PHP et les base de données

Dr N. BAME

Introduction

- De nombreuses applications PHP utilisent une base de données.
- Bien que la quasi-totalité des SGBD soit supportée, le plus couramment utilisé avec PHP est MySQL.
  - Des systèmes permettant d'installer automatiquement une plate-forme Apache/PHP/MySQL, tels que WAMPServer et EasyPHP.
  - MySQL est doté de nombreux avantages et ses fonctionnalités sont généralement tout à fait suffisantes et adaptées à la majorité des applications.

phpMyAdmin

- outil développé en PHP destiné à faciliter **la gestion d'un ensemble de bases de données MySQL** et cela à l'aide d'un simple navigateur.

✓ permet de gérer l'ensemble d'un serveur MySQL aussi bien qu'une simple base de données.

✓ permet d'opérer facilement les tâches d'administration courantes, de créer une structure de tables rapidement ou de tester les requêtes pendant le développement.

Interactions PHP/BD

- Il existe plusieurs possibilités pour utiliser une base de données avec PHP.

- On peut pour chaque type de SGBD, utiliser une extension native dédiée (mysqli pour MySQL, oci8 pour Oracle…).
  - ✓ Bien que ces extensions aient des similitudes entre elles, on aura alors à manipuler des fonctions spécifiques différentes selon le SGBD.

- L'autre solution récente avec PHP 5 est d'utiliser PDO.
  - ✓ extension qui permet de travailler de manière unifiée quel que soit votre SGBD.
  - ✓ méthode résolument tournée vers l'avenir, offrant beaucoup de souplesse et de puissance.

Connexion à la base de données

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

- Une fois que la connexion est établie, on peut faire toutes les opérations que l'on veut sur la base de données

- Il existe plusieurs moyens de se connecter à une base de données MySQL :
  - ✓ L'extension mysql: fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Leur nom commence toujours par mysql_.
  - ✓ L'extension mysqli: fonctions améliorées d'accès à MySQL. Elles proposent plus de fonctionnalités et sont plus à jour.
  - ✓ L'extension PDO : 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.

Approche classique PHP 4 : mysql

Il faut noter qu'en utilisant l'extension mysql classique on n'aura pas accès :

- ✓ aux fonctionnalités de MySQL 5 ;
- ✓ à la version objet ;
- ✓ aux requêtes préparées.

L'extension mysql (1)

- Connexion à la base de données

1. Connexion au serveur

La fonction **mysql_connect(),** permet de se connecter à un serveur MySQL et possède trois arguments principaux :

- ✓ l'adresse du serveur
- ✓ le nom d'utilisateur
- ✓ le mot de passe pour l'authentification

2. Choix de la base de données

La fonction **mysql_select_db(),** permet de choisir une base de données à utiliser sur un serveur MySQL. Elle prend deux arguments principaux :

- ✓ Le nom de la base de données à utiliser
- ✓ La ressource retournée par mysql_connect() : optionnel

```
<?php
    $link = mysql_connect('localhost', 'toto', 'ptiti');
    $db = mysql_select_db('scolarite', $link);
?>
```

L'extension mysql (2)

Requêtes

- On exécute une (seule) requête SQL avec la fonction **mysql_query().**
  - Cette fonction prend au moins un paramètre : une requête SQL sous forme de chaîne.
  - La fonction retourne FALSE en cas d'échec (colonne ou table invalide, droits insuffisants, pas de connexion, etc).
- **Remarque** : La requête SQL ne doit pas finir par un point-virgule.
- La requête peut être de n'importe quel type (sélection, mise à jour, destruction, etc.).
- Dans le cas d'un SELECT, SHOW, EXPLAIN ou DESCRIBE, mysql_query() retourne une ressource qui sera ensuite utilisée pour lire le résultat.

