

Les moyens  
Les étapes  
PDO

COMMUNIQUER AVEC UNE BD

UE WEBAPP (G) : Ingénierie des applications web

51

  
IMT Atlantique  
Brest - Nantes - La Rochelle

Communiquer avec une BD ► les moyens

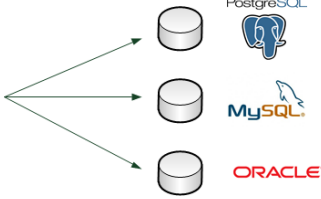
Plusieurs solutions

- Extension mysql\_ : accéder à et communiquer avec une BD MySQL → ancienne; ne plus utiliser
- Extension mysqli\_ : fonctions améliorées et enrichies d'accès à une BD MySQL → récente et à jour
- Extension PDO : classes permettant d'accéder à n'importe quel type de BD → récente et recommandée

Que choisir ?


- PDO car
  - Indépendance du code php vis-à-vis des SGBD
  - Croissance de son utilisation par la communauté

PDO

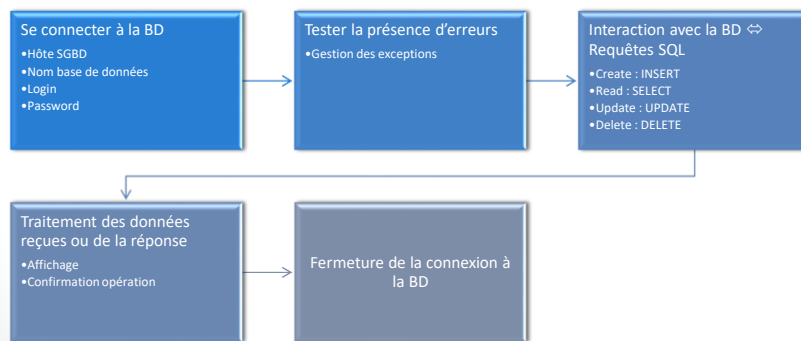


UE WEBAPP (G) : Ingénierie des applications web

52

  
IMT Atlantique  
Brest - Nantes - La Rochelle

## Communiquer avec une BD ► les étapes



## Communiquer avec une BD ► PDO

- Les méthodes à utiliser en fonction des requêtes SQL sont :
  - **exec()** pour INSERT, UPDATE, DELETE
    - Retourne le nombre de lignes modifiées mais ne retourne pas de résultat pour une requête SELECT
  - **query()** pour SELECT, EXPLAIN, SHOW, DESC utilisé une seule fois dans le programme
    - Retourne une instance de la classe PDOStatement
    - Après l'exécution de la méthode query() les données sont mises en mémoire mais pas affichées
    - Il faut utiliser un mécanisme spécifique pour récupérer les données.
  - **prepare()** et **execute()** pour une requête utilisée plusieurs fois dans le programme
- Remarque :
  - Les méthodes exec() et query() ne font qu'envoyer un ordre à la base de données

## Communiquer avec une BD ▶ PDO

### ▶ la classe *PDOStatement*

- offre deux méthodes pour traiter les données :
  - `fetchAll()` : `PDOStatement::fetchAll( [fetch_style])`
    - Retourne les données sous la forme d'un tableau PHP.
  - `fetch()` : `PDOStatement::fetch( [fetch_style [ , cursor_orientation [ , cursor_offset]])`
    - Permet une lecture séquentielle du résultat.
    - À utiliser pour de très gros volumes de données.
- Les deux méthodes acceptent un paramètre facultatif le `fetchStyle` :
  - `FETCH_ASSOC` : retourne un tableau associatif indexé par les nom de colonnes,
  - `FETCH_BOTH` (par défaut) : retourne un tableau indexé par les noms de colonnes et par les numéros de colonnes (indice de départ 0),
  - `FETCH_OBJ` : retourne un objet dont les propriétés sont les noms des colonnes.

```

1. <?php
2. //configure data to connect to database
3. $server="localhost";
4. $admin='phpUser';
5. $password='phpUser_pwd.iaw';
6. $bd="demo";
7. try{
8. //connect to database
9. $pdo=new PDO("mysql:host=$server;dbname=$bd", $admin,
    $password, array(PDO::ATTR_ERRMODE =>
    PDO::ERRMODE_EXCEPTION));
10. }
11. // Test Error
12. catch(PDOException $e){
13.     die("Error : ".$e->getMessage());
14. }
15. ?>

