

Web (Persistence)



Andrea G. B. Tettamanzi

Université de Nice Sophia Antipolis

Département Informatique

andrea.tettamanzi@unice.fr

CM - Séance 9

Interfaçage avec des bases de données

Plan

- Connexion à une SGBD
- Interroger la BD
- Exploitation résultats
- Métadonnées
- Encodage
- Jointure et « doublons »
- Fichiers et images

Connexion à un SGBD

- La connexion se fait en trois temps :
 - La connexion
 - La sélection de la base de donnée
 - Exploitation de la bd (selon droits & besoins)
- Support de plusieurs SGBD par php (mysql, postgres, etc.)
- Les fonctions varient selon la SGBD
 - Les noms
 - Certaines capacités particulières (rétro-conception)
 - Mais les principes restent...
- Encapsulation orientée objet : PHP Data Objects (PDO)
- API (pilotes) spécifiques pour chaque SGBD (ex. : MySQL)

Connexion en PDO

- ```
$user = "visiteur";
$pass = "toctoc";
$dbh = new PDO('mysql:host=foo.net; dbname=bar',
 $user, $pass,
 array(PDO::MYSQL_ATTR_INIT_COMMAND
 => "SET NAMES 'UTF8'")
);
```
- Le type de SGBD est précisé dans le 1 er paramètre
- des erreurs de connexion => PDOException
- Fermeture automatique ou `$dbh = null;`
- Connexion persistante possible

# Connexion avec pilote MySQL

- `$connexion = mysql_connect('foo.net', 'visiteur', 'toctoc');`
  - Connexion à un serveur mysql nommé foo.net
  - En tant que l'utilisateur nommé « visiteur »
  - Avec le mot de passe « toctoc »
- Si la connexion échoue, `$connexion` vaut « faux »
  - fonction `exit()` ou `die()` pour mettre fin au script (test or die)
  - Ou redirection
- La connexion au serveur sera fermée
  - À la fin du script
  - Par appel explicite à `mysql_close( $connexion )`.
- Suppression des messages d'erreur en cas d'échec avec `@`

## *Exemple de connexion (PDO)*

```
try {
 $dbh = new PDO('mysql:host=eutерpe.unice.fr;
 dbname=coursRenevierGonin',
 $user, $pass);
} catch (PDOException $e) {
 echo "Erreur !: " .
 $e->getMessage() . "
";
 die();
}
```

## *Exemple de connexion (MySQL)*

```
$server = "euterpe.unice.fr";
if ($_SERVER["SERVER_NAME"]=="localhost")
 $server = "localhost";
$connexion = mysql_connect($server, 'visiteur',
 'toctoc');
if (! $connexion)
{
 die("<div>pb de connexion</div>");
}
```



# Requête SQL : interroger la BD (PDO)

- Requête directe
  - `public PDOStatement PDO::query ( string $statement )`
  - D'autres variantes pour préciser le retour
  - (voir aussi `PDOStatement::setFetchMode() )`
- Requête directe sans retour
  - `public int PDO::exec ( string $statement )`
- Requête(s) préparée(s)
  - `public PDOStatement PDO::prepare ( string $statement`
  - `[, array $driver_options = array() ] )`
  - `public bool PDOStatement::execute ([ array $input_parameters ] )`
- Transactions possibles (`beginTransaction ; commit ; rollBack`)
  - Atomicité, Cohérence, Isolation et Durabilité (ACID)

# Requête SQL : interroger la BD (MySQL)

- Sélection de la BD avec `mysql_select`  
`$bd = @mysql_select_db('test', $connexion);`
  - Choix de la BD `test` au travers de la connexion `$connexion`
  - Le second paramètre peut être omis, mais c'est la dernière connexion ouverte (si elle existe) qui est prise en compte
- `$requete = mysql_query("select * from magasin;", $connexion);`
  - `mysql_query` permet de faire n'importe quel requête SQL
  - Ici : sélection de tout ce qui est dans la table choisie
  - N.B. : `$bd` n'apparaît pas, mais IL FAUT faire le choix de BD
  - Le second paramètre peut être omis, mais c'est la dernière connexion ouverte avec `mysql_connect` (si elle existe) qui est prise en compte

