Introduction

PHP permet un interfaçage très simple avec un grand nombre de bases de données. Lorsqu'une base de données n'est pas directement supportée par **PHP**, il est possible d'utiliser un driver ODBC, pilote standard pour communiquer avec les bases de données.

La communication avec les bases de données se fait à l'aide de requêtes **SQL**, un langage de quatrième génération reconnu par l'ensemble des systèmes de gestion de bases de données (**SGBD**).

Dans les exemples ci-dessous, le système de gestion de bases de données utilisé est **MySQL**, un **SGBD** gratuit et rapide.

L'exploitation de **MySQL** avec **PHP** s'effectue généralement en 5 étapes :

1. Connexion à MySQL
2. Sélection de la base de données
3. Requête sur la base de données
4. Exploitation des résultats de la requête
5. Fermeture de la connexion à MySQL

I. Les solutions proposées par PHP

Pour pouvoir travailler avec la base de données en PHP, il faut d'abord s'y connecter.

Il va donc falloir que **PHP** s'authentifie : on dit qu'*il établit une connexion* avec **MySQL**. Une fois que la connexion sera établie, vous pourrez faire toutes les opérations que vous voudrez sur votre base de données !

PHP propose plusieurs moyens de se connecter à une base de données MySQL.

- L'extension ***mysql_*** : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Leur nom commence toujours par **mysql_**. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- L'extension ***mysqli_*** : ce sont des fonctions améliorées d'accès à MySQL. Elles proposent plus de fonctionnalités et sont plus à jour.
- L'extension **PDO** (***PHP Data Objects***) : c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.

II. Utilisation de PDO

1. Connexion

Pour ce connecter à MySQL avec PDO, nous allons avoir besoin de quatre renseignements :

- **le nom de l'hôte** : c'est l'adresse de la machine où **MySQL** est installé (comme une adresse IP). Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeur ***localhost*** (cela signifie « sur le même ordinateur »). Néanmoins, il est possible que votre hébergeur web vous indique une autre valeur à renseigner. Dans ce cas, il faudra modifier cette valeur lorsque vous enverrez votre site sur le Web ;
- **la base** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle test.
- **le login** : il permet de vous identifier. Renseignez-vous auprès de votre hébergeur pour le connaître. Le plus souvent (chez un hébergeur gratuit), c'est le même login que vous utilisez pour le FTP ;
- **le mot de passe** : il y a des chances pour que le mot de passe soit le même que celui que vous utilisez pour accéder au FTP. Renseignez-vous auprès de votre hébergeur.

> Pour l'instant, nous faisons des tests sur notre ordinateur à la maison. On dit qu'on travaille « **en local** ». Par conséquent, le nom de l'hôte sera ***localhost***.
>
> Quant au login et au mot de passe, par défaut le login est ***root*** et il n'y a pas de mot de passe.

Exemple :

Voici donc comment on doit faire pour se connecter à **MySQL** via **PDO** sur la base « ***projet*** »

```
<?php
    $bdd = new PDO('mysql:host=localhost;dbname=projet;charset=utf8', 'root', '');
?>
```

$bdd est un objet qui représente la connexion à la base de données. On crée la connexion en indiquant dans l'ordre dans les paramètres : le nom d'hôte (***localhost***), la base de données (***test***) et le login (***root***).

➢ *Tester la présence d'erreurs*

Si vous avez renseigné les bonnes informations (nom de l'hôte, de la base, le login et le mot de passe), rien ne devrait s'afficher à l'écran. Toutefois, s'il y a une erreur (vous vous êtes trompés de mot de passe ou de nom de base de données, par exemple), **PHP** risque d'afficher toute la ligne qui pose l'erreur, ce qui inclut le mot de passe.

En cas d'erreur, **PDO** renvoie ce qu'on appelle une exception qui permet de « capturer » l'erreur.

Exemple :

```
<?php
    try {
            $bdd = new PDO('mysql:host=localhost;dbname=projet;charset=utf8', 'root', '');
    } catch (Exception $e) {
            die('Erreur : ' . $e->getMessage());
    }
?>
```

Explication :

Sans trop rentrer dans le détail, il faut savoir que **PHP** essaie d'exécuter les instructions à l'intérieur du bloc « **try** ». S'il y a une erreur, il rentre dans le bloc « **catch** » et fait ce qu'on lui demande (ici, on arrête l'exécution de la page en affichant un message décrivant l'erreur).

Si au contraire tout se passe bien, PHP poursuit l'exécution du code et ne lit pas ce qu'il y a dans le bloc catch. Votre page PHP ne devrait donc rien afficher pour le moment.

2. Récupérer les données

Une fois connecté, il faut donc effectuer les requêtes. **PDO** supporte de nombreuses méthodes de soumission de requêtes mais commençons par la plus simple : **query().**

query() est un attribut de la classe **PDO** qui va prendre en argument la requête et qui va retourner un tableau.

Pour afficher le contenu d'une table, après s'être connecté à la base, il faudra d'abord sélectionner la table.

