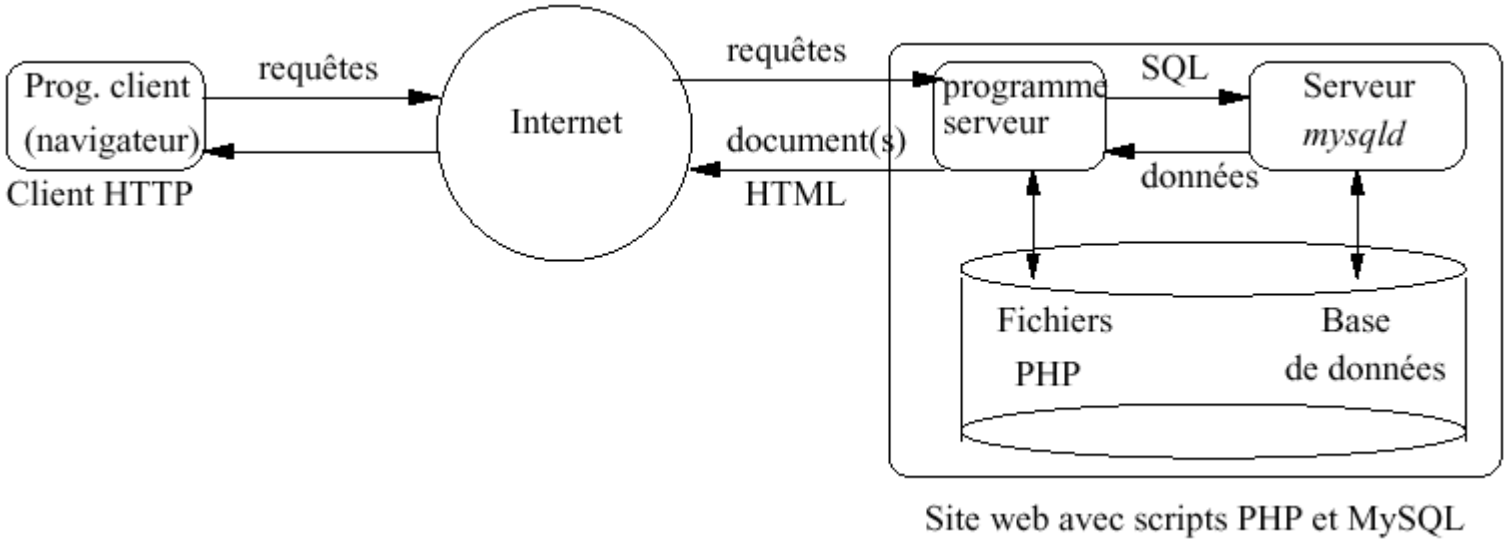
PHP /MySQL
Interface d'accès aux BDDs
PDO

Youssef CHAHIR

PLAN

- ✔ Architecture
- ✔ Créer une interface PHP/MySQL :
  - ➔ Établir une connexion
  - ➔ Exécuter une requête
  - ➔ Gérer les erreurs
  - ➔ Exploiter les résultats de la requête
  - ➔ Fermer la connexion

Architecture d'un site web avec MySQL/PHP



Site web avec scripts PHP et MySQL

- **Navigateur**: *Interface graphique*
  - Rôle : Permettre à l'utilisateur de visualiser et d'interagir avec l'information ;
- **MySQL** : Serveur de données
- **Fichiers PHP** : Serveur d'application
  - Associé à Apache qui se charge de transférer les documents produits sur l'Internet.

Bases De Données

- MySQL
  - `mysql_connect($host, $user, $pass)`
- MySQLi
  - `new mysqli($host, $user, $pass)`
- PostgreSQL
  - `pg_connect("host=$host port=$port dbname=$db")`
- SQLite
  - `new sqlite_database("db.sqlite")`

La solution PDO

- Interface commun à plusieurs SGBD .
- Caractéristiques
  - Performance
    - Avantage de la puissance de PHP 5
  - Puissance
    - Ecrit en C,==> RAPIDE !
    - Fonctions particulières (pour spécialistes)
  - Facilité
    - Clarté
  - Extension
    - Drivers peuvent être chargés

Drivers diponibles

- Oracle OCI [PDO_OCI]
- ODBC V3, IBM DB2 [PDO_ODBC]
- MySQL 3.x [PDO_MYSQL]
- Postgres [PDO_PGSQL]
- SQLite 3.x [PDO_SQLITE]
- Firebird [PDO_FIREBIRD]

