M3.23.4 SI

PHP est la persistance des données

Jean-François Berdjugin
IUT1 Grenoble, dept. SRC 2009

Introduction

- Comment assurer la persistance des données

=>

- Des fichiers
  - simples
  - XML
  - SQLite
- Des SGBD
  - Sans couche d'abstraction
  - Avec une couche d'abstration
    - Procédurale
    - Object Relationnel Model

fichiers

```
<?php

$name="d:\beurk.txt";
$f = fopen($name,'w');
if (!isset($f))
die("Unable to open file!");

if (is_writable($name))
{
$ok = fwrite($f, "blabla");
if ($ok === false)
{
die ('ecriture impossible '.$f.'');
}
$ok = fwrite($f, "blibli");
if ($ok === false)
{
die ('ecriture impossible '.$f.'');
}
}
else
die ("ecriture impossible");


fclose($f);
?>
```

```
<?php
$name="d:\beurk.txt";
$f = fopen($name,'r');
if (!isset($f))
die("Unable to open file!");
$contents = '';
while (!feof($f)) {
  $contents .= fread($f, 8192);
}
echo $contents;
fclose($f);
?>
```

fichiers

Avantage :

- Un début de persistance

Problèmes :

- Droits d'accès
- Structure du contenu

XML

Avantages :

- Une solution au contenu
- Plusieurs API possibles
- Utile pour les RSS
- Utile pour AJAX

XML

```
<users>
   <user>
        <email> vl@ujf-grenoble.fr </email>
        <password> passvl </password>
   </user>
   <user>
        <email> jfb@ujf-grenoble.fr </email>
        <password> passjfb </password>
   </user>
</users>
```

XML

```
$smpXML = simplexml_load_file('../ressources/users.xml');

foreach ($smpXML->user as $user)
{echo "$user->email <br/>";
echo "$user->password <br/>";
}
```

```
$smpXML = simplexml_load_file('../ressources/users.xml');

$ajoutUser = $smpXML->addChild('user');
$ajoutUser->addChild('email','un mail');
$ajoutUser->addChild('password','un pass');
echo $sxe->asXML();
```

XML

- Reste le problème des droits d'accès et des accès concurent

⇒Utiliser un SGBD

- SQLite un SGBD léger qui utilise des fichiers

SQLite

Avantages

- Fourni avec PHP

Inconvénients

- Manque de fonctionnalité
- Non optimisé

=> SGBD « lourd »

SQLite

```
<?php
$db = sqlite_open("base_cm");
@sqlite_query($db,"CREATE TABLE user (email varchar(50), password varchar(50))");
@sqlite_query($db,"INSERT INTO user(email, password) VALUES('vl@ujf-grenoble.fr','passvl')");
@sqlite_query($db,"INSERT INTO user(email, password) VALUES('jfb@ujf-grenoble.fr','passjfb')");


$result = sqlite_query($db,"SELECT * FROM user");



while($val = sqlite_fetch_array($result)) {
echo $val["email"]."<br/>";
echo $val["password"]."<br/>";
}


sqlite_close($db);
?>
```

SGBD sans couche d'abstraction

Avantage

- Optimisation

Inconvénients

- Non portabilité

MySQL

```
$host = "localhost";
$user = "user";
$password = "password";
$database = "user_db";

$conn = mysql_connect($host, $user, $password);
if (!$conn) {
   die( "Connexion impossible");

}
mysql_select_db($database);

$result = mysql_query("SELECT * FROM user;");
if ($result > 0) {
   while($row = mysql_fetch_assoc($result)) {
      echo $row["email"] . "<br/>";
      echo $row["password"] . "<br/>";
   }
}
mysql_close();
```

SGBD avec couche d'abstraction

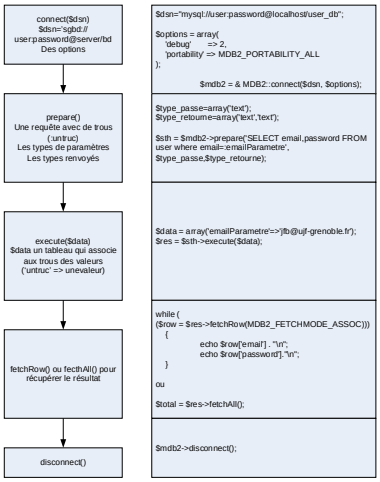
Etre le plus indépendant possible du SGBD et donc le plus portable.

Exemple : PEAR MDB2, AdoDB, Zend_db, PDO

PEAR MDB2 avec requête préparée

Les requêtes préparées limites les problèmes d'encodage et l'SQL injection.

Requête préparée



PEAR MDB2 exemple

```
$dsn="mysql://user:password@localhost/user_db";
$options = array(
   'debug'       => 2,
   'portability' => MDB2_PORTABILITY_ALL
);

$mdb2 = & MDB2::connect($dsn, $options);
if (MDB2::isError($mdb2))
      die($mdb2->getDebugInfo().'<BR/>'.$mdb2->getMessage());


$type_passe=array('text'); //les types des arguments passes en
        parametres
$type_retourne=array('text','text'); //les types retour de la requete
//la requete
$sth = $mdb2->prepare('SELECT * FROM user where
        email=:emailParametre',
$type_passe,$type_retourne);
//un jeu de paramètre
$data = array('emailParametre'=>'vl@ujf-grenoble.fr');

$res = $sth->execute($data); //execution de la requete avec les
        paramètres choisi
if (PEAR::isError($res))
die($res->getMessage());
```

```
//affichage des résultats

if ($row = $res->fetchRow(MDB2_FETCHMODE_ASSOC))
{
$email= $row['email'] . "\n";
$password = $row['password']."\n";
echo "$email $password";
}
else
{
echo "utilisateur pas la";
}
```

Pourquoi continuer avec du SQL ?

Utiliser des ORM est manipuler des objets classiques.

Nombreuses solutions : Propel, Doctrine, PHPMyObject, EZPDO, ...

EZPDO exemple

- Un fichier de configuration

```
<options>
 <source_dirs>classes</source_dirs>
 <compiled_dir>compiled</compiled_dir>

  <default_dsn>mysql://user:password@localh
  ost/user_db</default_dsn>
</options>
```

EZPDO exemple

- Une classe

```
class User
{

/**
 * @orm char(256)
 */
public $email;

/**
 * @orm char(256)
 */
public $password;
}
```

EZPDO exemple

- Un gestionnaire d'entité

```
require_once('../../lib/ezpdo/ezpdo_runtime.php');
epLoadConfig('config.xml');
$m = epManager::instance();

//creation
$user = $m->create('User');
$user->email = 'mon email';
$user->password = 'mon mot de passe';
$m->commit($user); //creation effecttive

//recherche
//creation du modele
$user = $m->create('User');
$user->email = 'mon email';
$user->password=null;
$users = $m->find($user);

foreach ($users as $user)
echo $user->email.'<br/>'.$user->password;
```

EZPDO

- Annotations
  - Pour les tables
  - Pour les attributs
  - Pour les associations (@orm [has|composed_of] [one|many] NomClasse [inverse[(var)]])
- Reste limité la clef primaire est imposée