On demande à effectuer une requête sur la base de données comme suit :

```
$reponse = $bdd->query('Votre requête');
```

$bdd est l'objet PDO déjà créé

Votre requête est une requête SQL standard (SELECT, …)

$reponse contient maintenant la réponse de MySQL.

Exemple :

```
$reponse = $bdd->query('SELECT id,nom,email FROM personne');
```

➢ *Afficher le résultat d'une requête*

Le problème, c'est que ***$reponse*** contient quelque chose d'inexploitable. MySQL nous renvoie beaucoup d'informations qu'il faut organiser.

Pour récupérer une entrée, on prend la réponse de **MySQL** et on y exécute ***fetch(),*** ce qui nous renvoie la première ligne.

```
<?php
    $donnees = $reponse->fetch();
?>
```

$donnees est un tableau (array) qui contient champ par champ les valeurs de la première entrée

Il faut faire une boucle pour parcourir les entrées une à une. Chaque fois que vous appelez **$reponse->fetch(),** vous passez à l'entrée suivante. La boucle est donc répétée autant de fois qu'il y a d'entrées dans votre table.

Exemple :

```
$reponse = $bdd->query('SELECT nom FROM personne');
while ($donnees = $reponse->fetch())
{
   echo $donnees['nom'] . '<br />';
}
```

Cet exemple permet d'afficher le nom de chaque personne sur une ligne.

Quelle est la différence entre `$reponse` et `$donnees` ?

- **$reponse** contenait toute la réponse de MySQL en vrac, sous forme d'objet.
- **$donnees** est un array renvoyé par le ***fetch().***

Chaque fois qu'on fait une boucle, fetch va chercher dans **$reponse** l'entrée suivante et organise les champs dans l'array **$donnees.**

3. Fermeture de la connexion à MySQL

Lorsque la connexion à la base de données a réussi, une instance de la classe **PDO** est retournée à votre script. La connexion est active tant que l'objet **PDO** l'est. Pour clore la connexion, vous devez détruire l'objet en vous assurant que toutes ses références sont effacées. Vous pouvez faire cela en assignant ***NULL*** à la variable gérant l'objet. Si vous ne le faites pas explicitement, PHP fermera automatiquement la connexion lorsque le script arrivera à la fin.

Il existe un fonction ***closeCursor()*** qui est utilisé à tort pour fermer une connexion à une base de données.
La fonction ***closeCursor()*** (comme son nom l'indique) permet de libérer la connexion à la base de données permettant ainsi à d'autres requêtes SQL d'être exécutées.

Exemple :

```
$reponse->closeCursor();
```

4. Insérer des données

Cette fois, au lieu de faire appel à ***query()*** (que l'on utilisait pour récupérer des données), on va utiliser ***exec()*** qui est prévue pour exécuter des modifications sur la base de données.

Dans un site dynamique, il est intéressant de prévoir la possibilité d'insérer des données en ligne. Généralement on utilise un formulaire.

Soit un formulaire d'inscription :

```
<html>
  <head>
    <title>formulaire</title>
  <head>
  <body>
    <h2>Pour vous inscrire :</h2>
        <form method="post" action="insertion.php">
                <label>Nom : </label>
                <input type="text" name="nom"><br>
                <label>Email : </label>
                <input type="text" name="email"><br>
                <input type="submit" name="submit" value="Insérer">
        </form>
  </body>
  </html>
```


La page d'insertion **insertion.php** :

```
<?php
  try {
        $bdd = new PDO('mysql:host=localhost;dbname=projet;charset=utf8', 'root',
  '');
  } catch(Exception $e) { die('Erreur : '.$e->getMessage());  }
// On récupère les données du formulaire
        $nom = $_POST['nom'] ;
        $email = $_POST['email'] ;
// On ajoute une entrée dans la table personne
$bdd->exec("INSERT INTO personne (nom, email) VALUES ('$nom', '$email')");
  echo "La personne a bien été ajoutée ! " ;
?>
```

5. Modifier des données

La requête ***UPDATE*** vous permet de modifier les enregistrements d'un ou plusieurs champs dans votre table.

De la même manière que l'insertion , en PHP on fait appel à **exec()** pour effectuer des modifications.

Soit un formulaire d'inscription :

```
<html>
  <head>
    <title>formulaire</title>
  <head>
  <body>
    <h2>Pour modifier votre mail :</h2>
        <form method="post" action="update.php">
                <label>Nom : </label>
                <input type="text" name="nom"><br>
                <label>Email : </label>
                <input type="text" name="email"><br>
                <input type="submit" name="submit" value="Insérer">
        </form>
  </body>
  </html>
```

Le fichier ***update.php*** :

```
<?php
  try {
        $bdd = new PDO('mysql:host=localhost;dbname=projet;charset=utf8', 'root', '');
  } catch(Exception $e) {
        die('Erreur : '.$e->getMessage());
  }
// On récupère les données du formulaire
        $nom = $_POST['nom'] ;
        $email = $_POST['email'] ;
// On ajoute une entrée dans la table personne
$nb_modifs = $bdd->exec("UPDATE personne SET email='$email' WHERE nom='$nom' ");

  echo $nb_modifs . ' entrées ont été modifiées !';
  ?>
```

On peut utiliser est un variable qui permet de récupérer le nombre d'enregistrements modifiés (dans cet exemple c'est la variable **$nb_modifs).**

6. Supprimer un enregistrement

La requête ***DELETE*** permet de supprimer des enregistrements de la base de données. Il faut faire avec cette requête car il n'y a aucun moyen de récupérer les données.

