Introduction à PDO

- Il existe trois façons (extensions) de se connecter à une base de données MySQL via le langage PHP.
  - Extension Mysql, qui ne supporte pas les procédures stockées.
  - Extension Mysqli, qui supporte les procédures stockées. Cette méthode est très performante puisqu'elle est côté serveur. Elle peut s'utiliser avec n'importe quel langage de programmation. (un peu comme le driver Type 2 de JDBC)
  - Extension PDO, qui est propre au langage PHP. (un peu comme le driver de type 4 de JDBC)
- PDO est donc une API d'accès aux données (n'importe quel SGBD), orienté objet utilisé par PHP. Il est disponible avec les versions PHP 5 et plus.
- PDO présente plusieurs avantages (mis à part qu'il est objet) il s'utilise avec les procédures stockées et les requêtes paramétrées ce qui permet de se protéger contre les injections SQL. C'est la plus populaire pour utiliser PHP.

Introduction à PDO

- **Les classes PDO,** les classes PDO, PDOStatement et PDOExceptions sont les plus importantes:
- **PDO** : une instance de PDO représente la connexion à une base de données. Le plus souvent une seule instance de PDO par exécution de PHP.

Cette classe contient entre autre les méthodes exec(), query() et prepare().

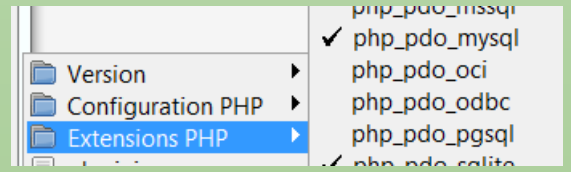
- **PDOStatement** : une instance de PDOStatement représente une requête vers la base. Permet de préparer la requête puis de consulter son résultat.

Cette classe contient entre autre les methodes fetch(), bindParmeter(), rowCount(),execute()

- **PDOException** pour la gestion des Exceptions.

Introduction à PDO

- Connexion à la base de données:
  - Vérifier que l'extension PDO est activée.



  - Établir la connexion avec les paramètres suivants
    - Nom du serveur
    - Nom de la bd
    - Le nom de l'usager
    - Le mot de passe.
  - Syntaxe:

$connexion = new PDO('mysql:host=$hote;dbname=$basededonnee;charset=utf8', $user, $mpasse);

  - Exemple

```
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', 'ruby2016');
```

Introduction à PDO

```
try
{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', 'ruby2016');
}
catch (PDOException $e)
{
echo('Erreur de connexion: ' . $e->getMessage());
        exit();
}
```

Le exit() peut être remplacé par die: **die('Erreur : ' . $e->getMessage())**

pour fermer la connexion:

$mybd=null;

Introduction à PDO

Exécution des requêtes sans paramètres:

- Requêtes DML, on utilise la méthode **PDO ::exec() :** Cette méthode permet d'envoyer une requête de type DML à la base de données. Elle prends comme paramètre un string qui est la requête à exécuter et retourne un entier indiquant le nombre de lignes affectées.

public int PDO::exec (string $statement)

- Requêtes SELECT, on utilise la méthode **PDO ::query()** Cette méthode permet d'envoyer une requête SELECT sans paramètres à la base de données. Elle retourne PDOStatement (un curseur). On utilise la méthode fetch() pour lire le jeu de résultat obtenu.

public PDOStatement PDO::query (string $statement)

Introduction à PDO

```
<?php
try
{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', ‘ruby', ''‘ruby2016);
$insertion = $mybd->exec("INSERT INTO chats(nom, race) VALUES ('ram','domestique')");
 echo('total insertion est ' . $insertion);
$suppresion = $mybd->exec("delete from chats where race =‘bengale'");
  echo('total suppression est ' . $suppresion);
  }
   catch (PDOException $e)
{  echo('Erreur de connexion: ' . $e->getMessage()); exit(); }
$mybd=null;
?>
```

Introduction à PDO

```
<?php
try
{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', 'ruby2016');
$resultat = $mybd->query("SELECT * FROM CHATS limit 5");
   while ($donnees = $resultat->fetch())
                {
                echo $donnees[1] . $donnees[2] .'<br />';
                }
echo ($resultat->rowCount());// nombre totale de lignes
$resultat->closeCursor();
}
  catch (PDOException $e)
{     echo('Erreur de connexion: ' . $e->getMessage()); exit(); }
$mybd=null;
?>
```

Introduction à PDO

La commande FETCH (de PDOStatement) permet de lire ligne par ligne le contenu du curseur. À chaque fois que cette commande est appelée, le curseur avance au prochain enregistrement dans l'ensemble actif.

Le résultat de la lecture est envoyé dans un tableau de nom donnees (pour notre exemple).

Par défaut, (le type de fetch n'est pas défini) la lecture du tableau de données se fait par l'indice de colonne. L'indice de la première colonne est **ZÉRO.**

Parfois, il est nécessaire de dire à votre programme comment vous voulez lire vos données. Comme par exemple accéder au tableau par le nom des colonnes.

- PDO::FETCH_ASSOC: retourne un tableau indexé par le nom des colonnes;
- PDO::FETCH_OBJ: retourne un objet dont les propriétés correspondent aux nom de colonnes.

