Introduction Approfondie à PHP pour le Développement Web

Abdelweheb GUEDDES

14 février 2025

Séance 7 : Introduction aux Bases de Données avec PDO et PHP

Présentation des Bases de Données

Concept des Bases de Données Une base de données est un système structuré pour le stockage et la manipulation de données. Elle est essentielle pour les applications web dynamiques qui doivent gérer une grande quantité d'informations structurées et persistantes (comme les utilisateurs, les articles, les commentaires etc...). Dans ce cours, nous allons utiliser le SGBD MySQL avec l'extension PDO de PHP.

Le Rôle de MySQL MySQL est un système de gestion de bases de données (SGBD) open source, très populaire et souvent utilisé avec PHP. Il offre des fonctionnalités pour organiser, stocker et interroger efficacement les données.

Structure d'une Base de Données Une base de données contient des tables. Chaque table est composée de colonnes (ou champs) qui définissent le type de données et de lignes (ou enregistrements) qui contiennent les données.

- **Tables** : Contiennent les données d'un même type (ex : utilisateurs, articles, produits).
- **Colonnes** : Définissent les attributs ou types de données d'une table (ex : nom, prénom, date).

— **Lignes** : Contiennent les valeurs pour chaque colonne (par exemple : la ligne correspond à un utilisateur spécifique).

Création de la Base de Données

Choix de la méthode de création Vous pouvez créer une base de données MySQL de deux manières :

— En utilisant un outil tel que MySQL server directement.
— En utilisant des requêtes SQL directement dans PHP.

Nous allons vous présenter les 2 approches pour que vous puissiez choisir celle qui vous convient le mieux.

Méthode 1 : Création avec MySQL Server

La méthode la plus courante est de créer votre base de données en utilisant un outil tel que MySQL Server :

— Lancez votre serveur (WAMP ou un équivalent).
— Ouvrez phpMyAdmin, souvent accessible via `http://localhost/phpmyadmin`
— Créez une base de données nommée `coursphp`.
— Dans cette base de données, créez une table nommée `etudiants`, avec les colonnes suivantes :
  — `id` : type INTEGER, avec l'attribut clé primaire ('PRIMARY KEY') et auto-incrémenté ('AUTO_INCREMENT').
  — `nom` : type TEXT.
  — `prenom` : type TEXT.
  — `email` : type TEXT

Méthode 2 : Création avec des Requêtes SQL dans PHP

Si vous n'avez pas accès à phpMyAdmin, vous pouvez créer la base de données et les tables en utilisant des requêtes SQL en PHP :

```php
<?php
    $serveur = "localhost";
    $utilisateur = "root";
    $motDePasse = "";
    try{
    $connexion = new PDO("mysql:host=$serveur", $utilisateur,
    $motDePasse);
      $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::
   ERRMODE_EXCEPTION);
```

```
8       }
9       catch(PDOException $e)
10      {
11        die("La connexion a échoué : " . $e->getMessage());
12       }
13
14
15     $requeteCreationBase = "CREATE DATABASE coursphp";
16    try{
17      $connexion->exec($requeteCreationBase);
18        }
19    catch(PDOException $e){
20      die("La création de la base de données a échoué : " . $e
       ->getMessage());
21    }
22    try {
23     $connexion->exec("USE coursphp");
24      } catch (PDOException $e)
25      {
26          die("Impossible de sélectionner la base de données: "
       . $e->getMessage());
27      }
28
29      $requeteCreationTable = "CREATE TABLE etudiants (
30          id INT AUTO_INCREMENT PRIMARY KEY,
31          nom TEXT,
32          prenom TEXT,
33         email TEXT
34       )";
35      try{
36        $connexion->exec($requeteCreationTable);
37        }
38        catch(PDOException $e){
39            die("La création de la table a échoué : " . $e->
       getMessage());
40        }
41
42       echo "Base de données et table créées avec succès.";
43     $connexion = null;
44
45  ?>
```

Bien que cette méthode soit plus flexible, la première méthode est largement préferée pour des raisons pratiques.

Connexion à une Base de Données avec PDO

Introduction à PDO PDO (PHP Data Objects) est une extension de PHP qui fournit une interface abstraite pour accéder à différentes bases de données. Cela permet d'écrire du code qui est plus portable et plus sécurisé. Pour se connecter à une base de données avec PDO, nous allons utiliser la classe PDO et le bloc 'try catch' pour attraper les exceptions et les afficher (lorsque des erreurs se produisent).

```php
<?php
    $serveur = "localhost";
    $utilisateur = "root";
    $motDePasse = "";
    $baseDeDonnees = "coursphp";

    try {
        $connexion = new PDO("mysql:host=$serveur;dbname=$baseDeDonnees", $utilisateur, $motDePasse);
         $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        echo "<p>Connexion réussie à la base de données avec PDO !</p>";
    } catch (PDOException $e) {
        die("La connexion a échoué : " . $e->getMessage());
    }
?>
```