Exemple

```
<?php
    $link = mysql_connect('localhost', 'toto', 'ptiti');
    $db = mysql_select_db('scolarite', $link);

    $req = "SELECT nom, prenom FROM etudiant WHERE classe= 'L3GL'";
    $resultat = mysql_query ($req, $db);

?>
```

L'extension mysql (3)

Récupération du résultat

- La fonction **mysql_fetch_array()** permet de récupérer les enregistrements après l'exécution d'une sélection est
  - ✓ Elle prend en paramètre la ressource résultat (résultat de mysql_query).
  - ✓ Elle **retourne une ligne de résultat** sous forme d'un tableau associatif, d'un tableau indexé ou des deux.
    - Par défaut, le tableau retourné est à la fois associatif et indexé.
    - Dans un tableau associatif, l'index du tableau est le **nom du champ** correspondant à la **colonne**.
    - Dans un tableau indexé, les colonnes sont numérotées à partir de **zéro**.

- **Remarque :** mysql_fetch_array() ne retourne qu'une seule ligne de résultat.
  - ✓ Pour passer à la suivante, il faut exécuter la fonction à nouveau.
  - ✓ Elle retournera FALSE quand il n'y aura plus de lignes à lire.

Exemple

```php
<?php
    $link = mysql_connect('localhost', 'toto', 'ptiti' );
    $db = mysql_select_db('scolarite', $link);
    $req = "SELECT nom, prenom FROM etudiant WHERE classe= 'L3GL'";
    $resultat = mysql_query ($req );

    $ligne = mysql_fetch_array($resultat);
    while ($ligne) {
    // Affiche le champ prenom
    echo $ligne['prenom'], ' ';
    // Affiche le champ nom
    echo $ligne['nom'], '<br>';

    echo $ligne[0].', '. $ligne[1]; //Idem
    $ligne = mysql_fetch_array($resultat);
    }
?>
```

L'extension mysql (4)

- Si plusieurs colonnes portent le même nom, la dernière colonne sera prioritaire.
  - ✓ Dans une requête affichant des noms de colonnes identiques, le mieux est de les renommer

- Le second paramètre de mysql_fetch_array() peut être :
  - ✓ **MYSQL_ASSOC** : le résultat est uniquement un tableau associatif (index=nom de colonne)
  - ✓ **MYSQL_NUM** : le résultat est uniquement un tableau indexé numériquement.
  - ✓ **MYSQL_BOTH** (par défaut) : les deux

Il existe deux autres fonctions possibles :

- **mysql_fetch_row() :**
  - équivaut à mysql_fetch_array($resultat, MYSQL_NUM)
- **mysql_fetch_assoc() :**
  - équivaut à mysql_fetch_array($resultat, MYSQL_ASSOC)

L'extension mysql (5)

Insertion avec auto-incrément

- L'identifiant unique d'une table peut être un entier auto-incrémenté.

✓ L'avantage est qu'il n'y a pas besoin de gérer cet identifiant, c'est MySQL qui le détermine tout seul.

✓ Mais dans certains cas, il peut être nécessaire de récupérer la valeur de ce champ auto-incrémenté après une insertion pour, par exemple, mettre à jour un autre enregistrement (liste chaînée, jointure, etc).

✓ Pour cela, on utilise la fonction **mysql_insert_id().**

Exemple

```
$resultat=mysql_query("insert into etudiant (nom,    prenom) values (‘Diagne', Modou ')");
$id=mysql_insert_id();
$resultat=mysql_query("select nom, prenom from etudiant where mat=$id");
$ligne=mysql_fetch_array($resultat);
echo $ligne[‘prenom'].', '. $ligne[‘nom']; // Modou, Diagne
```

L'extension mysqli (1)

- Permet de profiter des fonctionnalités de MySQL 4.1 et +
- Disponible à partir de PHP 4.1.3
- PHP doit être compilé avec le support de l'extension mysqli (linux)
- L'extension mysqli doit être activée dans php.ini (windows)

L'extension mysqli A (1)

- Connexion

La fonction **mysqli_connect(),** possède 4 arguments principaux :

✓l'adresse du serveur

✓le nom d'utilisateur

✓le mot de passe pour l'authentification

✓la base de données à utiliser

```
<?php
    $link = mysqli_connect('localhost', 'toto', 'ptiti', 'scolarite');
    …
?>
```

L'extension mysqli A (2)

Requêtes

- La fonction **mysqli_query()** permet d'envoyer une requête au serveur MySQL
- Elle prend deux paramètres :
  - un identifiant de connexion vers le serveur
  - une requête SQL

```
<?php
    $bd = mysqli_connect('localhost', 'toto', 'ptiti', 'scolarite');

    $req = "SELECT nom, prenom FROM etudiant WHERE classe= 'L3GL'";
    $resultat = mysqli_query( $bd, $req );
?>
```

