Connexion à une base de données en PHP

Qu'est ce qu'une base de données ?

Un base de données (BDD) est un ensemble d'information qui est organisé afin d'être facilement consulté, géré et mis à jour.
L'utilisation d'une BDD est souvent indispensable pour mettre en place un site web dynamique. Cela va permettre de stocker de données utile pour le site :

- liste des utilisateurs
- catalogue de produits
- trace des transactions effectuées

PHP prend en charge un grand nombre de bases de données d'une manière simple et efficace.

MYSQL et PHP

L'extension MySQLi permet d'accéder à une base de données. Nous allons maintenant nous connecter à notre base de données. Créer un fichier connexion.php et ajoutez le code suivant avec vos paramètres de connexion :

```php
<?php
    // Création de la connexion
    $connexion = new mysqli("localhost", "username", "password", "myDB");

    // Vérification de la connexion
    if (!$connexion->conect_error) {
        printf("Erreur : Connexion impossible : ". $connexion->connect_error);
        exit();
    }
    printf("Succès : Connexion réussite !")

?>
```


Effectuer des requêtes

```php
<?php
    $requete = "SELECT `id`, `prenom`, `nom` FROM `personnes`";
    $result = $connexion->query($requete);

    if (!$result) {
        printf("Lecture impossible");
    } else{
        // Lecture des résultats
        while($row = $result->fetch_array(MYSQLI_NUM)) {
            foreach($row as $data) {
                printf($data." ");
            }
            printf("<br>");
        }

        // Destruction de l'objet $result
        $result->free();
    }

    // Fermeture de la connexion
```

```
    $connexion->close();
?>
```

Effectuer des requêtes

La méthode mysqli_result la plus utile pour lire des données dans un tableau est *fetch_array()* dont la syntaxe est :

```
$result->fetch_array( int type );
```

Elle retourne un tableau qui peut être indicé (si la constante *type* vaut *MYSQLI_NUM*),
associatif (si la constante *type* vaut *MYSQLI_ASSOC*)

Les méthodes suivantes permettent également de récupérer une ligne de résultat à la fois :

```
$result->fetch_array( int type );
```

retourne un tableau associatif.

```
$result->fetch_array( int type );
```

retourne un tableau dont les indices de 0 à N sont les positions des attributs.

Effectuer des requêtes avec paramètres

Lorsque nous voulons faire une recherche dans notre base de données, nous utilisons la commande *WHERE* .

```php
<?php
    $requete = "SELECT `prenom`, `nom` FROM `personnes` WHERE `id` = $id";
    $result = $connexion->query($requete);

    if (!$result) {
        printf("Lecture impossible");
    } else{
        // Lecture des résultats
        foreach($row = $result->fetch_array(MYSQLI_NUM) as $data) {
            printf($data." ");
        }
        // Destruction de l'objet $result
        $result->free();
    }
    // Fermeture de la connexion
    $connexion->close();
?>
```


Effectuer des insertions

```php
<?php
    $requete = "INSERT INTO `personnes` (`prenom`, `nom`, `email`)
    VALUES ('John', 'Doe', 'j.doe@exemple.fr')";

    if (!$result = $connexion->query($requete);) {
        printf("Erreur : ".$connexion->error);
    } else{
        printf("Ajout réussi !")
?>
```

Requêtes préparées

Cela permet de créer des requêtes SQL qui ne sont pas directement utilisables mais qui contiennent des paramètres auquelson peut donner des valeurs différentes en fonction des besoins, pour des appels répétitifs.

```php
<?php
    $requete = "SELECT `prenom`, `nom` FROM `personnes` WHERE `id` = ?";
    $resultprep = $connexion->prepare($requete);
    $resultprep->bind_param("?", $id);
    $resultprep->execute();
    $resultprep->bind_result( $prenom, $nom);

    while ($resultprep->fetch()){
        printf("$prenom $nom");
    }
    $result->free();

    // Fermeture de la connexion
    $connexion->close();
?>
```

PDO et PHP

PDO (PHP Data Objectifs) est une extension orientée objet qui permet aussi de se connecter à une base de données depuis PHP. Elle n'est pas liée à MySQL comme *mysqli* .
Nous allons maintenant nous connecter à notre base de données.

```php
<?php
// Connexion
$host = "localhost"; // le chemin vers le serveur
```

```php
$port = "3306";
$dbname = "dbname"; // le nom de votre BDD
$username = "user"; // nom d'utilisateur pour se connecter
$password = "pass"; // mot de passe pour se connecter

try {
    $conn = new PDO('mysql:host='.$host.';port='.$port.';dbname='.$bdname, $username, $password);
}
catch(PDOException $except){
    printf("Echec de la connexion");
}

?>
```

Effectuer des requêtes SQL

Pour envoyer une requête au serveur, nous le choix entre plusieurs méthodes.
Pour celles qui ne retournent pas de résultats, il existe la méthode *exec()* des objets *PDO* dont la syntaxe est la suivante :

```php
integer $conn->exec( string $requete)
```

Elle est utilisée pour les requêtes *INSERT* , *UPDATE* ou *DELETE* .

Efectuer des requêtes SQL

Pour les requêtes qui vont retourner des résultats, il faut utiliser la méthode *query()* dont la syntaxe est la suivante :

```php
object $conn->query( string $requete)
```

Elle retourne FALSE en cas d'erreur, ou sinon, un objet de la classe PDOStatement représentant l'ensemble des lignes de résultats.

Exemple

```php
<?php
    $requete1 = "UPDATE  `personnes` SET `prenom` = 'Joe' WHERE `id` = 1";
    $nb = $conn->exec($requete1);
    printf("$nb ligne(s) modifiée(s)");

    $requete2 = "SELECT * FROM `personnes` ORDER BY nom";
    $result = $conn->queyr $requete2);

    if (!$result) {
        printf("Lecture impossible");
    } else {
        while ($row = $result->fetch(PDO::FETCH_NUM)) {
            foreach ($row as data) {
                printf($data." ");
            }
        }
        $result->closeCursor();
    }
    $conn = null;
?>
```

Les requêtes préparées

```php
<?php
    $requete = "SELECT `prenom`, `nom` FROM `personnes` WHERE `id` >= :id";
    $reqprep = $conn->prepare $requete);
    $reqprep->bindParam(':id', $id, PDO::PARAM_INT);
    $reqprep->execute();
    $reqprep->bindColumn('prenom', $prenom);
    $reqprep->bindColumn('nom', $nom);

    printf("Il y a". $reqprep->rowCount()." client(s) dont l'identifiant est supérieur ou égal à $id <br>");

    while ($reqprep->fetch PDO ::FETCH_BOUND)){
        printf("$prenom $nom <br>");
    }
    $reqprep->closeCursor();

    // Fermeture de la connexion
    $conn = null;
?>
```


MySQLi ou PDO

MySQLi :

- Spécialement conçu pour la base de données MySQL donc plus rapide
- Fonctionnalité avancées MySQL
- Manière procédurale ou en orientée objet

PDO :

- Utilisation sur tout types de BDD
- Utilisation simple et clair