Introduction

PDO est une interface d'accès aux bases de données

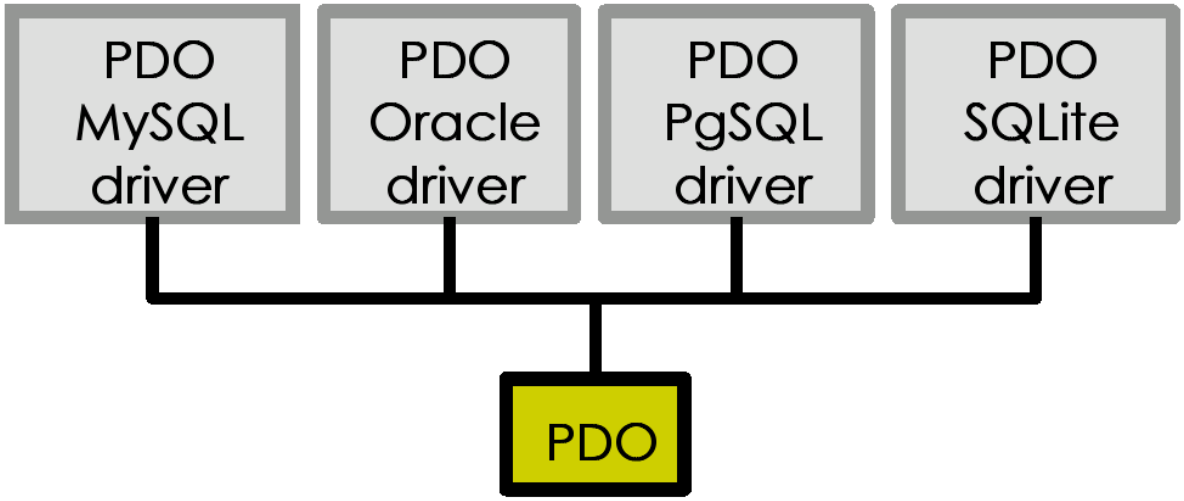
PDO (*PHP Data Objects*) est une interface pour accéder à une base de données depuis PHP. Elle gère la connexion, l'envoie des requêtes, la déconnexion à la base de données. Elle permet de changer plus facilement de système de gestion de bases de données.

PDO n'est pas un système d'abstraction complet

- Il ne manipule pas les requêtes.
- Il met à disposition des classes et des fonctions communes à un ensemble de SGBD.
- Si certains SGBD ne disposent pas de certaines fonctionnalités (gestion des transactions, requêtes paramétrées, etc.), il les simule.

Architecture et principe

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.



Remarque : il existe des pilotes pour MS-SQL/PHP dont celui de MS (recommandé)

Installation et utilisation

Pour utiliser PDO et mysql il suffit d'activer les extensions suivantes (fichier php.ini):
- php_pdo.dll
- php_pdo_mysql.dll

Pour utiliser PDO et MS-SQL installer "Microsoft Drivers 3.0 for PHP for SQL Server" ET activer l'extension php_pdo_sqlsrv_54_nts.dll

Les principales classes disponibles avec l'extension PDO sont les suivantes :

- La classe PDO gère l'accès aux SGBD ainsi que les fonctionnalités de base :
  - Connexions
  - Lancement des requêtes
- La classe PDOStatement gère une liste de résultats.
- La classe PDOException est une classe d'exception personnalisée interne pour PDO.

Connexion au serveur de données

Premier exemple : Connexion simple à PDO

Déclaration du DSN (Data Source Name), connexion et requête

```
<?php
// DSN pour se connecter à MySQL
$dsn = 'mysql:host=localhost ; dbname=mabase';

// DSN pour se connecter à MS Sql Server
// $dsn = 'sqlsrv:server=(local)\S2008R2 ; database=mabase';

// Création d'un objet pour manipuler des requêtes
$dbh = new PDO($dsn, 'login', 'pass');

// Execution d'une requête SELECT
$result = $dbh->query('SELECT * from FOO');

// Itération sur les résultats d'une requête
foreach ($result as $row) {
  print_r($row);
}
```

Remarque : Pour changer de base de données ou de SGBD, il suffit donc de changer le DSN.

