7. Les droits d'accès et l'encapsulation

Exemple :

```
<?php
class Membre
{
   private $pseudo;
   private $mdp;
public function setPseudo($nouveauPseudo)
   {      $this->pseudo = $nouveauPseudo;   }
}
?>
```


Plan du cours

Chapitre II : Les transactions avec MySQL et PDO

1. **Introduction**
2. **Connexion à la base de donnée avec PDO**
3. **Exécuter des requêtes avec PDO**
4. **Les requêtes préparées**

Objectifs du chapitre II

1. Introduire PDO et sa relation avec MYSQL.
2. Expliquer les principes de PDO et sa relation avec la POO.
3. Savoir se connecter a une base de donnée MYSQL avec PDO.
4. Savoir utiliser les méthodes qui exécutent des requêtes SQL que fournit la classe PDO.
5. Maîtriser les méthodes avancées de la classe PDO et deduire leurs avantages.

INTERNATIONAL INSTITUTE OF TECHNOLOGY
North American Private University

Développement web avancé

1. Introduction

Définition :

PDO : Php Data Object

- C'est une classe PHP destinée à communiquer avec un serveur de base de données.
- PDO représente une couche d'abstraction, c'est-à-dire, il va permettre de communiquer avec la plupart des serveurs de base de données : MySQL, Oracle, SqlServer, etc...
- PDO va permettre de sécuriser les requêtes et de favoriser la réutilisation du code grâce aux requêtes préparées.

2. Connexion à la BD avec PDO

Se connecter à votre serveur SQL :

```
<?php
$cnx= new PDO('mysql:host=localhost;dbname=votre_base', 'utilisateur', 'mot_de_passe');
//ajouter cette instruction pour permettre l'affichage des messages d'erreurs
$cnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
?>
```


INTERNATIONAL INSTITUTE OF TECHNOLOGY
North American Private University

Développement web avancé

Chapitre II : Les transactions avec MySQL et PDO

2. Connexion à la BD avec PDO

Interprétation des exceptions PDO :

```
<?php
try {
  $connexion = new PDO('mysql:host=localhost;dbname=nom_db', 'root', '');
$connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
}
catch ( PDOException $e ) {
  echo "Connection à MySQL impossible : ", $e->getMessage();
  exit(); //ou die();
} ?>
```

3. Exécuter des requêtes avec PDO

PDO distingue dans un CRUD deux types de requêtes :

- ✓ les requêtes de sélection : SELECT => query($requete)
- ✓ les requêtes de modification insertion : UPDATE – INSERT – DELETE => exec($requete)

3. Exécuter des requêtes avec PDO

➢ **Exécuter une requête avec la méthode exec() :**

Cette méthode retourne que le nombre de lignes affectées.

EX :

```
<?php
$sql = "DELETE FROM Membre WHERE login= 'abc' ";
$nb = $pdo->exec($sql);
echo $nb.' membres ont été supprimés.';
 ?>
```

3. Exécuter des requêtes avec PDO

➢ **Exécuter des requêtes avec la méthode query() :**

Cette méthode retournera un jeu de résultats sous la forme d'un objet **PDOStatement**

```
<?php
$query = 'SELECT * FROM Membres WHERE id=1';
//retourne une seule ligne
$ligne = $pdo->query($query)->fetch();
-----------------------------------------------------------------
$query = 'SELECT * FROM Membres';
//retourne plusieurs lignes
$tab= $pdo->query($query)->fetchAll();
?>
```


3. Exécuter des requêtes avec PDO

Quelques alternatives de PDOStatement:

- *PDO::FETCH_NUM* : retourne un tableau indexé par le numéro de la colonne comme elle est retourné dans votre jeu de résultat, commençant à 0
- *PDO::FETCH_ASSOC*: retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats
- *PDO::FETCH_BOTH* (défaut): retourne un tableau indexé par les noms de colonnes et aussi par les numéros de colonnes, commençant à l'index 0, comme retournés dans le jeu de résultats
- *PDO::FETCH_OBJ* : retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats

3. Exécuter des requêtes avec PDO

Comment Attribuer les PDOStatement:

- *Deux méthodes sont possible pour définir le PDOStatement pour une requête :*

Méthode 1 :

```
$resultats=$cnx->query(''select * from membres'');
$resultats->setFetchMode(PDO::FETCH_OBJ);
while( $resultat = $resultats->fetch() ){ … }
```

Méthode 2 :

```
while( $resultat = $resultats->fetch(PDO::FETCH_OBJ) ){ … }
```


3. Exécuter des requêtes avec PDO

Récupérer le contenu d'une table :

Avec Fetch()

```
$resultats=$cnx->query(''select * from membres'');
$resultats->setFetchMode(PDO::FETCH_OBJ);
Echo ''<table>'';
while( $resultat = $resultats->fetch() ){
Echo ''<tr><td>''.$resultat->nom. ''</td><td>''.$resultat->prenom. ''</td></tr>'';
 }
Echo ''</table>'';
```