L'extension mysqli A (3)

Récupération du résultat

- Trois fonctions pour récupérer le résultat d'une requête :
  - **mysqli_fetch_assoc()** : Récupère le résultat sous forme de tableau associatif
  - **mysqli_fetch_row()** : Récupère le résultat sous forme de tableau indexé
  - **mysqli_fetch_object()** : Récupère le résultat sous forme d'objet

```
<?php
    $bd = mysqli_connect('localhost', 'toto', 'ptiti', 'scolarite');
    $req = "SELECT nom, prenom FROM etudiant WHERE classe = 'L3GL'";
    $resultat = mysqli_query( $bd, $req );

    $ligne = mysqli_fetch_assoc($resultat);
    while ($ligne ) {
    // Affiche le champ prenom
    echo $ligne ['prenom'], ' ';
    // Affiche le champ nom
    echo $ligne ['nom'], '<br>';
    $ligne = mysqli_fetch_assoc($resultat);
    }
?>
```

L'extension mysqli A (4)

Récupération du résultat

- Trois fonctions pour récupérer le résultat d'une requête :
  - **mysqli_fetch_assoc()** : Récupère le résultat sous forme de tableau associatif
  - **mysqli_fetch_row()** : Récupère le résultat sous forme de tableau indexé
  - **mysqli_fetch_object()** : Récupère le résultat sous forme d'objet

```
<?php
    $link = mysqli_connect('localhost', 'toto', 'ptiti', 'scolarite');
    $req = "SELECT nom, prenom FROM etudiant WHERE classe = 'L3GL'";
    $resultat = mysqli_query( $link, $req );

    while ($ligne = mysqli_fetch_row($resultat) ) {
    // Affiche le champ prenom
    echo $ligne [1], ' ';
    // Affiche le champ nom
    echo $ligne [0], '<br>';
    }
?>
```

L'extension mysqli A (5)

Récupération du résultat

- Trois fonctions pour récupérer le résultat d'une requête :
  - **mysqli_fetch_assoc()** : Récupère le résultat sous forme de tableau associatif
  - **mysqli_fetch_row()** : Récupère le résultat sous forme de tableau indexé
  - **mysqli_fetch_object()** : Récupère le résultat sous forme d'objet

```
<?php
    $link = mysqli_connect('localhost', 'toto', 'ptiti', 'scolarite');
    $req = "SELECT nom, prenom FROM etudiant WHERE classe = 'L3GL'";
    $resultat = mysqli_query( $link, $req );

    while ($ligne = mysqli_fetch_assoc($resultat)) {
    // Affiche le champ prenom
    echo $ligne ['prenom'], ' ';
    // Affiche le champ nom
    echo $ligne ['nom'], '<br>';
    }
?>
```

L'extension mysqli (8)

Fermeture de la connexion

- La fonction **mysqli_close()** permet de fermer la connexion.
- Elle prend en argument l'identifiant de connexion

```
<?php
    $link = mysqli_connect('localhost', 'toto', 'ptiti', 'scolarite');
    $req = "SELECT nom, prenom FROM etudiant WHERE classe = 'L3GL'";
    $resultat = mysqli_query( $link, $req );
    $ligne = mysqli_fetch_assoc($resultat);
    while ($ligne ) {
    // Affiche le champ prenom
    echo $ligne ['prenom'], ' ';
    // Affiche le champ nom
    echo $ligne ['nom'], '<br>';
    }

    //ferme la connexion
    mysqli_close($link);
?>
```

Mysqli : accès objet (1)

1. **Initialisation de la connexion**

```
$db = new mysqli('localhost','Moussa','mnd94','Scolarite');
```

2. **Envoi d'une requête au serveur MySQL**

```
$resultat = $db->query("SELECT * FROM Etudiants");
```

3. Si la requête est un SELECT, lecture ligne par ligne du résultat reçu

```
while ($ligne = $resultat->fetch_assoc() ) {…}
```

4. La fermeture de la connexion est automatiqu en fin de page. Sinon affecter la valeur NULL à la ressource de connexion

```
$db = NULL;
```