Exécution de Requêtes SQL avec PDO

Préparation de la requête avec `prepare` La fonction 'prepare()' permet de préparer une requête SQL en évitant les injections SQL. Elle prend en paramètre la requête SQL.

```php
<?php
$requete = "SELECT * FROM etudiants";
$statement = $connexion->prepare($requete);
?>
```

Exécution de la requête avec `execute` La fonction 'execute()' permet d'exécuter une requête préparée. Elle ne prend pas de paramètres.

```php
<?php
$statement->execute();
?>
```

Récupération des résultats avec `fetch` et `fetchAll`

— 'fetch()' : permet de récupérer une seule ligne de résultat. Elle retourne 'false' lorsque toutes les lignes ont été parcourues.

— 'fetchAll()' : permet de récupérer toutes les lignes de résultats sous forme d'un tableau associatif.

```php
<?php
    $serveur = "localhost";
    $utilisateur = "root";
    $motDePasse = "";
    $baseDeDonnees = "coursphp";
  try {
        $connexion = new PDO("mysql:host=$serveur;dbname=$baseDeDonnees", $utilisateur, $motDePasse);
        $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        die("La connexion a échoué : " . $e->getMessage());
    }
$requete = "SELECT * FROM etudiants";
   $statement = $connexion->prepare($requete);
     $statement->execute();
    echo "<ul>";
    while ($row = $statement->fetch(PDO::FETCH_ASSOC)) {
        echo "<li>Nom: " . $row["nom"] . ", Prenom: " . $row["prenom"] . ", Email: " . $row["email"] . "</li>";
    }
    echo "</ul>";
?>
```

Exercice Pratique Guidé (TP7)

Appliquez les connaissances acquises en utilisant PDO avec le TP suivant :

Objectif du TP

Créer une page PHP nommée `tp7.php` qui :

1. Se connecte à une base de données MySQL nommée "coursphp" en utilisant l'extension PDO et les paramètres de connexion appropriés.
2. Sélectionne toutes les lignes de la table nommée "etudiants" à l'aide d'une requête `SELECT`.

3. Affiche les données de la table (nom, prenom et email) dans une liste HTML (balises ‘<ul>‘ et ‘<li>‘), en utilisant une boucle ‘while‘ et la fonction ‘fetch()‘

Instructions

1. Créez un fichier `tp7.php` dans votre dossier web.
2. Connectez-vous à la base de données "coursphp" en utilisant PDO.
3. En cas d’échec de connexion, affichez le message d’erreur avec ‘getMessage()‘.
4. Préparez une requête SQL pour récupérer toutes les données de la table "etudiants" à l’aide de la fonction ‘prepare()‘ de PDO et en enregistrant la requête dans une variable ‘$statement‘.
5. Exécutez cette requête en utilisant la fonction ‘execute()‘ de l’objet statement.
6. Utilisez une boucle `while` et la fonction ‘fetch(PDO : :FETCH_ASSOC)‘ pour parcourir les résultats et récupérer chaque ligne sous forme de tableau associatif (PDO : :FETCH_ASSOC permet de recupérer les resultats sous forme de tableau associatif).
7. Utilisez la fonction ‘echo‘ et les balises HTML ‘<ul>‘ et ‘<li>‘ pour afficher les données de chaque étudiant.

Code Complet

```php
<?php
    $serveur = "localhost";
    $utilisateur = "root";
    $motDePasse = "";
    $baseDeDonnees = "coursphp";

    try {
        $connexion = new PDO("mysql:host=$serveur;dbname=$baseDeDonnees", $utilisateur, $motDePasse);
        $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        die("La connexion a échoué : " . $e->getMessage());
    }
   $requete = "SELECT * FROM etudiants";
```

```php
    try{
         $statement = $connexion->prepare($requete);
         $statement->execute();
    }catch (PDOException $e) {
        die("Erreur lors de la préparation ou de l'exécution de la requête : " . $e->getMessage());
    }
    echo "<ul>";
   while ($row = $statement->fetch(PDO::FETCH_ASSOC)) {
         echo "<li>Nom: " . $row["nom"] . ", Prenom: " . $row["prenom"] . ", Email: " . $row["email"] . "</li>";
   }
   echo "</ul>";
  $connexion=null;
?>
```

À Compléter par l'Étudiant

Modifiez le code précédent pour utiliser d'autres fonctionnalités de PDO :

1. Utilisez 'fetchAll()' au lieu de 'fetch' pour afficher les données.
2. Ajoutez une condition SQL pour n'afficher que les étudiants du groupe "Info1" (en utilisant la clause 'WHERE' dans la requête SQL).
3. Trier les résultats par ordre alphabétique sur le nom de famille, en utilisant la clause 'ORDER BY' et le nom du champ.
4. Limitez le nombre de résultats affichés à 2 en utilisant la clause 'LIMIT'.