PHP : Accès aux bases de données

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2020

L'API PDO

PDO - PHP Data Objects (PHP 5+)

Fournit une interface d'abstraction à l'accès aux SGBD.

- Des pilotes (drivers) sont implémentés dans l'interface PDO pour différentes BDD.
- On utilise les mêmes fonctions pour exécuter des requêtes SQL quelle que soit la BDD utilisée.

Bases de données

Bases de données supportées

- CUBRID
- Sybase/MS SQL
- Firebird/Interbase
- IBM DB2
- IBM Informix
- MySQL 3, 4, 5
- Oracle
- ODBC v3
- PostgreSQL
- SQLite 2, 3
- Microsoft SQL Server / SQL Azure

Connexion

Etablie en instanciant la classe `PDO`

Le constructeur accepte différents paramètres :

- Data Source Name (DSN) : type de SGBD, hôte, port, BDD ...
- Nom d'utilisateur (optionnel).
- Mot de passe (optionnel).
- Tableau d'options : persistance/cache des connexions, compression, sécurité ...

Connexion à MySQL (pdo-connexion-open.php)

```php
$sgbd="mysql"; // choix de MySQL
$host="localhost";
$user="lesaint";
$pass="lesaint";
$pdo = new PDO("$sgbd:host=$host",$user,$pass);
```

Gestion des erreurs

Trois modes de gestion d'erreurs (connexion, requêtes)

- Mode "silencieux" (par défaut) : aucune exception émise mais un rapport d'erreurs accessible via `PDO::errorInfo()`.
- Mode "avertissement" : émission de `E_WARNING` sans interruption du script.
- Mode "exception" : déclenchement d'une exception par instanciation de la classe `PDOException`.

Class `PDOException`

- Fournit différentes méthodes de diagnostic d'erreur dont `getCode()` qui renvoie un code d'erreur standard SQLSTATE.

Gestion des erreurs

Rapport d'erreur (pdo-connexion-error.php)

```
 1 $sgbd="mysql";
 2 $host="localhost";
 3 $db="l3_crs_test";
 4 $user="lesaint";
 5 $pass="lesaint";
 6 try {
 7    $pdo = new
PDO("$sgbd:host=$host;dbname=$db;charset=utf8", $user, $pass);
 8    // active le déclenchement d'exceptions
 9    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
10    $pdo->query("This is no SQL!");
11 } catch(PDOException $e) {
12       //42000
13       echo $e->getCode();
14       //SQLSTATE[42000]: Syntax error or access violation: 1064 You have an
error in your SQL syntax ...
15       echo $e->getMessage();
16 }
```

Déconnexion

Une connexion reste active tant que l'objet `PDO` l'est

Pour fermer une connexion avant la fin du script, détruire l'objet et ses références (eg. instances de `PDOStatement` référençant l'objet).

Fermeture optionnelle (pdo-connexion-close.php)

```php
$pdo = new PDO('mysql:host=localhost;dbname=test');
$sth = $pdo->query('SELECT * FROM foo');
// détruire toute référence à l'objet PDO
$sth = null;
$pdo = null;
```

Mise en cache de connexions persistantes

Avec option `PDO::ATTR_PERSISTENT => true` à l'instanciation de PDO.

Requêtes SQL standard et requêtes préparées

Requête SQL standard

Requête écrite suivant la syntaxe SQL (de type `string`).

Requête préparée (prepared statement)

Requête SQL paramétrée à l'aide de marqueurs nommés (syntaxe `:nom`) ou interrogatifs (syntaxe `?`).
Exécutée une ou plusieurs fois en substituant à chaque exécution les marqueurs avec les valeurs d'un tableau :

- Associatif dont les clés correspondent aux marqueurs nommés.
- Indicé où les valeurs sont substituées aux marqueurs interrogatifs par ordre d'apparition.

Requêtes SQL standard et requêtes préparées

Requête SQL standard

Passée à la méthode `query` de l'objet `PDO`.

- Renvoie une instance de `PDOStatement`.

Requête préparée (prepared statement)

Passée à la méthode `prepare` de l'objet `PDO`.

- Renvoie une instance de `PDOStatement`.

Exécutée ensuite en invoquant la méthode `execute` sur l'instance de `PDOStatement` avec le tableau de valeurs désirées.

Récupération du jeu de résultats

Avec les méthodes `fetch*` de l'instance `PDOStatement`.