```

### Communiquer avec une BD ▶ PDO ▶ Connexion et contrôle

Fichier connexion.php

```

1. <?php
2.     include("connexion.php");
3.     $sql="SELECT * FROM `person`";
4.     //Interact with database
5.     $pdoStatement=$pdo->query($sql);
6.     $result = $pdoStatement->fetchAll(PDO::FETCH_ASSOC);
7.     print_r($result);
8.     //other solution
9.     echo "<br><br>";
10.    $pdoStatement = $pdo->query($sql);
11.    while ($row = $pdoStatement->fetch())
12.        echo "<br>\nFirstName = ". $row['first_name'], "<br>" .
13.            " LastName = " . $row['last_name'], "<br>" .
14.            " Mail = " . $row['mail'], "<br>";
15.    "<br>";
16.    if ($pdo) {
17.        $pdo = NULL; // close the connexion
18.    }
19.    ?>

```

### Communiquer avec une BD ▶ PDO ▶ Lecture tableau

Les données sont retournées dans un tableau associatif en mémoire

```

1. <?php
2.     include("connexion.php");
3.     $sql="SELECT * FROM `person`";
4.     $pdoStatement=$pdo->query($sql);
5.     $result = $pdoStatement->fetchAll(PDO::FETCH_OBJ); //
return array of anonymous object instance of stdClass with
column names as properties
6.     foreach ($result as $key => $value) {
7.         echo $key."<br>";
8.         print_r($value);
9.         echo "<br/>";
10.    }
11.    ?>

```

### Communiquer avec une BD ▶ PDO ▶ Lecture objet anonyme

Les données sont retournées dans des objets instances de la classe stdClass

```
1. <?php
2.     include("connexion.php");
3.     include("Person.class.php");
4.     $sql="SELECT * FROM `person`";
5.     $pdoStatement=$pdo->query($sql);
6.     $pdoStatement->setFetchMode(PDO::FETCH_CLASS, 'Person');
7.     while ($person = $pdoStatement->fetch())
8.         echo $person;
9. ?>
```

Fichier Person.class.php


```
1. <?php
2. class Person{
3.     public function __toString() :String
4.     {
5.         return "<br/>FirstName =". $this->first_name."<br/>". "LastName =". $this-
6.         >last_name."<br/>". "Mail =". $this->mail."<br/>";
7.     }
8. }
```

**Communiquer avec une BD ▶ PDO ▶ Lecture objet spécifique**

Les données sont retournées dans des instances de la classe spécifiée dans le code.  
Les attributs sont ajoutées à la classe et ensuite le constructeur de la classe est appelée

UE WEBAPP (G) : Ingénierie des applications web

59




```
1. <?php
2.     include("connexion.php");
3.     $sql="SELECT * FROM `person` where `mail` LIKE :domain";
4.     //Interact with database
5.     $pdoStatement=$pdo->prepare($sql);
6.     $pdoStatement->execute(array(':domain'=>"%@imt-atlantique.fr"));
7.     $result = $pdoStatement->fetchAll(PDO::FETCH_ASSOC);
8.     print_r($result);
9.     if ($pdo) {
10.         $pdo = NULL ; // close the connexion
11.     }
12. ?>
```

**Communiquer avec une BD ▶ PDO ▶ Lecture avec paramètres**

UE WEBAPP (G) : Ingénierie des applications web

60



```

1. $firstName="Issam";
2. $lastName="REBAI";
3. $mail="issam.rebai@imt-atlantique.fr";
4. $sql="INSERT INTO person(`lastName`,`firstName`,`mail`)
   VALUES ('$lastName','$firstName','$mail')";
5. //Interact with database
6. $nbResult=$pdo->exec($sql);
7. echo "Result of request :". $nbResult;
8. if ($pdo) {
9.     $pdo = NULL ; // close the connexion
10. }
11. ?>

