



Développement

PDO : Accès aux BD en PHP Objet

EEMI, Ecole Européenne des Métiers de l'Internet

Par Philippe Giraud

philippe.giraud@enseignant-eemi.com



Etat des lieux

- Nous savons comment accéder à MySQL
 - Avec des fonctions spécifiques
 - mysql_connect, mysql_select, mysql_query ...
- Et si on devait changer de BD ?
 - Il faudrait réécrire nos programmes
 - Avec les fonctions adaptées
- Peut-on s'isoler et être compatible ?



PDO : PHP Data Objects

- Permet de :
 - Créer du code d'accès aux bases de données
 - Indépendamment du moteur utilisé
 - Ou presque ...
- Avec une écriture « objet »
 - Avantage à la réutilisation
 - Gestion d'erreurs facilitée
- Ecriture obligatoire en PHP 6 (à vérifier ...)

Configuration serveur

- Comme pour d'autres sujets
 - Il faut vérifier le paramétrage du serveur
 - Fichier php.ini
 - Exemple WAMP

```
;extension=php_pdo_firebird.dll  
;extension=php_pdo_mssql.dll  
extension=php_pdo_mysql.dll  
;extension=php_pdo_oci.dll  
;extension=php_pdo_odbc.dll  
;extension=php_pdo_pgsql.dll  
extension=php_pdo_sqlite.dll
```

Nom du driver	Bases de données supportées
PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC et win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 et SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D

Vérification dans phpinfo()

PDO

PDO support	enabled
PDO drivers	mysql, sqlite, sqlite2

pdo_mysql

PDO Driver for MySQL, client library version	5.0.44
--	--------

Directive	Local Value	Master Value
pdo_mysql.connect_charset	no value	no value

pdo_sqlite

PDO Driver for SQLite 3.x	enabled
PECL Module version	(bundled) 1.0.1 \$Id: pdo_sqlite.c 293036 2010-01-03 09:23:27Z sebastian \$
SQLite Library	3.3.7

Création de la connexion

- Un nouvel objet : PDO
- Au moins 5 paramètres
 - Serveur, Port, BD, User, Password

```
<?php
$PARAM_hote='localhost'; // le chemin vers le serveur
$PARAM_port='3306';
$PARAM_nom_bd='sdz'; // le nom de votre base de données
$PARAM_utilisateur='root'; // nom d'utilisateur pour se connecter
$PARAM_mot_passe=''; // mot de passe de l'utilisateur pour se connecter
$connexion = new
PDO('mysql:host='.$PARAM_hote.';port='.$PARAM_port.';dbname='.$PARAM_nom_bd,
$PARAM_utilisateur, $PARAM_mot_passe);
?>
```



Faire mieux ?

- Gérer les erreurs
 - Gérer les exceptions
 - Try / Catch
- Préciser des paramètres
 - Charset par exemple

Code de connexion

```
// Connexion au serveur
try
{
    $dns = 'mysql:host=localhost;dbname=giraud';
    $utilisateur = 'giraud';
    $motDePasse = 'xxxxxx';

    // Options de connexion
    $options = array(
        PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION);

    $connection = new PDO( $dns, $utilisateur, $motDePasse, $options );
}
catch ( Exception $e )
{
    echo "Connexion à MySQL impossible : ", $e->getMessage();
    die();
}
```




Try, Throw, Catch

- Throw permet de faire générer des exceptions à une traitement (une fonction par exemple)
- Un bloc « try » appelle la fonction
- Un bloc « catch » récupère le code de l'exception éventuellement rencontrée

Exemple

```
<?php
//create function with an exception
function checkNum($number)
{
    if($number>1)
    {
        throw new Exception("Value must be 1 or below");
    }
    return true;
}
//trigger exception in a "try" block
try
{
    checkNum(2);
    //If the exception is thrown, this text will not be shown
    echo 'If you see this, the number is 1 or below';
}
//catch exception
catch(Exception $e)
{
    echo 'Message: ' . $e->getMessage();
}
?>
```



Requête

- Ce que l'on doit faire :
 - Définir la requête SQL et l'exécuter
 - Choisir le mode de fetch (tableau, objet ...)
 - Traiter le retour
 - Faire une boucle si nécessaire
- Et gérer les éventuelles exceptions ...