Création et connexion à une base de données

Création d'une BDD (pdo-create-db.php)

```php
require 'pdo-connexion-open.php';
$db="l3_crs_test";
$str="CREATE DATABASE IF NOT EXISTS $db CHARACTER SET=utf8 COLLATE utf8_general_ci";
$pdo->query($str);
```

Connexion avec sélection de la BDD (pdo-connexion-db.php)

```php
$sgbd="mysql"; // choix de MySQL
$host="localhost";
$db="l3_crs_test";
$user="lesaint";
$pass="lesaint";
$pdo = new PDO("$sgbd:host=$host;dbname=$db",$user,$pass);
```

Création de table

Création d'une table (pdo-create-table.php)

```php
require 'pdo-connexion-db.php';
$str="CREATE TABLE `l3_crs_test`.`Employee`
    ( `id` INT NOT NULL AUTO_INCREMENT ,
    `name` VARCHAR(40) NOT NULL ,
    `salary` FLOAT NOT NULL ,
    `age` INT(100) NOT NULL , PRIMARY KEY (`id`))
  ENGINE=InnoDB
  CHARSET=utf8 COLLATE utf8_general_ci;";
try
{
    $pdo->query($str);
} catch(PDOException $e) {
    echo $e->getMessage();
}
```

Description de table

Information sur une table (pdo-read-table.php)

```php
require 'pdo-connexion-db.php';
$stt=$pdo->query("DESCRIBE Employee");
while ($record=$stt->fetch()) {
  var_dump($record);
}
```

Résultat (format mysql CLI)

```
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(40) | NO   |     | NULL    |                |
| salary   | float       | NO   |     | NULL    |                |
| age      | int(100)    | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
```

Insertion d'un enregistrement

pdo-record-create.php

```
1 require 'pdo-connexion-db.php';
2 $str="INSERT INTO Employee (name,salary,age) VALUES
('superman',10000,85)";
3 $pdo->query($str);
```

Insertion en cascade avec requête préparée

pdo-prepared-statement.php

```php
require 'pdo-connexion-db.php';
$qry="INSERT INTO Employee (name,salary,age) VALUES (:nom,:salaire,:age)";
$stt=$pdo->prepare($qry);
$data=array(
    array(':nom'=>'batman',':salaire'=>10.5,':age'=>78),
array(':nom'=>'spiderman',':salaire'=>80,':age'=>55));
foreach($data as $row) {
  $stt->execute($row);
}
```

Sélection d'enregistrements

Récupérer chaque ligne résultat

Avec `PDOStatement::fetch([int $style...])`.

Plusieurs possibilités selon valeur de `$style`

- `FETCH_NUM` : tableau indicé.
- `FETCH_ASSOC` : tableau associatif.
- `FETCH_BOTH` : tableau avec indices et clés.
- `FETCH_CLASS` : objet anonyme ou de classe prédéfinie.
- `FETCH_INTO` : objet mis à jour.

Récupération en tableau indicé

Utilisation de FETCH_NUM (pdo-record-read-num.php)

```php
require_once('pdo-connexion-db.php');
$qry="SELECT * FROM Employee WHERE name='superman'";
$stt=$pdo->query($qry);
while ($record=$stt->fetch(PDO::FETCH_NUM)) {
  print_r($record);
}
```

```
Array
(
    [0] => 1
    [1] => superman
    [2] => 10000
    [3] => 85
)
```

Récupération en tableau associatif

Utilisation de FETCH_ASSOC (pdo-record-read-assoc.php)

```php
require_once('pdo-connexion-db.php');
$qry="SELECT * FROM Employee WHERE age<80";
$stt=$pdo->query($qry);
while ($record=$stt->fetch(PDO::FETCH_ASSOC)) {
  print_r($record);
}
```

```
Array
(
    [id] => 2
    [name] => batman
    [salary] => 10.5
    [age] => 78
)
```

```
Array
(
    [id] => 3
    [name] => spiderman
    [salary] => 80
    [age] => 55
)
```

Récupération en tableau mixte

Utilisation de FETCH_BOTH (pdo-record-read-both.php)

```php
require_once('pdo-connexion-db.php');
$qry="SELECT * FROM Employee WHERE name LIKE '%atm%'";
$stt=$pdo->query($qry);
while ($record=$stt->fetch(PDO::FETCH_BOTH)) {
  print_r($record);
}
```

```
Array
(
    [id] => 2
    [0] => 2
    [name] => batman
    [1] => batman
    [salary] => 10.5
    [2] => 10.5
    [age] => 78
    [3] => 78
)
```

Récupération en objet

Utilisation de `FETCH_CLASS` sur une classe donnée `C`

Invoquer `setFetchMode(PDO::FETCH_CLASS,'C')` sur l'objet `PDOStatement` avant appel à `fetch/fetchAll`.

- Crée une instance de `C` par ligne résultat en associant propriétés et colonnes par leurs noms.

Règles d'application

- Le constructeur est sans arguments.
- S'il existe une propriété de même nom qu'une colonne, et quelle qu'en soit la visibilité, elle prend sa valeur.
- Sinon, la méthode magique `__set` est appelée.
- Ou si `__set` n'est pas définie, une propriété publique sera créée de même nom et valeur que la colonne.

