

## Développement Web 2

Bertrand Estellon

Aix-Marseille Université

April 1, 2014

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍 ↺

Bertrand Estellon (AMU)

Développement Web 2

April 1, 2014 1 / 436

PHP

Base de données

Les requêtes et fonctions utiles

## PDO

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

- ▶ `$db->exec($request)` : pour modifier la base de données
- ▶ `$db->query($request)` : pour extraire des données de la base

Exemples :

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

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

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

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍 ↺

Bertrand Estellon (AMU)

Développement Web 2

April 1, 2014 165 / 436

PHP

Base de données

PDO

## PDO

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

Ouverture de la base :

```
<?
$dbHost = $_SERVER['dbHost']; $dbBd = $_SERVER['dbBd'];
$dbPass = $_SERVER['dbPass']; $dbLogin = $_SERVER['dbLogin'];
$url = 'mysql:host='.$dbHost.';dbname='.$dbBd;
$db = new PDO($url, $dbLogin, $dbPass);
if (!$db) die("impossible d'ouvrir la base de données.");
$this->createDataBase();
?>
```

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍 ↺

Bertrand Estellon (AMU)

Développement Web 2

April 1, 2014 164 / 436

PHP

Base de données

Les requêtes et fonctions utiles

## Injections SQL

Le code suivant :

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

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍 ↺

Bertrand Estellon (AMU)

Développement Web 2

April 1, 2014 166 / 436

## PDO

Pour faire 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";
}
?>
```

## PDO

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";
}
?>
```

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

## PDO

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

Chaîne non échappée : Chaîne ' particulière

Chaîne échappée : 'Chaîne " particulière'

## Base de données du projet

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

```
<?
public function saveSurvey(&$survey) {
    $this->connection->beginTransaction();

    $query = $this->connection->prepare("INSERT INTO surveys(owner,question)".
        "VALUES (?,?)");
    if ($query===false) { $this->connection->rollback(); return false; }

    $r = $query->execute(array($survey->getOwner(), $survey->getQuestion()));
    if ($r === false) { $this->connection->rollback(); return false; }

    $id = $this->connection->lastInsertId();
    $survey->setId($id);

    $responses = &$survey->getResponses();
    foreach ($responses as &$response) {
        if ($this->saveResponse($response)===false) {
            $this->connection->rollback(); return false;
        }
    }
    $this->connection->commit(); return true;
}
?>
```

## Mapping objet-relationnel (ORM)

## 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
- ▶ Redbean
- ▶ ...

## La classe Database

```
<? 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) { ... }
}
?>
```

## Redbean – Introduction

## Initialisation de la connexion à la base de données :

```
<?
$dbHost = $_SERVER['dbHost']; $dbBd = $_SERVER['dbBd'];
$dbPass = $_SERVER['dbPass']; $dbLogin = $_SERVER['dbLogin'];
$url = 'mysql:host='.$dbHost.'.dbname='.$dbBd;
R::setup($url, $dbLogin, $dbPass);
?>
```

## Pour vider la base de données :

```
<?
R::nuke();
?>
```

## Création et sauvegarde d'un bean :

```
<?
$user = R::dispense('user');
$user->nickname = $nickname;
$user->password = $password;
R::store($user);
?>
```

## Redbean – Chargement des beans

Chargement d'un bean à partir de son identifiant :

```
<?
$user = R::load('user', $_SESSION['id']);
/* retourne NULL si le bean n'a pas été trouvé. */
?>
```

Charger un bean dans la base de données à partir d'une requête :

```
<?
$user = R::findOne('user', 'nickname = ? AND password = ?',
    array($nickname, $password));
/* retourne NULL si le bean n'a pas été trouvé. */
?>
```

Charger plusieurs beans :

```
<?
$surveys = R::find('survey', 'question like ?',
    array('%'.$keyword.'%'));
/* retourne un tableau. */
?>
```



## Redbean – One-to-many

Le chargement de l'intégralité du sondage est automatique :

```
<?
$surveys = R::find('survey', 'owner = ?', array($_SESSION['id']));

foreach ($surveys as &$survey) {
    $total = 0;
    foreach ($survey->ownResponse as $r)
        $total += $r->count;

    if ($total===0) {
        foreach ($survey->ownResponse as &$r)
            $r->percentage = 0;
    } else {
        foreach ($survey->ownResponse as &$r)
            $r->percentage = (100*$r->count)/$total;
    }
}
?>
```



## Redbean – One-to-many

Association "one-to-many" entre les sondages et les réponses :

```
<?
$survey = R::dispense('survey');
$survey->owner = $_SESSION['id'];
$survey->question = $question;

$survey->ownResponse = array();
foreach ($titles as $title) {
    $response = R::dispense('response');
    $response->title = $title;
    $response->count = 0;
    $survey->ownResponse[] = $response;
    /* Le nom du champ est important ! */
}

R::store($survey); /* Tout est sauvé */
?>
```



Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.