## Valeur de retour (PDO::query)

- En cas d'échec, PDO::query renvoie FALSE
- Sinon, La classe PDOStatement encapsule les réponses
  - « Représente une requête préparée et, une fois exécutée, le jeu de résultats associé. »
  - Résultats à parcourir
  - PDOStatement implements Traversable
  - Donc, utilisable avec un foreach...
- « PDO::errorCode — Renvoie le SQLSTATE associé avec la dernière opération sur la base de données »
- PDO::errorInfo — Renvoie les informations associées à l'erreur lors de la dernière opération sur la base de données

## Valeur de retour (*mysql\_query*)

- Pour les requêtes du type SELECT, SHOW, DESCRIBE ou EXPLAIN, `mysql_query( )` renvoie une ressource en cas de succès, ou FALSE en cas d'erreur.
- Pour les autres types de requêtes, UPDATE, DELETE, DROP, etc., `mysql_query( )` retourne TRUE en cas de succès ou FALSE en cas d'erreur.
- `mysql_query( )` échouera et retournera FALSE si l'utilisateur n'a pas les autorisations nécessaires pour accéder à la (aux) table(s) référencée(s) par la requête.

## Accéder aux erreurs (*mysql\_query*)

- Les erreurs retournées par le serveur MySQL ne génèrent plus de message d'alerte.
- `mysql_error( $connexion )` retourne le message d'erreur généré par la dernière commande MySQL. Notez que cette fonction ne retourne que le texte de l'erreur la plus récente (n'incluant pas `mysql_error()` et `mysql_errno()`)
- `mysql_errno( $connexion)` retourne le numéro d'erreur de la dernière commande MySQL.

## *Exploitation résultats (PDO)*

- PDOStatement::rowCount — Retourne le nombre de lignes affectées
- Un PDOStatement peut être vu comme une liste de tableau associatif (dont les clefs sont les nom des colonne des tables)
- Possibilité d'obtenir la réponse colonne par colonne avec :
  - PDOStatement::fetchColumn ([ int \$column\_number = 0 ] )
  - Retourne une colonne depuis la ligne suivante d'un jeu de résultats
- Possibilité de transformer une ligne (= une réponse) en un objet avec :
  - public mixed PDOStatement::fetchObject ([ string \$class\_name = "stdClass" [, array \$ctor\_args ] ] )

# Exploitation résultats (MySQL)

- Utilisez `mysql_num_rows( )` pour trouver le nombre de lignes retournées pour une requête du type `SELECT` ou `mysql_affected_rows( )` pour trouver le nombre de lignes affectées par les requêtes du type `DELETE`, `INSERT`, `REPLACE`, ou `UPDATE`.
- La ressource de résultat (cas d'un `SELECT`) retournée doit être passée par l'une des fonctions permettant d'explorer le résultat des tables, pour accéder aux données retournées.
  - `mysql_fetch_array( $requete )` : retourne un tableau qui contient la ligne demandée et déplace le pointeur de données interne d'un cran ou `FALSE` s'il n'y a plus de lignes.

# Exploitation résultats (MySQL)

- Le type de tableau retourné dépend d'un second paramètre facultatif `result_type`.
  - `MYSQL_BOTH` (défaut), vous récupèrerez un tableau contenant des indices associatifs et numériques.
  - LES CLEFS SONT LES NOMS DES CHAMPS DANS LA TABLE DE LA BASE DE DONNEES
  - En utilisant `MYSQL_ASSOC`, vous ne récupèrerez que les indices associatifs, en utilisant `MYSQL_NUM`, vous ne récupèrerez que les indices numériques (indice = n° de colonne du champ dans la table)
- Existe aussi la version objet :
  - `mysql_fetch_object`



## Exemple (PDO)

```
$pdo = new PDO('mysql:host=localhost;dbname=test;
 charset=utf8', 'login', 'password');
$list = "";
$resultats = $pdo->query("select * from exemple;");
if ($resultats && ($resultats->rowCount() >0)) {
 $list = "";
 foreach ($resultats as $ligne) {
 $adr = "$adrbase".$ligne['adresse'];
 $list .= " cours {$ligne['cours']} /
{$ligne['adresse']} est {$ligne['description']}
";
 $list .= "le fichier est disponible <a href=\"\"
 $adr.\"\">à cette adresse";
 }
 $list .= "";
}
else $list .= "<div>pb de requete</div>";
```

## Exemple (MySQL)

```
$requete = mysql_query("select * from exemple;",
$connexion);
$list = "";
if ($requete) {
 $list = "";
 while ($resultat = mysql_fetch_array($requete)) {
 $adr = "$adrbase".$resultat['adresse'];
 $list .= " cours ".$resultat['cours']." /
".$resultat['adresse']." est
".$utf8_encode($resultat["description"])."
";
 $list .= "le fichier est disponible <a href=\"".
$adr."\">à cette adresse";
 }
 $list .= "";
}
else $list .= "<div>pb de requete</div>";
```