3. Exécuter des requêtes avec PDO

Récupérer le contenu d'une table :

Avec Fetch()

```
//Ecriture plus simplifié
Echo ''<table>'';
while( $resultat =$cnx->query(''select * from membres'')->fetch(PDO::FETCH_OBJ) )
{ Echo ''<tr><td>''.$resultat->nom. ''</td><td>''.$resultat->prenom. ''</td></tr>''; }
Echo ''</table>'';
```


3. Exécuter des requêtes avec PDO

Récupérer le contenu d'une table :

Avec FetchAll()

```
Echo ''<table>'';
foreach( $resultat =$cnx->query(''select * from membres'')->fetchAll(PDO::FETCH_OBJ) ):
Echo ''<tr><td>''.$resultat->nom. ''</td><td>''.$resultat->prenom. ''</td></tr>'';
Endforeach;
Echo ''</table>'';
```

3. Les requêtes préparées :

Définition et avantages des requêtes préparées :

- Une requête préparée est une sorte de modèle compilé pour le SQL que vous voulez exécuter, qui peut être personnalisé en utilisant des variables en guise de paramètres.
- La plupart des bases de données supportent le concept des requêtes préparées.
- La requête ne doit être analysée (ou préparée) qu'une seule fois, mais peut être exécutée plusieurs fois avec des paramètres identiques ou différents.
- En utilisant les requêtes préparées, vous évitez ainsi de répéter le cycle analyse/compilation/optimisation.
- les requêtes préparées utilisent moins de ressources et s'exécutent plus rapidement.
- les requêtes préparées fore la sécurité contre les injections SQL.

INTERNATIONAL INSTITUTE OF TECHNOLOGY
North American Private University

Développement web avancé

Chapitre II : Les transactions avec MySQL et PDO

3. Les requêtes préparées :

les marqueurs nommés et les marqueurs interrogatifs :

- La requête SQL peut contenir zéro ou plusieurs marqueurs nommés (*:nom*) ou marqueurs interrogatifs (*?*) pour lesquels les valeurs réelles seront substituées lorsque la requête sera exécutée.
- Vous ne pouvez pas utiliser les marqueurs nommés et les marqueurs interrogatifs dans une même requête SQL ; choisissez l'un ou l'autre.

marqueurs nommés :

Ex : $res = $cnx->prepare("INSERT INTO membres (login, mdp) VALUES (:login, :mdp)");

marqueurs interrogatifs :

Ex : $res = $cnx->prepare("INSERT INTO membres (login, mdp) VALUES (?, ?)");

3. Les requêtes préparées :

Affectation des valeurs :

$res->**bindParam**();

Cette méthode permet d'affecter les valeurs envoyées à la place des marqueurs.

marqueurs nommés :

$res->**bindParam**(':login', $login);

$res->**bindParam**(':mdp', $mdp);

marqueurs interrogatifs :

$res->**bindParam**(1, $login);

$res->**bindParam**(2, $mdp);

exécution de la requête préparée :

$res->execute();

INTERNATIONAL INSTITUTE OF TECHNOLOGY
North American Private University

Développement web avancé

Chapitre II : Les transactions avec MySQL et PDO

3. Les requêtes préparées :

Exemple #1 Insertions répétitives en utilisant les requêtes préparées (marqueurs nommés)

```
<?php
$res = $cnx->prepare("INSERT INTO membres (login, mdp) VALUES (:login, :mdp)");
$ res ->bindParam(': login ', $ login);
$ res ->bindParam(': mdp ', $ mdp);

// insertion d'une ligne
$ login  = 'abcd';
$ mdp  = 'azerty';
$ res ->execute();

// insertion d'une autre ligne avec des valeurs différentes
$ login  = 'xyz';
$ mdp  = '123';
$ res ->execute();
?>
```

3. Les requêtes préparées :

Exemple #2 Insertions répétitives en utilisant les requêtes préparées (marqueurs ?)

```
<?php
$res = $cnx->prepare("INSERT INTO membres (login, mdp) VALUES (?, ?)");
$ res ->bindParam(1, $ login);
$ res ->bindParam(2, $ mdp);

// insertion d'une ligne
$ login  = 'abcd';
$ mdp  = 'azerty';
$ res ->execute();

// insertion d'une autre ligne avec des valeurs différentes
$ login  = 'xyz';
$ mdp  = '123';
$ res ->execute();
?>
```


3. Les requêtes préparées :

Exemple #3 Insertions répétitives en utilisant les requêtes préparées (sans bindParam())

```
<?php

$res = $cnx->prepare("SELECT * FROM membres where login = ?");
if ($res->execute(array($_GET['login']))) {
  while ($row = $res->fetch()) {
    print_r($row);
  }
}

?>
```