Introduction à PDO

```
<?php
try
{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', ‘ruby2016');
$resultat = $mybd->query("SELECT * FROM CHATS limit 8", PDO::FETCH_ASSOC)
while ($donnees = $resultat->fetch())
                {
  echo $donnees['nom'] . $donnees['race'] .'<br />';
                }
echo ($resultat->rowCount());// nombre totale de lignes
$resultat->closeCursor();
}
        catch (PDOException $e)
{echo('Erreur de connexion: ' . $e->getMessage()); exit();}
$mybd=null;
?>
```

Introduction à PDO

```
<?php
try
{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', ‘ruby', ‘ruby2016');
    echo('connexion reussie') . '<br />';
   echo ('affichage du contenu de la table chats') . '<br />';
    $resultat = $mybd->query("SELECT * FROM CHATS limit 3");
   $resultat->setFetchMode(PDO::FETCH_OBJ);
    while ($donnees = $resultat->fetch())
                    {
                    echo $donnees->nom . $donnees->race .'<br />';
                    }
echo ($resultat->rowCount());// nombre totale de lignes
$resultat->closeCursor();
}
//suite du code catch et fermeture de la connexion
```

Introduction à PDO

Exécution des requêtes avec paramètres:

Les requêtes paramétrées sont incontournables lorsque nous essayons d'accéder à une base de données via le web.

Elles ont un avantage majeur qui est de réduire considérablement les injections SQL. De plus ce sont des requêtes préparées donc exécutées plus d'une fois et précompilées.

- En PHP les paramètres peuvent être représentés par le **?** Ou **:nomparametre**
- La methode execute() du PDOStatement permet d'executer des requêtes paramétrées.

prepare → bind → execute

Introduction à PDO

La méthode **prepare() :** cette méthode de la classe PDO permet de passer une requête paramétrée au SGBD.

Public PDOStatement PDO::prepare (string $statement).

La méthode **bindParam()** de la classe PDOStatement permet de lier les variables aux valeurs.

public bool PDOStatement::bindParam ($parameter , &$variable [, int $data_type = PDO::PARAM_STR [, int $length]])

Exemples:

$stmt1->bindParam(1, $nom);

$stm->bindParam(2, $race, PDO::PARAM_STR);

$stm->bindParam(2, $race, PDO::PARAM_STR,20);

La méthode **execute(),** La méthode execute() de la classe PDOStatement permet d'exécuter la requête avec ses paramètres. Elle retourne True ou false selon le succès de l'exécution de la requête:

public bool PDOStatement::execute ([array $input_parameters])

Introduction à PDO

```
Try {
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', 'ruby2016');
$stmt1 = $mybd->prepare("INSERT INTO chats(nom, race) VALUES (?, ?)");
$stmt1->bindParam(1, $nom);
$stmt1->bindParam(2, $race);
// affectation de valeurs aux paramètres
$nom = 'Beau';
$race = 'Chat';
// executer la requête
$total= $stmt1->execute();
echo('total insertion est ' . $total);
  }
  catch (PDOException $e)
{echo('Erreur de connexion: ' . $e->getMessage()); exit();  }
 $mybd=null;
```

Introduction à PDO

```
try{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', 'ruby2016');
$stmt1 = $mybd->prepare("select * from chats where race  like ? ;)");
// Liaison des paramètres
$stmt1->bindParam(1, $race, PDO::PARAM_STR, 20);
Affectation des valeurs aux paramètres
$race = 'Abyssin';
// executer la requête
 $stmt1->execute();
   while ($donnees = $stmt1->fetch())
{  echo $donnees[1] . $donnees[2] .'<br />';  }
echo ($stmt1->rowCount());
$stmt1->closeCursor();
 }
// suite du code: Catch() et surtout fermer la connexion
```

Introduction à PDO

Remarque : les paramètres peuvent être représentés par :nomprametre au lieu du ?

Exemple:

```
$stmt1 = $mybd->prepare("INSERT INTO chats(nom, race) VALUES (:pnom,:prace)");

$nom = 'BeauBeau';

$race = 'domestique';

$stmt1->bindParam(':pnom', $nom);

$stmt1->bindParam(':prace', $race);

// executer la requête

$total= $stmt1->execute();
```

Introduction à PDO

Appel de procédures et fonctions:

- Lorsqu'on appelle une procédure stockée on procède de la même manière qu'une requête paramétrée. On utilise le PDO Statement avec le la méthode prepare() sauf qu'au lieu qu'elle prenne comme paramètre la requête, on utilise un CALL de la procédure.
- Il faudra, en plus de faire le bind des paramètres en IN, préciser le type des paramètres en OUT.

Introduction à PDO

Exemple1 : appel de procédure avec un paramètre en IN et dont le résultat est un ensemble d'enregistrements

```
delimiter |

CREATE  PROCEDURE ListeChats(in prace varchar(10))

Begin

select * from chats where race = prace;

end |
```

Introduction à PDO

```
<?php
try
{
$mybd = new PDO('mysql:host=localhost;dbname=exercice1;charset=utf8', 'ruby', 'ruby2016');
  $stm = $mybd->prepare("CALL ListeChats(?)", array(PDO::ATTR_CURSOR, PDO::CURSOR_FWDONLY));
  $race='Abyssin';
  $stm->bindParam(1, $race);
   $stm->execute();
while ($donnees = $stm->fetch())
                    {
                    echo $donnees[1] . '<br />';
                    }
$stm->closeCursor();
}
// suite du code: catch() et surtout fermer le BD.
```

Introduction à PDO

Consultez ceci:

http://salihayacoub.com/420Ke9/Mysql/Semaine1/Mysql1.pdf

Page 29.