Récupération en objet

Classe de récupération (pdo-employee.php)

```
 1 class Employee {
 2       public $name = "anonymous";
 3       private $id;
 4       private $salary;
 5       // OPTIONNEL public function __construct() {}
 6       function setName($name) { $this->name = $name; }
 7       function setId($id) { $this->id = $id; }
 8       function setSalary($salary) { $this->salary = $salary; }
 9       public function __set($p,$v) {
10             echo "calling __set for property $p\n";
11             $this->$p = $v;
12       }
13       public function __toString() {
14             return "employee: id=$this->id name=$this->name
salary=$this->salary age=$this->age\n";
15       }
16 }
```

Récupération en objet

Avec `FETCH_CLASS` (pdo-record-read-class.php)

```php
require_once('pdo-connexion-db.php');
require_once('pdo-employee.php');
$qry="SELECT * FROM Employee";
$stt=$pdo->query($qry);
$stt->setFetchMode(PDO::FETCH_CLASS,'Employee');
while ($employee=$stt->fetch()) {
   echo $employee."\n";
   var_dump($employee);
}
```

Ajoute une nouvelle propriété publique `age` pour chaque objet créé par appel à `__set`.

Récupération en objet

Utilisation de `FETCH_INTO` sur une instance existante `o`

Invoquer `setFetchMode(PDO::FETCH_INTO,$o)` sur l'objet `PDOStatement` avant appel à `fetch/fetchAll`.

- Met à jour l'objet `o` à chaque ligne résultat en associant propriétés et colonnes par leurs noms.

Règles d'application

- Le constructeur est sans arguments.
- S'il existe une propriété publique de même nom qu'une colonne elle prend sa valeur.
- Sinon, la méthode magique `__set` est appelée.
- Ou si `__set` n'est pas définie, une exception est lancée.

Récupération en objet

Avec FETCH_INTO (pdo-record-read-class-into.php)

```php
require_once('pdo-connexion-db.php');
require_once('pdo-employee.php');
$qry="SELECT * FROM Employee";
$stt=$pdo->query($qry);
$emp = new Employee();
$stt->setFetchMode(PDO::FETCH_INTO,$emp);
while ($stt->fetch())
   echo $emp."\n";
   var_dump($emp);
```

__set

- Initialise les propriétés inaccessibles `id` et `salary`.
- Ajoute et initialise une nouvelle propriété publique `age`.

Récupération des lignes résultats en un seul appel

Avec `PDOStatement::fetchAll([int $style...])`.

Avec `FETCH_NUM` (pdo-record-fetchall-num.php)

```php
require_once('pdo-connexion-db.php');
$qry="SELECT * FROM Employee";
$stt=$pdo->query($qry);
$records=$stt->fetchAll(PDO::FETCH_NUM);
foreach($records as $record) {
  print_r($record);
}
```

Avec `FETCH_CLASS` (pdo-record-fetchall-class.php)

```php
require_once('pdo-connexion-db.php');
require_once('pdo-employee.php');
$qry="SELECT * FROM Employee";
$stt=$pdo->query($qry);
$emps=$stt->fetchAll(PDO::FETCH_CLASS,'Employee');
foreach($emps as $emp) {
  echo $emp;
}
```

Modification et suppression d'enregistrement

Exemple (pdo-record-update.php)

```
1 require_once('pdo-connexion-db.php');
2 $qry="UPDATE Employee SET salary=200 WHERE
name='superman'";
3 $pdo->query($qry);
```

Exemple (pdo-record-delete.php)

```
1 require 'pdo-connexion-db.php';
2 $qry="DELETE FROM Employee WHERE id=2";
3 $pdo->query($qry);
```

Transactions

Effectuent un ensemble d'opérations de manière groupée

Peuvent être annulées (rollback).

Le SGBD doit garantir les propriétés d'atomicité, cohérence, isolation et durabilité des transactions (ACID)

- A Une transaction s'exécute totalement ou pas du tout (eg. en cas de panne, défaillance).
- C Une transaction doit amener la BD dans un état conforme aux règles définies (eg. contraintes d'intégrité).
- I Toute transaction doit s'exécuter comme si elle était la seule (eg. deux transactions "simultanées" doivent aboutir au même état que celui obtenu en les exécutant séquentiellement et quelle que soit la séquence).
- D Les résultats d'une transaction confirmée sont enregistrés de façon permanente.

Transactions

Certains SGBD ne prennent pas les transactions en charge.

La requête SQL a été exécutée avec succès.

```
show engines
```

Profilage [Éditer en ligne] [Éditer] [Créer le code source PHP] [Actualiser]

+ Options

Engine	Support	Comment	Transactions	XA	Savepoints
InnoDB	DEFAULT	Supports transactions, row-level locking, and fore...	YES	YES	YES
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary...	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it...	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	*NULL*	*NULL*	*NULL*

Transactions

Exemple (pdo-transaction.php)

```php
// code omis préparant la requête $qry
$pdo->beginTransaction();
if ($pdo->exec($qry) === FALSE) {
    $pdo->rollback();
}
$pdo->commit();
```