L'extension PDO (1)

PDO = PHP Data Object

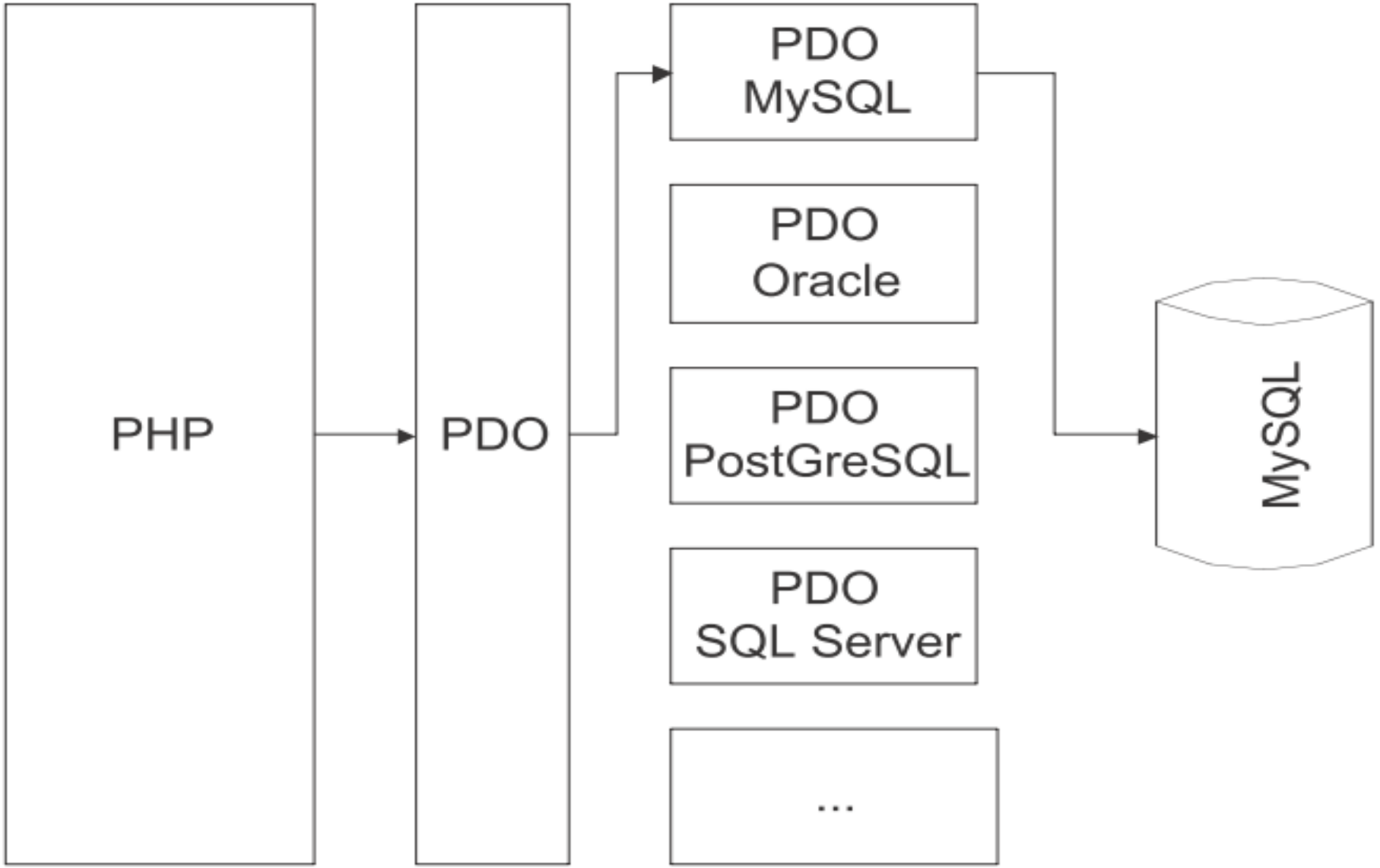
- apporte un confort d'utilisation et une abstraction plus importants que les anciennes fonctions natives propres à chaque SGBD.