Connexion au serveur de données (suite)

Second exemple : Connexion simple à PDO

La première étape consiste à déclarer les variables qui vont permettre la connexion à la base de données (ce sont les paramètres des fonctions de connexion à la base).
Ces variables sont :

- $user : le nom d'utilisateur
- $passwd : le mot de passe
- $host : l'hôte (machine sur laquelle le SGBD est installé)
- $bdd : le nom de la base de données

La deuxième étape est de construire le DSN (Data Source Name)

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

Enfin, la connexion se fait par la création d'un objet de la classe PDO

```
$dbh= new PDO($dsn, $user, $pass);
```

Il ne faut pas oublier de fermer la base $dbh=null;

Le fichier de connexion connexion-PDO.php

```
<?php
$user = 'root';
$pass = 'mysql';
// Data Source Name
$dsn = 'mysql:host=localhost;dbname=mabase';
try{
//Tentative de connexion : on crée un objet de la classe PDO
  $dbh= new PDO($dsn, $user, $pass);
//S'il y a des erreurs de connexion, un objet PDOException
//est lancé. On peut gérer l'exception si on le souhaite
}
catch (PDOException $e){
  print "Erreur ! :" . $e->getMessage() . "<br/>";
  die(); // Quitter le script
}
?>
```

Ouverture de la base et creation d'une base: createDataBase()

```
<?
$dbusername = "root" ;
$dbpassword = "root" ;
$pdo = new PDO("mysql:host=localhost", $dbusername,
$dbpassword);
$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

$dbname = "`"."base_test"."`";
$pdo->query("CREATE DATABASE IF NOT EXISTS $dbname");
$pdo->query("use $dbname");
?>
```

Faire des requêtes: exec() ou query()

Une fois l'instance de PDO construite, pour envoyer une requête au serveur, on peut utiliser deux méthodes de la classe PDO :

- query() pour extraire des données de la base, pour la sélection, pour la gestion des tables
  - PDO::query exécute une requête SQL et retourne un jeu de résultats
- exec() pour modifier la base de données, pour les mises à jour
  - PDO::exec exécute une requête SQL et retourne le nombre de lignes affectées
  - PDO::exec() ne retourne pas de résultat pour une requête SELECT.

Exemples :

```
<?
$db->exec("CREATE TABLE IF NOT EXISTS users (".
" pseudo char(20),". " password char(50)". ");");
?>

<?
$res = $db->exec('UPDATE users SET password="'.
md5($password).'" WHERE pseudo="'.$pseudo.'";');
echo "nombre de lignes modifiees = $res";
?>

<?
$res = $db->query("select pseudo from users;");
?>
```

Exemple d'exécution d'une requête DELETE

```
<?php
$dbh = new PDO('odbc:sample', 'root', 'root');
/* Effacement de certaines lignes de la table FRUIT */
$count = $dbh->exec("DELETE FROM fruit WHERE couleur =
'rouge'");
/* Retourne le nombre de lignes effacées */
print("Effacement de $count lignes.\n");
?>
```

Requête de création de table

Exemple de création de table

Fichier creation-table.php

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

//Création de la table personne
$sql="CREATE TABLE personne ( id_personne INTEGER PRIMARY KEY
, nom VARCHAR( 20 ) NOT NULL, prenom VARCHAR( 20 ) , depart
INTEGER( 2 ))";
$sth = $dbh->exec($sql);

//Création de la table sport
$sql="CREATE TABLE sport ( id_sport INTEGER PRIMARY KEY ,
design VARCHAR( 30 ) UNIQUE NOT NULL)";
$sth = $dbh->exec($sql);

//Création de la table pratique
$sql="CREATE TABLE pratique ( id_personne INTEGER NOT NULL ,
id_sport INTEGER NOT NULL , niveau TINYINT,  PRIMARY KEY
(id_personne,id_sport))";