Requête

```
// Récupération des données
try {
    // On envoie la requête
    $select = $connection->query("SELECT * FROM minitp");
    // On indique que nous utiliserons les résultats en tant qu'objet
    $select->setFetchMode(PDO::FETCH_OBJ);
    // Nous traitons les résultats en boucle
    while( $enregistrement = $select->fetch() )
    {
        // Affichage des enregistrements
        echo '<h1>', $enregistrement->MINITP_NAME, ' ', $enregistrement->MINITP_MAIL, '</h1>';
    }
}
catch ( Exception $e )
{
    echo "Une erreur est survenue lors de la récupération des créateurs";
}
```

Query vs Exec

- Attention :
 - « query » ne s'utilise que pour récupérer des données
 - « exec » s'utilise pour toutes actions qui créent ou modifient des données
 - Insert, Update, Delete

```
❏<?php
    // on modifie le mot de passe de tous les utilisateurs (Juste pour l'exemple...)
    $nombre_changement=$connexion->exec("UPDATE membres SET mot_pass='toto'");

    // On affiche le nombre de lignes affectées par la demande
    echo "La requête à modifié : $nombre_changement lignes.";
?>
```

Fermer le curseur

- Il faut libérer les ressources utilisées par la requête
- C'est-à-dire :
 - Fermer le curseur qui traite les réponses

```
// Terminer la requête en fermant le curseur  
$select->closeCursor();
```



Requête préparée

- But :
 - Préparer une requête munie de marqueurs
 - Pour transmettre facilement des paramètres
 - Meilleure gestion serveur, car pré-compilation
 - Donc de l'optimisation
 - Surtout si une requête est appelée plusieurs fois
 - Améliorer la sécurité
 - Rendre plus difficile l'injection SQL



Requête préparée

- Passage de paramètres :
 - Par marqueurs nominatifs
 - Tableau associatif des paramètres
 - Par marqueur « ? »
 - Tableau scalaire des paramètres
- La première solution est à privilégier

Marqueurs nominatifs

```
echo "<h2>Préparation d'une requête avec marqueur nominatif</h2>";  
// on prépare la requête  
$requete_2 = $connexion->prepare("SELECT * FROM minitp WHERE MINITP_ID=:id");  
// on exécute la requête avec une première valeur  
$requete_2->execute(array("id"=>5));  
$ligne = $requete_2->fetch(PDO::FETCH_OBJ);  
echo "<div>5 = ".$ligne->MINITP_NAME."</div>";  
// on ré-exécute la requête avec une deuxième valeur  
$requete_2->execute(array("id"=>19));  
$ligne = $requete_2->fetch(PDO::FETCH_OBJ);  
echo "<div>19 = ".$ligne->MINITP_NAME."</div>";
```

Marqueurs « ? »

```
echo "<h2>Préparation d'une requête avec marqueur ?</h2>";  
// on prépare la requête  
$requete_3 = $connexion->prepare("SELECT * FROM minitp WHERE MINITP_ID=?");  
// on exécute la requête avec une première valeur  
$requete_3->execute(array(2));  
$ligne = $requete_3->fetch(PDO::FETCH_OBJ);  
echo "<div>2 = ".$ligne->MINITP_NAME."</div>";  
// on ré-exécute la requête avec une deuxième valeur  
$requete_3->execute(array(8));  
$ligne = $requete_3->fetch(PDO::FETCH_OBJ);  
echo "<div>8 = ".$ligne->MINITP_NAME."</div>";
```



Sécuriser une connexion PDO

- Sans utiliser les requêtes préparées
- On demande à l'objet PDO
 - De protéger les paramètres
 - En mettant les guillemets nécessaires

Sécuriser une connexion PDO

```
echo "<h2>Récupération des données</h2>";
// Récupération des données
try {
    // On fixe le critère
    $nom = "moi";
    // On envoie la requête
    $select = $connexion->query("SELECT * FROM minitp WHERE MINITP_NAME=".
                                $connexion->quote($nom, PDO::PARAM_STR));
    // On indique que nous utiliserons les résultats en tant qu'objet
    $select->setFetchMode(PDO::FETCH_OBJ);
    // Nous traitons les résultats en boucle
    $enregistrement = $select->fetch();
    // Affichage des enregistrements
    echo '<div>', $enregistrement->MINITP_NAME, ' ', $enregistrement->MINITP_MAIL, '</div>';
}
catch ( Exception $e )
{
    echo "Une erreur est survenue lors de la récupération des créateurs";
}
```

Sécuriser une connexion PDO

- Paramètres possibles :

Valeurs possibles pour le deuxième argument :

- `PDO::PARAM_STR` : pour une chaîne de caractères ;
- `PDO::PARAM_INT` : pour le type 'integer' de SQL ;
- `PDO::PARAM_NULL` : pour le type NULL de SQL ;
- `PDO::PARAM_BOOL` : pour un booléen ;
- `PDO::PARAM_LOB` : pour le type 'objet large' de SQL.

- Par défaut : `PARAM_STR`

- ATTENTION !

- Ne pas mettre les guillemets autour des chaînes

- C'est la méthode qui le fera



Bibliographie/**W**ebographie

- Certains contenus, concepts ou schémas de ce document sont extraits des sites et livres ci-dessous :
 - <http://www.w3schools.com/>
 - <http://www.siteduzero.com/>



Merci pour votre attention !

Philippe Giraud

Mail : philippe.giraud@enseignant-eemi.com