Connexion à un serveur MySQL

Paramètres de connexion:

- define(DB_DSN, "mysqlhost.info.unicaen.fr;port=3333");
- define(DB_NAME, "chahir");
- define(PASSWD, "toto");
- define(USER, "chahir");

Fonction:

- $dsn = "mysql:host=" . DB_DSN . ";dbname=" . DB_NAME;
- $db = new PDO($dsn, USER, PASSWD);
  - Etablir une connexion avec serveur (MySQL), pour un compte utilisateur, et de mot de passe secret.
  - $options = array(
  - PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8", PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION, PDO::ATTR_PERSISTENT => true);
- $db = new PDO($dsn, USER, PASSWD, $options);

Connexion persistante : elle n'est pas fermée à la fin du script, mais mise en cache et réutilisée lorsqu'un autre script demande une connexion en utilisant les mêmes paramètres

Exemple

```php
<?php
try {
  $dns = 'mysql:host=mysql.info.unicaen.fr;port=3333;dbname=chahir';
  $utilisateur = 'chahir';
  $motDePasse = 'toto';

  // Options de connection
  $options = array(
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
    PDO::ATTR_PERSISTENT => true
  );
  $connection = new PDO( $dns, $utilisateur, $motDePasse, $options );
} catch ( Exception $e ) {
  echo "Connection à MySQL impossible : ", $e->getMessage();
  die();
}
?>
```

PLAN

- ✔ Architecture
- ✔ Interface phpMyADmin
- ✔ Créer une interface PHP/MySQL :
  - ➔ Établir une connexion
  - **Exécuter une requête**
    - › Exploiter les résultats de la requête
    - › Traiter les erreurs
    - › Fermer la connexion

Exécuter une requête

- Deux manières :

- Exécution Directe
  - Utilisation de la méthode query()
  - Utilisation de la méthode exec()
- Requête préparée *
  - Utilisation de la méthode prépare

Exécution Directe

Utilisation de la méthode query()

- Requêtes pour récupérer les informations
  - SELECT
- Retourne FALSE en cas d'erreur

Exemple:

$res = $db->query("SELECT * FROM Regions");

• On doit indiquer comment on compte utiliser les résultats ?

PDO_FETCH_NUM	Array with numeric keys
PDO_FETCH_ASSOC	Array with string keys
PDO_FETCH_OBJ	$obj->name holds the 'name' column from the row

- En tant qu'objet :

$res->setFetchMode(PDO::FETCH_OBJ);

Mode de récupération?

- On peut préciser le mode : avec setFetchMode(...)

  Exemple :

  $resultats->setFetchMode(PDO::FETCH_OBJ); // on dit qu'on veut que le résultat soit récupérable sous forme d'objet

- Sinon, on le précise pour fetch()

```
$res = $db->query("SELECT * FROM films");
while ($ligne = $res->fetch(PDO::FETCH_NUM)){
        // tableau numérique
        echo  $ligne[0] ;
}
$res = $db->query("SELECT * FROM films ");
while ($ligne = $res->fetch(PDO::FETCH_ASSOC)){
        // tableau associatif
        echo  $ligne['titre'] ; //print_r($row);
}
```

Exemple

```
...
$requete="SELECT titre FROM Film where titre like '%la%' ";
$res=$db->query($requete););
while( $ligne = $res->fetch(PDO::FETCH_OBJ) ) /
{
      echo $ligne->titre.'<br />'; // on affiche les membres
}

ou
while( $ligne = $res->fetch(PDO::FETCH_ASSOC) )
{
echo $ligne['titre'].'<br />'; // on affiche les membres
}

...
```

fetchAll()

- fetchAll() permet de récupérer tous les résultats d'une requête.

```
$res1 = $db->query($requete);
$tab = $res1->fetchAll(PDO::FETCH_ASSOC);

//var_dump($tab);
for ($i=0; $i<count($tab); $i++) {
        echo $tab[$i]["titre"]."-".$tab[$i]["genre"]."<br/>";
}
```

Exécution Directe

Utilisation de la méthode exec()

