Bases de données avancées

Introduction à PDO pour MySQL

Raphaël Fournier-S'niehotta (fournier@cnam.fr)

EPN 5 – Informatique
Conservatoire National des Arts & Métiers, Paris, France

Drivers

- `mysql, mysqli`
- PDO
  - plus sûr (échappements)
  - plus portable (pas seulement MySQL)
  - gestion des erreurs

Exemple

```php
<?php
$link = mysql_connect('localhost', 'user', 'pass');
mysql_select_db('testdb', $link);
mysql_set_charset('UTF-8', $link);
```

- Maintenant : on crée un objet PDO
- 4 paramètres : DSN, username, password, tableau d'options
- DSN (*Data Source Name*: chaîne d'options pour spécifier le driver et les détails de connexion

```php
<?php
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8mb4',
                                'username', 'password');
```

Avec options

```php
<?php
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8mb4',
'username', 'password', array(PDO::ATTR_EMULATE_PREPARES => false,
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
```

On peut aussi interagir après création de l'objet :

```php
<?php
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8mb4',

$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

Requêtes SELECT simples

```php
<?php
foreach($db->query('SELECT * FROM table') as $row) {
    echo $row['field1'].' '.$row['field2']; //etc...
}
```

ou

```php
<?php
$stmt = $db->query('SELECT * FROM table');

while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['field1'].' '.$row['field2']; //etc...
}
```

Modes pour FETCH

- `PDO::FETCH_ASSOC` retourne un tableau associatif avec les attributs comme clés.
- `PDO::FETCH_NUM` retourne les lignes sous forme de tableau numérique
- par défaut, `PDO::FETCH_BOTH` est utilisé, doublant la taille (généralement inutile)

Requêtes INSERT, UPDATE, DELETE

```php
<?php
$affected_rows = $db->exec("UPDATE table SET field='value'");
echo $affected_rows.' were affected'
```

Même chose pour les INSERT et DELETE

Avec des variables

```php
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=? AND name=?");
$stmt->execute(array($id, $name));
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

- la méthode `prepare()` et est compilée avec les ? comme emplacements
- la méthode `execute()` envoie les paramètres au serveur et exécute la requête. Comme les paramètres et la requête sont envoyés séparément, les injections SQL les plus classiques sont évitées.
- ATTENTION : éviter les quotes autour des paramètres

Variables typées

```php
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=? AND name=?");
$stmt->bindValue(1, $id, PDO::PARAM_INT);
$stmt->bindValue(2, $name, PDO::PARAM_STR);
$stmt->execute();
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Emplacements nommés

```php
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=:id AND name=:name");
$stmt->bindValue(':id', $id, PDO::PARAM_INT);
$stmt->bindValue(':name', $name, PDO::PARAM_STR);
$stmt->execute();
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```php
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=:id AND name=:name");
$stmt->execute(array(':name' => $name, ':id' => $id));
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

INSERT, UPDATE, DELETE

```php
    <?php
$stmt = $db->prepare("INSERT INTO table(field1,field2,field3,field4,field5)
  VALUES(:field1,:field2,:field3,:field4,:field5)");
$stmt->execute(array(':field1' => $field1, ':field2' => $field2,
  ':field3' => $field3, ':field4' => $field4, ':field5' => $field5));
$affected_rows = $stmt->rowCount();
```

```php
      <?php
$stmt = $db->prepare("DELETE FROM table WHERE id=:id");
$stmt->bindValue(':id', $id, PDO::PARAM_STR);
$stmt->execute();
$affected_rows = $stmt->rowCount();
```

```php
<?php
$stmt = $db->prepare("UPDATE table SET name=? WHERE id=?");
$stmt->execute(array($name, $id));
$affected_rows = $stmt->rowCount();
```

Attention aux fonctions

```php
<?php
//Ne Marche PAS !
$time = 'NOW()';
$name = 'BOB';
$stmt = $db->prepare("INSERT INTO table(`time`, `name`) VALUES(?, ?)");
$stmt->execute(array($time, $name));
```

```php
<?php
$name = 'BOB';
$stmt = $db->prepare("INSERT INTO table(`time`, `name`) VALUES(NOW(), ?
$stmt->execute(array($name));
```

Mais pour les arguments, c'est bon :

```php
<?php
$name = 'BOB';
$password = 'badpass';
$stmt = $db->prepare("INSERT INTO table(`hexvalue`, `password`) VALUES(HEX(?),
$stmt->execute(array($name, $password));
```

Requêtes LIKE

```php
<?php
//Ne Marche PAS !
$stmt = $db->prepare("SELECT field FROM table WHERE field LIKE %?%");
$stmt->bindParam(1, $search, PDO::PARAM_STR);
$stmt->execute();
```

```php
<?php
$stmt = $db->prepare("SELECT field FROM table WHERE field LIKE ?");
$stmt->bindValue(1, "%$search%", PDO::PARAM_STR);
$stmt->execute();
```

Avec boucles

- L'intérêt des requêtes préparées : être appelées de nombreuses fois à la suite, avec des valeurs différentes.
- La requête est compilée, puis appelée avec des arguments différents
- On gagne significativement en performance par rapport à des appels systématiques à `mysql_query`

```php
<?php
$values = array('bob', 'alice', 'lisa', 'john');
$name = '';
$stmt = $db->prepare("INSERT INTO table(`name`) VALUES(:name)");
$stmt->bindParam(':name', $name, PDO::PARAM_STR);
foreach($values as $name) {
    $stmt->execute();
}
```

Transactions

```php
    <?php
try {
    $db->beginTransaction();

    $db->exec("SOME QUERY");

    $stmt = $db->prepare("SOME OTHER QUERY?");
    $stmt->execute(array($value));

    $stmt = $db->prepare("YET ANOTHER QUERY??");
    $stmt->execute(array($value2, $value3));

    $db->commit();
} catch(PDOException $ex) {
    //Something went wrong rollback!
    $db->rollBack();
    echo $ex->getMessage();
}
```

Erreurs

Vieux !

```php
<?php
//connected to mysql
$result = mysql_query("SELECT * FROM table", $link) or die(mysql_error($link));
```

Pas de gestion des erreurs dans le code, affichage à l'utilisateur (sécurité)

Erreurs

PDO a 3 modes de gestion des erreurs :

- `PDO::ERRMODE_SILENT`. Chaque résultat doit être vérifié, et `$db->errorInfo();` pour les détails
- `PDO::ERRMODE_WARNING` lève des Warnings Php
- `PDO::ERRMODE_EXCEPTION` lève des PDOException. Sans doute le plus propre

```php
<?php
try {
  //connect as appropriate as above
  $db->query('hi'); //invalid query!
} catch(PDOException $ex) {
  echo "An Error occured!"; //user friendly message
  some_logging_function($ex->getMessage());
}
```