# Métadonnées (PDO)

- `public int PDOStatement::columnCount ( void )`
  - retourne le nombre de colonne dans la réponse
  - (après `execute( )` si requête préparée)
- `public array PDOStatement::getColumnMeta ( int $column )`
  - FONCTION EXPERIMENTALE DEPEND DU DRIVER
  - Permet d'obtenir des informations pour la colonne `$column+1`
    - `Name ; table ; driver:decl_type ; native_type (php)`
- Liste des BD / Tables accessibles
  - `SHOW databases ;`
  - `SHOW tables ;`
  - ... puis traitement de la requête ...
- Sinon Dans MySQLi (`mysqli_result::fetch_field`)

# Métadonnées (MySQL)

- `mysql_num_fields( $requete )`
  - retourne le nombre de champs d'un jeu de résultat en cas de succès, ou FALSE si une erreur survient.
- `mysql_num_rows( $requete )`
  - retourne le nombre de lignes dans un jeu de résultats en cas de succès, ou FALSE si une erreur survient.

# Métadonnées (MySQL)

- `mysql_fetch_field( $requete)`
  - retourne un objet contenant les informations sur les champs. Cette fonction peut être utilisée pour obtenir des informations sur les champs de la requête fournie.
    - `name` - nom de la colonne
    - `not_null` - 1 si la colonne ne peut pas être NULL
    - `primary_key` - 1 si la colonne est une clé primaire
    - `multiple_key` - 1 si la colonne est une clé non unique
    - `numeric` - 1 si la colonne est numérique
    - `type` - le type de la colonne
    - etc

# Encodage

- MySQL répond par défaut en iso-latin1
  - Conversion « basique » : utf8\_encode
  - Utilisation de l'extension « livrée de base »
- ```
$encoding_sql = mysql_client_encoding($connexion);
while ($result = mysql_fetch_row( $requete) ) {
    $table .= "<tr>\n\t<th>" .(++$i). "</th>\n";
    foreach($result as $value) {
        $txt = iconv($encoding_sql, "UTF-8", $value);
        // encodage cible : utf-8
        $table .= "\t<td>".$txt."</td>\n";
    }
    $table .= "</tr>\n";
}


|  |
|--|
|  |
|--|


```

Jointure et « Doublons »

- Indexation numérique
 - Tous les champs y sont
 - Dans l'ordre de la jointure, dans l'ordre des tables
- Indexation associative
 - Écrasement par la dernière valeur (dernière table de la jointure)

Fichiers et BD

- Stockage d'une « url » (ou chemin local)
- Ecriture d'un BLOB
 - Binary large object
 - `file_get_contents` pour convertir les fichiers en String
- Exemple de restitution :
 - `image.php?id=1&type=portrait`

Ajout d'une image dans une BD

```
public function addAfficheAuFilm(Data_Film $f, $image) {
    $retour = false;
    $fid = $this->filmToId($f);
    $ext3 = substr($file, -4);
    $ext4 = substr($file, -5);
    $type = "image/png";
    if (($ext3 == ".jpg") || ($ext4 == ".jpeg"))
        $type = "image/jpg";
    else if ($ext3 == ".gif") $type = "image/jpg";
    $content = addslashes(file_get_contents($image));
    if ($content) {
        $query = "insert into affiche (afficheId, film,
            image, type) values ('','$fid','$content',
            '$type')";
        $retour = $pdo->query($query);
    }
    return $retour;
}
```

Restitution d'une image dans une BD

```
require "../includes/connexion.inc";
if ( isset($_GET['id']) ) {
    $id = intval ($_GET['id']);
    $table = " afficheId, image, type FROM affiche WHERE
        afficheId = ";
    if (isset($_GET['type'])) {
        if ($_GET['type'] == "portrait")
            $table = " portraitId, image, type FROM portrait
                WHERE portraitId =";
    }
    $req = "SELECT $table ".$id;
    $ret = $pdo->query ($req) or die (mysql_error ());
    if ( !$ret[0] ) { echo "Id d'image inconnu"; }
    else {
        header ("Content-type: {$ret[2]}");
        echo $ret[1];
    }
}
else { echo "Mauvais id d'image"; }
```

Merci de votre attention

