PHP

Bertrand Estellon

Aix-Marseille Université

13 mars 2012

SQLite et PDO

SQLite écrit les données relatives à la base dans un unique fichier sur le disque dur du serveur. La commande **sqlite3** permet d'ouvrir la base et effectuer des requêtes SQL (pratique pour débugger).

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 moteur de bases de données (Par exemple, SQLite → MySQL).

Ouverture de la base :

```
<?
try {
 $bd = new PDO("sqlite:database.sqlite");
} catch (PDOException $e) {
 die('Erreur : ' . $e->getMessage());
}
?>
```

SQLite et PDO

Aix*Marseille université

Une fois l'instance de PDO construite, vous effectuez des requêtes avec :

- `$bd->exec($requete)` : pour modifier la base de données
- `$bd->query($requete)` : pour extraire des données de la base

Exemples :

```
$bd->exec("CREATE TABLE users (".
          " nickname char(20),".
          " password char(50)".
          ");");

$res =$bd->exec('UPDATE users SET password="'.
          md5($password).'" WHERE nickname="'.$nickname.'"; ');
echo "nombre de lignes modifiees = $res";

$res = $bd->query("select nickname from users;");
```

Injections SQL

Le code suivant :

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

exécute la requête SQL suivante :

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

Protection contre les injections SQL :

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

SQLite et PDO

Aix*Marseille université

Pour faire une requête SQL :

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

SQLite et PDO

Aix*Marseille université

Pour mettre toutes les lignes dans un tableau :

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

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

SQLite et PDO

Aix*Marseille université

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
- prepare : Prépare une requête à l'exécution et retourne un objet
- quote : Protège une chaîne pour l'utiliser dans une requête SQL PDO

```
$string = 'Chaine \' particuliere';
print "non echappee : $string\n";
print "echappee :" . $bd->quote($string) . "\n";
```

Chaine non echappee : Chaine ' particuliere
Chaine echappee : 'Chaine '' particuliere'

Base de données du projet

Aix*Marseille université

Les trois tables utilisées dans le projet :

```
users(nickname char(20), password char(50));

surveys(id integer primary key autoincrement,
        owner char(20), question char(255));

responses(id integer primary key autoincrement,
          id_survey integer,
          title char(255),
          count integer);
```

Exemple : sauvegarde d'un sondage

Aix*Marseille université

```
public function saveSurvey(&$survey) {
  if ($survey->getId()!==null) return false;
  $this->connection->beginTransaction();
  $r = $this->connection->prepare('INSERT INTO surveys(owner, '.
                                  'question) VALUES (?,?);');
  if ($r===false) {
    $this->connection->rollback(); return false;
  }
  if ($r->execute(array($survey->getOwner(),
                        $survey->getQuestion()))===false) {
    $this->connection->rollback(); return false;
  }
  $id = $this->connection->lastInsertId();
  $survey->setId($id);
  foreach ($survey->getResponses() as $response)
    if(!$this->saveResponse($response)) {
      $this->connection->rollback();
      return false;
    }
  $this->connection->commit();
  return true;
}
```

La classe Database

Aix*Marseille université

```
class Database {

  private $connection;

  public function __construct() { ... }

  public function checkPassword($nickname, $password) { ... }
  public function addUser($nickname, $password) { ... }
  public function updateUser($nickname, $password) { ... }
  public function saveSurvey(&$survey) { ... }
  public function loadSurveysByOwner($owner) { ... }
  public function loadSurveysByKeyword($keyword) { ... }
  public function vote($id) { ... }

  private function createDataBase() { ... }
  private function checkNicknameValidity($nickname) { ... }
  private function checkPasswordValidity($password) { ... }
  private function checkNicknameAvailability($nickname) { ... }
  private function saveResponse(&$response) { ... }
  private function loadSurveys($arraySurveys) { ... }
  private function loadResponses(&$survey, $arrayResponses) { ... }
}
```

Persistance et mapping objet-relationnel (ORM)

Aix*Marseille université

Les besoins :

- Sauvegarde simple des objets en base de données :

```
$user = new User();
$user->nickname = "bob";
$user->password = md5("truc");
$user->save();
```

- Création automatique des tables ;
- Chargement des objets ;
- Gestion des relations entre les objets/enregistrements ;
- ...

Quelques solutions en PHP :

- Propel
- Doctrine
- Zend Framework
- Redbean
- ...