$sth = $dbh->exec($sql);
$dbh=NULL;
?>
```

Requête de sélection

Pour une requête de sélection :

- On utilise la méthode query()
- Les données ne sont pas affichées, elles sont mises en mémoire
- Il faut donc aller les chercher et les afficher
  - La méthode fetchAll() retourne l'ensemble des données sous forme d'un tableau PHP et libère le SGBD
  - La méthode fetch() permet une lecture séquentielle du résultat
  - Le paramètre fetch_style détermine la façon dont PDO retourne les résultats (format des résultats)

Exemple 2 Lire tous les enregistrements

Fichier lire-enregistrements.php

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

//La requête SQL
$sql = "SELECT * FROM `infos_tbl` LIMIT 0 , 30";

//Recherche des données
$sth = $dbh->query($sql);

// On voudrait les résultats sous la forme
// d'un tableau associatif
$result = $sth->fetchAll(PDO::FETCH_ASSOC);

//Affichage des résultats
foreach ($result as $row){
  echo $row['nom'];echo '-';
  echo $row['prenom'];echo '-';
  echo $row['email'];echo '<br/>';
}
$dbh=NULL;
?>
```

Requête de sélection

Exemple 3 Lire tous les enregistrements un par un (sous MS-SQL)

```
<?php
try {
  $hostname = "(local)\SQLSERVER2008";       //host
  $dbname = "world";       //Nom de la bdd
  $username = "sa";        // Utilisateur par exemple 'sa'
  $pw = "sa";     // Mot de passe de l'utilisateur

  $dbh = new PDO ("sqlsrv:server=$hostname;database=$dbname",
"$username", "$pw");
}

catch (PDOException $e) {
  echo "Failed to get DB handle: " . $e->getMessage() . "\n";
  exit;
}

$stmt = $dbh->prepare("SELECT * FROM pays");
$stmt->execute();

while ($row = $stmt->fetch()) {
  print_r($row);
  echo "<br />";
}

unset($dbh); unset($stmt);
?>
```

Autre exemple

Exemple de comptage d'enregistrements

```
<?php
include_once('connexion-PDO.php');

//Solution 1) En utilisant une requête particulière

$sql = "SELECT COUNT(*) as nbe FROM `infos_tbl` WHERE
nom='Goldorak'";
$sth = $dbh->query($sql);
$result = $sth->fetchAll();
$nombre = $result[0]['nbe'];
echo $nombre;
echo "<br/>";

// Solution 2) En comptant le nombre d'éléments présents
// dans le tableau des résultats

$sql =
"SELECT nom, prenom FROM `infos_tbl` WHERE nom='Goldorak'";
$sth = $dbh->query($sql);
$result = $sth->fetchAll();
$nombre = count($result);
echo $nombre;
$dbh=NULL;
?>
```

Requête d'insertion/modification

On utilise la méthode exec() de la classe PDO

Fichier insertion-enreg.php

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

//Insertion d'un enregistrement
$sql = "INSERT INTO infos_tbl
(id,nom,prenom,email,icq,titre,url) VALUES ('', 'Foughali',
'Karim', 'karim.foughali@univ-paris13.fr', '' , 'Enseignant',
'www.free.fr')";

//Exécution de la requête
$dbh->exec($sql);
//exec() retourne le nombre de lignes modifiées

$dbh=NULL;
?>
```

Gestion des erreurs

On utilise les exceptions comme vu précedemment:

Fichier gestion-erreur-mod-excep.php

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

//On définit le handler d'erreur
$dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

try {
  sql = "INSERT INTO infos_tbl
(id,nom,prenom,email,icq,titre,url) VALUES ('', 'Foughali',
'Karim', 'karim.foughali@univ-paris13.fr', '' , 'Enseignant',
'www.free.fr')";

  $dbh->exec($sql);
//Si une erreur a lieu, une exception sera lancée
}

catch (PDOException $e){
  print "Erreur ! :" . $e->getMessage() ; "<br/>";
}

$dbh=NULL;
?>
```

Injections SQL

Le code suivant :

```
<?
$pseudo = 'aa"; DELETE FROM users; '.
'SELECT * FROM users WHERE pseudo="';
$password = "truc";
$res = $db->exec('UPDATE users SET password="'.
md5($password).'" WHERE pseudo="'.$pseudo.'";');
?>
```

exécute la requête SQL suivante :

```
UPDATE users SET password="...." WHERE pseudo="aa";
DELETE FROM users;
SELECT * FROM users WHERE pseudo="";
```

Les **requêtes préparées** comme protection contre les injections SQL :

```
<?
$r = $db->prepare('UPDATE users SET password = ? '. 'WHERE
pseudo = ?');
$r->execute(array(md5($password),  $pseudo));
?>
```

Requêtes préparées avec prepare

• Les requêtes préparées sont un modèle de requête enregistré sur le serveur le temps de l'exécution du script (par opposition aux procédures stockées qui sont stockées de manière permanente).
• Avec les requêtes paramétrées il n'est plus nécessaire de se protéger des attaques par injection SQL car le serveur sait ce qu'il attend.