- Le principal avantage de PDO est qu'on peut l'utiliser de la même manière pour se connecter à n'importe quel autre type de base de données (PostgreSQL, Oracle…)

- L'approche objet de PDO permet de plus d'étendre les fonctions d'accès à la base facilement et de manière transparente.

- PDO fonctionne avec un ensemble d'extensions
  - Une extension PDO de base qui définit l'interface commune.
  - Une extension Driver PDO relative à chaque SGBD que l'on veut gérer.

L'extension PDO (2)



L'extension PDO (3)

Activation

- Généralement, PDO est activé par défaut pour MySQL.
  - ✓ Pour l'activer ; dans le fichier de configuration de PHP **php.ini**, dé-commenter (enlever le point virgule devant) la ligne d'extension pdo du SGBD

```
;extension=php_pdo_firebird.dll
extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_odbc.dll
extension=php_pdo_pgsql.dll
;extension=php_pdo_sqlite.dll
```

L'extension PDO (3)

- Connexion

Pour se connecter à une base de données, on crée un nouveau objet PDO avec quatre éléments :

- ✓ l'adresse du serveur : host
- ✓ le nom d'utilisateur : username
- ✓ le mot de passe pour l'authentification : password
- ✓ la base de données à utiliser : dbname

```
<?php
    $bdd = new PDO('mysql:host= localhost;dbname=databaseName',
    'username', 'userPassword');
?>
```

- ✓ Le premier paramètre (commençant par mysql)) qu'on appelle DSN (Data Source Name) permet d'identifier le serveur MySQL et la base de données.

```
$dsn = 'mysql:host=localhost;dbname=scolarite';
```

L'extension PDO (4)

- Connexion

```
<?php
$dsn  =  'mysql:host=localhost;dbname=scolarite';
$bdd = new PDO($dsn, 'toto ', 'ptiti');
?>
```

– Le DSN peut disposer de données facultatives comme le port ou le socket (indiquant l'adresse de la socket unix pour la connexion locale) :

```
$port = '3307';
$socket = '/tmp/mysql.sock';
$dsn = "mysql:host=$hote;port=$port;dbname=$base";
$dsn2 = "mysql:unix_socket=$socket;dbname=$base";
```

L'extension PDO (5)

- Requêtes de manipulation : méthode **exec**

```
$sql="insert into etudiant values(12,'Diop','Moussa')";
$res=$bdd->exec($sql);
```

- Requêtes d'interrogation : méthode **query**

```
$sql="select * from etudiant" ;
$resultat=$bdd->query($sql);
```

- Récupération du résultat : méthode **fetch**

```
while  ($ligne= $resultat ->fetch())  {
echo  $ligne['NE'].' | '.$ligne['nom'].' | '. $ligne['prenom'] ;
}
```

L'extension PDO (8)

- Fermeture de la connexion
  - Affecter la valeur NULL à la ressource de connexion

```
$dsn  =  'mysql:host=localhost;dbname=scolarite';
$bdd = new PDO($dsn, 'toto ', 'ptiti');

$sql="select * from etudiant" ;
$resultat=$bdd->query($sql);
while  ($ligne= $resultat ->fetch())  {
echo  $ligne['NE'].' | '.$ligne['nom'].' | '. $ligne['prenom'] ;
}
//Fermeture de la connexion
$bdd=NULL;
```

L'extension PDO (8)

- Tester la présence d'erreurs
  - S’il y a une erreur (on est trompé 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
  - Pour éviter que les visiteurs puissent voir le mot de passe si une erreur survient lorsque le site est en ligne. Il est préférable de traiter l'erreur.
  - En cas d'erreur, PDO renvoie ce qu'on appelle une exception qui permet de « capturer » l'erreur.

Tester la présence d'erreurs

```
try
{
    $bdd = new PDO('mysql:host=
    localhost;dbname=databaseName', 'username', ‘pwd');
}
catch (Exception $e)
{
    die(‘Erreur : ‘.$e->getMessage());
}
```

Traquer les erreurs

- Pour afficher des détails sur l'erreur, il faut activer les erreurs lors de la connexion à la base de données via PDO.

- Lors de la création de l'objet de connexion, activer un paramètre à la fin pour activer les erreurs :

```
<?php
$bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
array(PDO::ATTR_ERRMODE =>
PDO::ERRMODE_EXCEPTION));
?>
```