- Requêtes de changement
  - Insertion INSERT
  - Suppression : DELETE
  - Mise à jour: UPDATE
- La valeur de retour indique le nombre de lignes affectés par l'opération ou FALSE en cas d'erreur
- Rq: Pas de changement ==> return 0
- Exemple:

```
$db = new PDO("...");
$res = $db->exec("UPDATE foo SET id='bar'");
$res = $db->exec("INSERT INTO foo (id) VALUES('bar')");
$res = $db->exec("DELETE foo WHERE id='bar'");
if ($res !== FALSE)  echo "Erreur !!";
```

Récupérer le dernier identifiant généré automatiquement

`lastInsertId()` ↰

- renvoie l'identifiant généré sur un champ de type auto_increment à la suite d'un ordre INSERT

```
<?php
....

if ($db->exec("INSERT INTO …")) {
    $id = $db->lastInsertId();
}
?>
```

Requête préparée

Utilisation de la méthode prepare

- Prépare une requête SQL a être exécutée
  - Marqueurs qui seront substitués lors de l'exécution
  - Deux types de marqueurs: ? et marqueurs nominatifs *
- Avantages:
  - Optimisation des performances pour des requêtes appelées plusieurs fois
  - Protection des injections SQL

* Pas traitée ici

Exemple

Utilisation sans marqueurs

```
<?php
      // ouverture d'une connexion
      ...
      $requete="SELECT titre FROM Film where titre like '%la%' ";
    $resultats=$db->prepare($requete"); // on prépare notre requête
     $resultats->execute();
     while($lignes=$resultats->fetch(PDO::FETCH_OBJ))
     {
           echo $lignes->titre.'<br />';
     }
    $res = null;
    $db = null;
?>
```

PLAN

- ✔ Architecture
- ✔ Interface phpMyADmin
- ✔ Créer une interface PHP/MySQL :
  - ➔ Établir une connexion
  - ➔ Exécuter une requête
    - › Exploiter les résultats de la requête
    - › **Traiter les erreurs**
    - › Fermer la connexion

Gestion des erreurs

- Deux méthodes:
  - errorCode() – Code de l'erreur en rapport avec le SQL
    - echo $db->errorCode() ;
    - Ex. 42000 == Syntax Error
  - errorInfo() – Erreur détaillée
    - print_r($db->errorInfo());
    - Ex. array(
      [0] => 42000,
      [1] => 1064
      [2] => You have an error in your SQL syntax; …
)

Gestion des erreurs

- PDO offre 3 modes d'erreur:

```
$db->setAttribute(PDO_ATTR_ERRMODE, $mode);
(1)PDO_ERRMODE_SILENT  / (2) PDO_ERRMODE_WARNING / (3)PDO_ERRMODE_EXCEPTION
```


```
if (!$db->query($sql)) {
    echo $db->errorCode() . "<br>";
    $info = $db->errorInfo(); // $info[0] == $db->errorCode()  unified error code
    // $info[1] is the driver specific error code; // $info[2] is the driver specific error string
  }
```


```
try {   $db->exec($sql);}
catch (PDOException $e) {  // display warning message print
    $e->getMessage();
    $info = $e->errorInfo();
    // $info[0] == $e->code;  unified error code // $info[1] is the driver specific error code
    // $info[2] is the driver specific error string
 }
```

Traitement des erreurs

- ✔ Ajout d'un opérateur @ évite la sortie du script.
- ✔ Fonction die() provoque l'interruption du script.

Exemple

```
<?php
try {
$dns = 'mysql:host=mysql.info.unicaen.fr;;dbname=chahir_bd'; $utilisateur = 'chahir'; $motDePasse = 'toto';
// Options de connection
$options = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",PDO::ATTR_ERRMODE =>
PDO::ERRMODE_EXCEPTION);
$connection = new PDO( $dns, $utilisateur, $motDePasse, $options );
echo "Connection etablie <br/>";
$today = date("j, n, Y");  echo "Nous sommes $today <br>";
$jour=date("j");  $mois=date("n");
// On envoie la requète
$select = $connection->query("SELECT * FROM fetes where jour=$jour and mois=$mois");
// On indique que nous utiliserons les résultats en tant qu'objet
$select->setFetchMode(PDO::FETCH_OBJ);
// Nous traitons les résultats en boucle
while( $ligne = $select->fetch() ){
// Affichage des enregistrements
echo '<h1>', $ligne->jour, ' ', $ligne->fete, '</h1>';
}
} catch ( Exception $e ) {
echo "Connection à MySQL impossible : ", $e->getMessage(); die();
}?>
```