```

### Communiquer avec une BD ▶ PDO ▶ Insertion

Requête construite par concaténation → Risque d'injection de code SQL

```

<?php
//....
$firstName=$_GET["firstName"];
$lastName=$_GET["lastName"];
$mail=$_GET["mail"];
$sql="INSERT INTO person(`lastName`,`firstName`,`mail`) VALUES
(?,?,?)";
$req= $pdo->prepare($sql);
//Interact with database
$nbResult=$req->execute(array($lastName, $firstName,$mail));
echo "Result of request :". $nbResult;
if ($pdo) {
    $pdo = NULL ; // close the connexion
}
?>

```

### Communiquer avec une BD ▶ PDO ▶ Insertion

Requête préparée avec marqueur ? → protection contre l'injection de code SQL

```
<?php
//....
$sql="INSERT INTO person(`lastName`,`firstName`,`mail`) VALUES
(:ln,:fn,:mail)";
$req= $pdo->prepare($sql);
//Interact with database
$nbResult=$req->execute(array('ln'=>$lastName,
'fn'=>$firstName,'mail'=>$mail));
?>
```

**Communiquer avec une BD ▶ PDO ▶ Insertion**

Requête préparée avec marqueurs nominatifs → [protection contre l'injection de code SQL](#)

UE WEBAPP (G) : Ingénierie des applications web

63




# Injection SQL (1)

**Communiquer avec une BD ▶ PDO ▶ Injection SQL**


UE WEBAPP (G) : Ingénierie des applications web

64



# Injection SQL (2/2)


Communiquer avec une BD ▶ PDO ▶ Injection SQL

UE WEBAPP (G) : Ingénierie des applications web  
65  


## Communiquer avec une BD ▶ PDO

### ▶ Les transactions

- Définition :
  - Une transaction est une unité de traitement faisant passer une BD d'un état cohérent à un autre état cohérent.
  - Une transaction regroupe une ou plusieurs opérations unitaires qui forment une action métier logique.
- Exemple → Opération bancaire
- Pour l'utilisateur, une transaction est un ensemble de mises à jour qui maintient la cohérence de la base, quel que soit le contexte d'exécution de ces mises à jours :
  - Panne (réseau, système, matériel, etc.)
  - Accès concurrent aux données
  - Contraintes d'intégrité
  - Droit d'accès
  - Distribution des données

UE WEBAPP (G) : Ingénierie des applications web  
66  




## Communiquer avec une BD ▶ PDO


### ▶ Les transactions

Une transaction offre 4 fonctionnalités majeurs, qui la caractérisent, connues sous l'acronyme ACID. Le SGBD assure que chaque transaction reste ACID.

A ▶ Atomicité	Une transaction est exécutée totalement ou pas du tout
C ▶ Cohérence	La base de donnée résultant de l'exécution de la transaction est cohérente par rapport aux contraintes d'intégrité sur les données
I ▶ Isolation	La transaction n'est pas altérée par l'exécution d'autres transactions par d'autres utilisateurs
D ▶ Durabilité	Une fois la transaction validée, son effet est permanent

UE WEBAPP (G) : Ingénierie des applications web

67



## Communiquer avec une BD ▶ PDO

### ▶ Les transactions


- Validation automatique `autocommit` (comportement par défaut)
  - PDO gère une transaction par instruction SQL.
    - Si l'instruction ne provoque pas d'erreurs, la transaction est validée
    - Sinon (instruction échoue) la transaction est annulée
- Comment faire pour valider plusieurs opérations sur une BD en une seule fois (une action logique)?
  1. Démarrer une transaction (désactive `autocommit`)
    - `PDO::beginTransaction(): bool` `$pdo->beginTransaction();`
  2. Réaliser les opérations SQL unitaires
    - Opérations CRUD `$pdo-> exec ("...") ;`
  3. Valider la transaction si toutes les opérations réussissent
    - `PDO::commit(): bool` `$pdo->commit();`

**Sinon Annuler la transaction**

  - `PDO::rollBack(): bool` `$pdo->rollBack();`

UE WEBAPP (G) : Ingénierie des applications web

68



Exercices de l'Activité A4 sur l'espace Moodle

**PAUSE ACTIVITÉ PRATIQUE**

UE WEBAPP (G) : Ingénierie des applications web

[ 69 ]

