sur un exemple :

```
<form method="post" action="traitement.php">
    <label for="username">Votre username</label>
    <input type="text" name="username" id="username" />
</form>
```

Zone de mot de passe

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une « zone de mot de passe », c'est-à-dire une zone où on ne voit pas à l'écran les caractères saisis. Pour créer ce type de zone de saisie, utilisez l'attribut type="password" .

```
<label for="password">Votre mot de passe :</label>
<input type="password" name="password" id="password" />
```

Rendre un champ obligatoire

Vous pouvez faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut required
<input type="text" name="username" id="username" required />

Le bouton d'envoi

Il ne nous reste plus qu'à créer le bouton d'envoi. Là encore, la balise <input /> vient à notre secours. Elle existe en quatre versions :

- type="submit" : le principal bouton d'envoi de formulaire. C'est celui que vous utiliserez le plus souvent. Le visiteur sera conduit à la page indiquée dans l'attribut

Pour créer un bouton d'envoi, on écrira donc par exemple :

```
<input type="submit" value="Envoyer" />
```

Dans la page mesFonctionsAccesBDD.php :

PARAMÉTRER LA CONNEXION

Pour établir la connexion, vous devez créer un DSN. Il s'agit en anglais du « Data Source Name ». Ce sont des paramètres permettant d'établir la connexion. Vous allez devoir renseigner plusieurs informations.

La première est le type de base de données avec lequel vous allez communiquer. Ici, ce sera MySQL. Vous allez ensuite devoir communiquer le nom de la base de données. Vous allez pouvoir ajouter des éléments comme le port de connexion.

Un autre paramètre qu'il est fortement conseillé d'ajouter est l'encodage. Par exemple l'UTF-8. Cela évitera des soucis d'affichage par la suite. Ci-dessous, voici un exemple de DSN.

```
$dsn =
'mysql:host=ADRESSE_DU_SERVEUR;dbname=VOTRE_BASE_DE_DONNEES;port=VOTRE_PORT;charset=VOTRE_ENCODAGE';
```

LA CONNEXION AU SERVEUR MYSQL

Maintenant, nous allons pouvoir créer la connexion au serveur MySQL. A la suite de cette dernière, nous allons effectuer une requête et afficher le résultat.

Dans un premier temps, il faut renseigner les paramètres permettant la connexion. Ceux-ci seront stockés dans une variable. Pour faire simple, on la nommera « pdo ». Voici un exemple :

```
$pdo = new PDO($dsn, 'VOTRE_NOM_UTILISATEUR_SERVEUR' , 'VOTRE_MOT_DE_PASSE');
```

Vous pouvez vérifier si des erreurs de relation entre votre script PHP et la base de données existent. Vous pouvez même en être informé par e-mail. Idéal si votre serveur de base de données ne fonctionne plus. Vous serez rapidement informé.

Si vous utilisez le script sur un serveur en ligne, il faut utiliser les blocs « try / catch » comme dans l'exemple ci-dessous.

```
try {
$pdo = new PDO($dsn, 'VOTRE_NOM_UTILISATEUR_SERVEUR' , 'VOTRE_MOT_DE_PASSE');
}
catch (PDOException $exception) {
 mail('VOTRE_EMAIL', 'PDOException', $exception->getMessage());
 exit('Erreur de connexion à la base de données');
}
return $pdo;
```

Si aucune erreur n'apparaît, nous sommes normalement connectés.

Ensuite la fonction testLoginMdp: Elle va servir à voir si le login et le mdp sont corrects.

On met en place la requête :

$pdoStatement = $pdo->prepare("SELECT count(*) as nbLogin FROM utilisateur WHERE login = :login and mdp= :password");

Les bindValue : Associe une valeur à un nom correspondant ou à un point d'interrogation (comme paramètre fictif) dans la requête SQL qui a été utilisée pour préparer la requête.

```
$bv1 = $pdoStatement->bindValue(':login', $login);
$bv2 = $pdoStatement->bindValue(':password', $mdp);
```

On l'exécute :
```
$execution = $pdoStatement->execute();
```

Et ducoup on regarde si il est trouvé dans la BDD ou pas :
```
if ($resultatRequete['nbLogin'] == 1) {
    $loginTrouve = true;
} else {
    $loginTrouve = false;
}
return $loginTrouve;
```

Ensuite dans la page indiquée par le FORM nous allons insérer du php pour vérifier la connexion:

Tout d'abord on appelle les fonctions :

include_once '../modeles/mesFonctionsAccesBDD.php';

Pour se connecter à la BDD :

$lePdo = connexionBDD(); ← C'est la fonction de connexion à la BDD

Pour récupérer la saisie de l'utilisateur : La méthode POST : Elle envoie un en-tête et un corps de message au serveur. Le corps est généralement constitué des données entrées dans le champ de formulaire par l'utilisateur.

```
$username=$_POST['username'];
$password=$_POST['password'];
```

Ici nous allons appeler la fonction pour voir si le login et le mdp s*ont corrects:*