Exemple #1 Prépare une requête SQL avec des paramètres nommés

```
<?php
/* Exécute une requête préparée en passant un tableau de
valeurs */
$sql = 'SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories AND couleur = :couleur';
$sth = $dbh->prepare($sql, array(PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':couleur' =>
'red'));
$red = $sth->fetchAll();
$sth->execute(array(':calories' => 175, ':couleur' =>
'yellow'));
$yellow = $sth->fetchAll();
?>
```

Exemple #2 Prépare une requête SQL avec des marqueurs

```
<?php
/* Exécute une requête préparée en passant un tableau de
valeurs */
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');
$sth->execute(array(150, 'rouge'));
$red = $sth->fetchAll();
$sth->execute(array(175, 'jaune'));
$yellow = $sth->fetchAll();
?>
```

Requêtes préparées avec prepare & bindParam

- On utilise ici la méthode bindParam avant d'appeler execute

Exemple #1 Exécution d'une requête préparée avec des emplacements nommés (:)

```
<?php
/* Exécution d'une requête préparée en liant des variables
PHP */
$calories = 150;
$couleur = 'rouge';
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories AND couleur = :couleur');
$sth->bindParam(':calories', $calories);
$sth->bindParam(':couleur', $couleur);
$sth->execute();
?>
```

Exemple #2 Exécution d'une requête préparée avec des marques de positionnement (?)

```
<?php
/* Exécution d'une requête préparée en liant des variables
PHP */
$calories = 150;
$couleur = 'rouge';
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');
$sth->bindParam(1, $calories);
$sth->bindParam(2, $couleur);
$sth->execute();
?>
```

Gestion des transactions avec PDO

Exemple

```
try {
$dbh = new PDO('mysql:host=localhost;dbname=mybase', 'root',
'',array(PDO::ATTR_PERSISTENT => true));

$dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

$dbh->beginTransaction();

$dbh->exec("insert into etudiants (id, nom, prenom) values
(23,'Walt', 'Disney')");

$dbh->exec("insert into salaires (id, montant,
changementDate) values (23, 50000, NOW())");

$dbh->commit();
}

catch (Exception $e) {
  $dbh->rollBack();
  echo "Échec : " . $e->getMessage();
}
```

Les opérations de base à connaitre

Pour effectuer une requête SQL :

```
<?
$res = $db->query("select * from sondages");
var_dump($res);
/* affiche 'object(PDOStatement)#2 (1) {
["queryString"]=> string(19) "select * from sondages" }' */
?>
```

Pour connaître le nombre de lignes :

```
<? echo "nombre de lignes : ".$res->rowCount()."\n"; ?>
```

Pour parcourir les lignes :

```
<?
$res = $db->query("select * from sondages");
while ($ligne = $res->fetch()) {
  echo $ligne['createur']." pose la question :
".$ligne['question']."\n";
}
?>
```

Pour mettre toutes les lignes dans un tableau :

```
<?
$res = $db->query("select * from sondages");
$lignes = $res->fetchAll();
foreach ($lignes as $ligne) {
  echo $ligne['createur']." pose la question :
".$ligne['question']."\n";
}
?>
```

Autres méthodes intéressantes

Voir aussi, dans la classe PDOStatement, les méthodes:

- bindColumn : attache une variable à une colonne
- errorInfo : information d'erreur
- fetchColumn : récupère la valeur dans une colonne donnée
- closeCursor : ferme le curseur

Voir aussi, dans la classe PDO, les méthodes:

- beginTransaction : démarre une transaction
- commit : valide une transaction
- rollback : annule une transaction
- errorInfo : Retourne les informations associées à l'erreur
- errorCode : Retourne le SQLSTATE associé avec la dernière opération
- quote : Protège une chaîne pour l'utiliser dans une requête SQL PDO

```
<?
$string = 'Chaine \' particulière';
print "non échappée : $string\n";
print "échappée :" . $bd->quote($string) . "\n";
?>

// Chaine non échappée : Chaine ' particulière
// Chaine échappée : 'Chaine '' particulière'
```