Manipulation de bases de données avec PHP

Par Robert DIASSÉ

Introduction

Les bases de données sont des outils essentiels pour stocker, organiser et manipuler des informations. En PHP, nous pouvons interagir avec des bases de données MySQL pour effectuer des opérations comme insérer, lire, mettre à jour ou supprimer des données (appelé **CRUD**). Ce cours couvre :

1. La création de bases de données avec **phpMyAdmin** et via un code SQL.
2. Une explication du fonctionnement des bases de données.
3. Une introduction au **CRUD**.
4. Deux façons de manipuler une base avec PHP : **MySQLi** et **PDO**.

---

1. Création d'une base de données

1.1. Création via phpMyAdmin

phpMyAdmin est une interface graphique permettant de gérer facilement les bases de données MySQL.

1. **Accès à phpMyAdmin** :
   - Lancez votre serveur local (XAMPP, WAMP, etc.).
   - Accédez à phpMyAdmin via votre navigateur en tapant : http://localhost/phpmyadmin.
2. **Création d'une base de données** :
   - Cliquez sur l'onglet **Bases de données**.
   - Entrez le nom de la base (par exemple : `gestion_etudiants`).
   - Sélectionnez l'encodage **utf8mb4_general_ci**.
   - Cliquez sur **Créer**.
3. **Création d'une table** :
   - Dans la base `gestion_etudiants`, cliquez sur **Créer une table**.
   - Donnez un nom à la table (par exemple : `etudiants`) et précisez le nombre de colonnes (ex : 5).
   - Définissez les colonnes (exemple ci-dessous) :

- `id` (INT, clé primaire, auto-incrément).
- `nom` (VARCHAR(50)).
- `prenom` (VARCHAR(50)).
- `email` (VARCHAR(100), unique).
- `date_inscription` (TIMESTAMP, valeur par défaut : CURRENT_TIMESTAMP).

NB: Le type Varchar nécéssite à chaque fois de préciser la taille de caractère, Ex: 50; => `prenom` (VARCHAR(50))

1.2. Création via un script SQL

Si vous préférez coder, voici un script SQL équivalent pour créer la base et la table :

```sql
CREATE DATABASE gestion_etudiants;

USE gestion_etudiants;

CREATE TABLE etudiants (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    prenom VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    date_inscription TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

---

2. Fonctionnement d'une base de données

Une base de données est un système qui :

- **Stocke des informations** sous forme de **tables**.
- Permet d'accéder, manipuler et gérer ces informations via le **SQL** (Structured Query Language).

Concepts clés :

- **Table** : Une structure composée de colonnes (champs) et de lignes (données).
- **Colonnes** : Définissent les types de données (par exemple, `VARCHAR`, `INT`, `DATE`).
- **Lignes** : Contiennent les données proprement dites.
- **Clé primaire** : Une colonne unique qui identifie chaque ligne.
- **Requêtes SQL** : Langage pour interagir avec les données (ex : `SELECT`, `INSERT`, `UPDATE`, `DELETE`).

---

3. CRUD : Qu'est-ce que c'est ?

Le **CRUD** représente les quatre opérations principales qu'on peut effectuer sur une base de données :

1. **Create (Créer)** : Ajouter de nouvelles données.
2. **Read (Lire)** : Récupérer des données.

3. **Update (Mettre à jour)** : Modifier des données existantes.
4. **Delete (Supprimer)** : Effacer des données.

Ces opérations peuvent être réalisées en PHP grâce à deux interfaces : **MySQLi** (procédural) et **PDO** (orienté objet).

---

4. Manipuler une base avec PHP

4.1. Connexion avec MySQLi (procédural)

Connexion :

```php
<?php
$host = "localhost";
$user = "root";
$password = "";
$dbname = "gestion_etudiants";

// Connexion à MySQL
$conn = mysqli_connect($host, $user, $password, $dbname);

// Vérification de la connexion
if (!$conn) {
    die("Erreur de connexion : " . mysqli_connect_error());
}
echo "Connexion réussie avec MySQLi !";
?>
```

Insertion :

```php
<?php
$nom = "Dupont";
$prenom = "Jean";
$email = "jean.dupont@example.com";

$sql = "INSERT INTO etudiants (nom, prenom, email) VALUES ('$nom', '$prenom',
'$email')";

if (mysqli_query($conn, $sql)) {
    echo "Nouvel étudiant ajouté avec succès !";
} else {
    echo "Erreur : " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Récupération des données :

```php
<?php
$sql = "SELECT * FROM etudiants";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        echo "ID : " . $row['id'] . " - Nom : " . $row['nom'] . " " .
$row['prenom'] . " - Email : " . $row['email'] . "<br>";
    }
} else {
    echo "Aucun étudiant trouvé.";
}

mysqli_close($conn);
?>
```

4.2. Connexion avec PDO

Connexion :

```php
<?php
$host = "localhost";
$dbname = "gestion_etudiants";
$user = "root";
$password = "";

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $user, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connexion réussie avec PDO !";
} catch (PDOException $e) {
    die("Erreur de connexion : " . $e->getMessage());
}
?>
```

Insertion avec requête préparée :

```php
<?php
$nom = "Durand";
$prenom = "Marie";
$email = "marie.durand@example.com";

try {
    $sql = "INSERT INTO etudiants (nom, prenom, email) VALUES (:nom, :prenom,
:email)";
    $stmt = $pdo->prepare($sql);
```

```
    $stmt->bindParam(':nom', $nom);
    $stmt->bindParam(':prenom', $prenom);
    $stmt->bindParam(':email', $email);

    $stmt->execute();
    echo "Nouvel étudiant ajouté avec succès !";
} catch (PDOException $e) {
    echo "Erreur : " . $e->getMessage();
}
?>
```

Récupération des données :

```
<?php
try {
    $sql = "SELECT * FROM etudiants";
    $stmt = $pdo->query($sql);

    foreach ($stmt as $row) {
        echo "ID : " . $row['id'] . " - Nom : " . $row['nom'] . " " .
$row['prenom'] . " - Email : " . $row['email'] . "<br>";
    }
} catch (PDOException $e) {
    echo "Erreur : " . $e->getMessage();
}
?>
```

---

5. Comparaison entre MySQLi et PDO

Critères	MySQLi	PDO
Support des bases	MySQL uniquement	MySQL, PostgreSQL, SQLite, etc.
Mode	Procédural ou orienté objet	Orienté objet
Requêtes préparées	Oui	Oui
Flexibilité	Limité à MySQL	Support multi-SGBD
Simplicité	Plus facile pour débutants	Plus adapté pour les projets avancés

---

6. Exercices pratiques

Exercice 1 : Connexion

Créez un fichier `connexion.php` qui se connecte à une base de données MySQL en utilisant :

- MySQLi (procédural).
- PDO.

Exercice 2 : CRUD

1. Créez une page pour insérer un nouvel étudiant.
2. Créez une page pour afficher tous les étudiants dans un tableau HTML.
3. Ajoutez une fonctionnalité pour modifier les informations d'un étudiant.
4. Ajoutez une fonctionnalité pour supprimer un étudiant.

Exercice 3 : Comparaison pratique

- Modifiez les scripts CRUD pour passer de MySQLi à PDO ou inversement.

Conclusion

Ce cours permet de comprendre comment manipuler une base de données avec PHP via MySQLi et PDO. En maîtrisant ces deux approches, vous serez capable de choisir celle qui convient le mieux à vos projets.