$testLoginMdp = testLoginMdp($lePdo, $username, $password);

Ducoup si c'est correcte:

```
if ($testLoginMdp == true) {
```

On crée la session avec $_SESSION et le username et on l'envoie sur la page menu :

```
 $_SESSION['username'] = $username;
header('Location: menu.php');
}
```

Mais si c'est incorrecte on le renvoie sur le formulaire:

```
else {
    header('Location: formConnexion.php?erreur=2'); // utilisateur ou mot de passe vide
}
```

SPRINT 2 : Pouvoir visualiser le détail d'un bien

Pour visualiser le détail nous allons dans le href et nous mettons en place le lien pour se diriger vers unBien.php avec le bon Id correspondant exemple:

```
foreach ($lesbiens as $unbien) {
      $info = '<tr> <td class=\'reference\'>' . '<a href=' . '../autres/unBien.php?id_bien=' . $unbien['id_bien'] . '>' . $unbien['id_bien'] . '</a> </td> '
```

Pour ça nous allons créer les différentes fonctions :
Pour récupérer un bien :

```
   $pdoStatement = $pdo->prepare("SELECT * FROM bien WHERE id_bien = :reference");
   $bv1 = $pdoStatement->bindValue(':reference', $ref);
```

Pour récupérer l'image correspondante :

```
   $pdoStatement = $pdo->prepare("SELECT chemin, chemin2, chemin3 FROM images WHERE id_bien = :reference");
   $bv1 = $pdoStatement->bindValue(':reference', $ref);
```

Ensuite on va créer la page unBien.php qui va servir à visualiser le bien:
on appel la connexion, la fonction getunbien, getlesimages et on get l'id du bien

```
include_once '../modeles/mesFonctionsAccesBDD.php';
$pdo = connexionBDD();
$ref = $_GET['id_bien'];
$unbien = getunbien($pdo, $ref);
$uneimage = getlesimages($pdo, $ref);
```

Dans le body on va mettre le la balise <h2> avec comme info le type et la ville du bien correspondant grâce à: `<h2><?php echo ' ' . $unbien['type'] . ' ' . $unbien['ville']; ?></h2>`

ensuite on met en place le diaporama…

et par la suite pour afficher toutes les infos il y a plus qu'à faire les echo
exemple : `<p><?php echo ' ' . $unbien['description']; ?></p>`

SPRINT 3 : Pouvoir rechercher un bien sur 6 critères : ville, type, tranche de prix, présence d'un jardin, surface mini, nombre de pièces mini

En reprenant la fonction getrecherche() on ajoute juste la partie surface mini et nombre de pièce mini dans la requête en reprenant la même mise en forme exemple:

```
function getrecherche($pdo, $ville, $type, $jardin, $min, $max, $surfacemin, $nbpieces) {
    if ($surfacemin != "") {
        $requete .= " and surface >= :surfacemin";
    }

    if ($nbpieces != "") {
        $requete .= " and nbpieces >= :nbpieces";
    }
$pdoStatement = $pdo->prepare($requete);
    if ($surfacemin != "") {
        $bv6 = $pdoStatement->bindValue(':surfacemin', $surfacemin);
    }

    if ($nbpieces != "") {
        $bv7 = $pdoStatement->bindValue(':nbpieces', $nbpieces);
    }
```

Pour la partie HTLM c'est-à-dire la recherche on ajoute juste les labels et inputs:

```
<label for="surface" title="Veuillez choisir votre surface minimum  " class="oblig">Veuillez
choisir votre surface minimum  : </label>
<input type="text" value="" name="surfacemin" >

<label for="nbpieces" title="Veuillez choisir votre nbpieces minimum  " class="oblig">Veuillez
choisir votre nombre de pièces minimum  : </label>
<input type="text" value="" name="nbpiecesmin" >
```

SPRINT 4 : Supprimer un bien

On crée d'abord la partie formulaire pour supprimer :

```
<form method="post" action="">
        <label for="id_bien">Saisir l'id du bien à supprimer:</label>
        <input type="number" name="id_bien" id="id_bien" />
    <input class="submit" type="submit" value="Supprimer"  />
</form>
```

On appelle les différentes fonctions :

```
<?php
include_once '../modeles/mesFonctionsAccesBDD.php';
$pdo = connexionBDD();
if (isset($_POST['id_bien'])) {  // s'il existe …
    $idbien = $_POST['id_bien'];
    suppBien($pdo, $idbien); // supprimer le bien
}
?>
```

Dans la partie fonction on creer la fonction pour supprimer, c'est le même fonctionnement qu' ajouter ou modifier un bien c'est juste la requête qui change:

```
function suppBien($pdo, $idbien) {
    $pdoStatement = $pdo->prepare("DELETE FROM bien WHERE id_bien = :id_bien");
    $bv1 = $pdoStatement->bindValue(':id_bien', $idbien);
    $execution = $pdoStatement->execute();
    return $execution;
}
```