Exemple :

```
$nb_supp = $bdd->exec("DELETE FROM personne WHERE nom='Abidi' ");
echo $nb_supp . ' entrées supprimées !'
```

Permet de supprimer tous les enregistrements de la table personne ayant pour nom 'Abidi'

La clause WHERE vous permet de choisir l'enregistrement que vous souhaitez supprimer, si vous n'utilisez pas cette clause, la table sera complètement vidée de son contenu.

Vous pouvez utiliser plusieurs instructions avec la clause **where**, par exemple.

```
$req = MySQL_query("DELETE FORM liste WHERE nom='$nom' AND email='$email' ");
```

III. Utilisation de MySQLi

L'extension MySQLi de PHP offre la possibilité de l'utiliser selon le paradigme de programmation procédurale ou celui de la programmation orientée objet. Dans cette partie on utilisera l'approche orientée objet.

1. Connexion

Pour ce connecter à MySQL avec l'extension MySQLi , nous allons avoir besoin des mêmes quatre renseignements utilisés avec PDO:

- **le nom de l'hôte**
- **la base**
- **le login**
- **le mot de passe**

Exemple :

Voici donc comment on doit faire pour se connecter à **MySQL** via **MySQLi** sur la base « ***projet*** »

```
<?php
      $bdd = new mysqli('localhost', 'root', '','projet');
?>
```

➢ *Tester la présence d'erreurs*

```
<?php
      $bdd = new mysqli('localhost', 'root', '','projet');
      if($bdd->connect_error) {
            die('Erreur : ' . $bdd-> connect_error);
      }
?>
```

2. Récupérer les données

Comme PDO, l'extension MySQLi offre des méthodes et des attributs prédéfinis permettant de récupérer des données à partir de la base MySQL.

Le principe est le même : on crée une requête SQL, on l'exécute et on exploite le résultat obtenu.

Exemple :

```
$reponse = $bdd->query('SELECT nom FROM personne');
if ($reponse-> num_rows > 0){
        while ($donnees = $reponse->fetch_assoc()){
            echo $donnees['nom'] . '<br />';
        }
} else {
            echo '0 résultats';
  }
```

3. Fermeture de la connexion à MySQL

La fermeture se fait en faisant appel à la méthodes prédéfinie **close()** de la classe **mysqli()**.

```
$bdd -> close();
```

4. Insérer des données

On reprendra ici le même formulaire de l'exemple de l'explication de l'insertion de données dans la base avec PDO.

Dans la page **insertion.php** le code sera le suivant :

```
<?php
    $bdd = new mysqli('localhost', 'root', '','projet');
    If ($bdd->connect_error) {
        die('Erreur : ' . $bdd-> connect_error);
    }
// On récupère les données du formulaire
        $nom = $_POST['nom'] ;
        $email = $_POST['email'] ;
// On créé la requête SQL
       $req = "INSERT INTO personne (nom, email) VALUES ('$nom', '$email')" ;
//On ajoute une entrée dans la table personne
       if ($bdd->query($req) === TRUE){
              echo "La personne a bien été ajoutée ! " ;
        } else {
              echo "Erreur : ".$bdd->error ;
        }
$bdd->close() ;
?>
```

5. Modifier des données

Pour la mise à jour des données dans la base de données avec MySQLi, on utilisera la même méthode query() de la classe mysqli() en passant comme paramètre à cette méthode la requête UPDATE du langage SQL.

On utilisera ici le même formulaire cité ci-dessus dans la partie de modification avec PDO.

```
<?php
    $bdd = new mysqli('localhost', 'root', '','projet');
    If ($bdd->connect_error) {
        die('Erreur : ' . $bdd-> connect_error);
    }
// On récupère les données du formulaire
        $nom = $_POST['nom'] ;
        $email = $_POST['email'] ;
// On créé la requête SQL
       $req = "UPDATE personne SET email='$email' WHERE nom='$nom' " ;
//On ajoute une entrée dans la table personne
```



```
        if ($bdd->query($req) === TRUE){
              echo "Modification effectuée avec succès ! " ;
        } else {
              echo "Erreur : ".$bdd->error ;
        }
$bdd->close() ;
?>
```

Annexe

La table ***personne***

personne (id, nom, email)

id	nom	email
1	Ouali	mohamed2010@gmail.com
2	Hafedh	hafedh77@lycos.com
3	Abidi	abidi@yahoo.fr
4	Houcem	houcem2015@gmail.com

Sources :
- https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/
- http://fr.flossmanuals.net/initiation-a-php/base-de-donnees-avec-pdo-php/
- http://php.net/docs.php