Programmation orientée objet en PHP

PDO : PHP Data Object

PDO est une couche d'abstraction qui permet de travailler avec (presque) n'importe quel type de Service de Gestion de Base de Données. PDO définit un certain nombre de classes qui permettent d'effectuer des traitements de persistance en se reposant sur des drivers (ou pilotes) spécifiques à chaque fournisseur de SGBD. L'extension PDO et les drivers doivent être activés si l'on souhaite utiliser PDO.
Le travail avec les SGBD se fait de façon transparente et homogène : quelque soit le SGBD, on utilise les mêmes objets et méthodes fournis par PDO.
L'extension PDO propose trois classes principales :

- PDO : qui réalisera le travail de persistance
- PDOStatement : qui encapsulera les requêtes et leurs résultats
- PDOException : exception déclenchée par les méthodes de PDO et PDOStatement.

La connexion à la base de donnée se fait lors de l'instanciation d'un objet PDO. Le constructeur de PDO prendra en paramètre trois chaînes de caractères :

- la connexion ou data source : cette chaîne contiendra l'hôte et le nom du schéma
- l'utilisateur de la base
- le mot de passe de l'utilisateur.

Le constructeur pourra aussi prendra en paramètre un tableau contenant diverses options de connexion. En cas de problème de connexion à la base une PDOException est levée.

Nous allons nous connecter à une base de donnée MySQL : le schéma est "test_pdo", le nom de l'utilisateur est "root" et le mot de passe est "root" (l'utilisateur et le mot de passe sont peut-être différents pour vous).

```php
<?php

$dsn = 'mysql::host=localhost;dbname=test_pdo';
$username = 'root';
$passwd = 'root';
try {
    $conn = new PDO($dsn, $username, $passwd,
            [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
} catch (PDOException $ex) {
    echo $ex->getMessage();
}
```

Nous avons ajouté au constructeur un tableau avec une option de configuration. Les options sont indiquées sous forme de couples clé/valeur. La clé et la valeur utilisées ici sont des constantes de la classe PDO. Cette option indique que les erreurs rencontrées par les classes PDO devront être traitées sous formes d'exceptions.

La classe PDO possède deux méthodes pour effectuer des requêtes simples, ces deux méthodes prennent en paramètre une requête SQL sous forme de chaîne de caractères :

- exec() : cette méthode exécute une requête et renvoie le nombre d'enregistrements affectés
- query() : cette méthode exécute une requête et renvoie un objet de type PDOStatement qui encapsule le résultat de la requête

La classe PDOStatement possède deux méthodes pour récupérer les résultats de la requête :

- fetch() : renvoie les résultats ligne à ligne
- fetchAll() : renvoie tous les résultats dans un tableau.

Les méthodes fetch() et fetchAll() peuvent prendre en paramètre un fetch_style sous forme de constante qui détermine la façon dont PDO retourne les résultats :

- PDO::FETCH_ASSOC → retourne un tableau associatif indexé par le nom de la colonne, comme retourné dans le jeu de résultats
- PDO::FETCH_BOTH → retourne un tableau indexé par les noms de colonnes mais aussi par les numéros de colonnes (commençant à l'indice 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 retournées dans le jeu de résultats

Par défaut le fetch est de type PDO::FETCH_BOTH.
La classe PDOStatement possède aussi une méthode closeCursor() qui permet de libérer la connexion du serveur, permettant ainsi à d'autres requêtes SQL d'être exécutées,

Voici un exemple ce script PHP contenant du HTML et affichant les personnes extraites de la base de données :

```php
<?php
// index.php
$dsn = 'mysql::host=localhost;dbname=test_pdo';
$username = 'root';
$passwd = 'root';
try {
    $conn = new PDO($dsn, $username, $passwd,
            [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    $query = 'SELECT * FROM personne';
    $stmt = $conn->query($query);
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $ex) {
    header('Location: error_page.html');
}
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Les personnes</title>
    </head>
    <body>
        <table>
            <tr>
                <th>Id</th>
                <th>Nom</th>
                <th>Prénom</th>
            </tr>
            <?php foreach ($result as $personne): ?>
                <tr>
                    <td><?php echo $personne['id'] ?></td>
                    <td><?php echo $personne['nom'] ?></td>
                    <td><?php echo $personne['prenom'] ?></td>
                </tr>
            <?php endforeach; ?>
        </table>
    </body>
</html>
```

En cas d'exception, le script redirige vers une page d'erreur avec un code erreur HTTP (500).

errorPage.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Erreur</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>Oups...</div>
    </body>
</html>
```

Voici un autre exemple de script PHP affichant les personnes extraites de la base de données :

```php
<?php
// index.php
$dsn = 'mysql::host=localhost;dbname=test_pdo';
$username = 'root';
$passwd = 'root';
try {
    $conn = new PDO($dsn, $username, $passwd,
            [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    $query = 'SELECT * FROM personne';
    $stmt = $conn->query($query);
} catch (PDOException $ex) {
    header('Location: error_page.html');
}
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Les personnes</title>
    </head>
    <body>
        <table>
            <tr>
                <th>Id</th>
                <th>Nom</th>
                <th>Prénom</th>
            </tr>
            <?php while ($personne = $stmt->fetch(PDO::FETCH_ASSOC)): ?>
                <tr>
                    <td><?php echo $personne['id'] ?></td>
                    <td><?php echo $personne['nom'] ?></td>
                    <td><?php echo $personne['prenom'] ?></td>
                </tr>
            <?php endwhile; ?>
        </table>
    </body>
</html>
```

Nous allons maintenant ajouter une personne à notre base de donnée. Nous créons donc un formulaire de saisie et un script d'ajout à la base.

formulaire.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Ajout d'une personne</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <form action="add_personne.php" method="post">
            <label>Nom : <input type="text" id="nom" name="nom"></label><br>
            <label>Prénom :
                <input type="text" id="prenom" name="prenom">
            </label><br>
            <input type="submit" value="Ajouter">
        </form>
    </body>
</html>
```

```php
<?php
```

```php

$dsn = 'mysql::host=localhost;dbname=test_pdo';
$username = 'root';
$passwd = 'root';
try {
    $conn = new PDO($dsn, $username, $passwd,
            [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    $nom = $_POST['nom'];
    $prenom = $_POST['prenom'];
    $query = "INSERT INTO personne (nom, prenom) VALUES ('$nom', '$prenom')";
    $conn->exec($query);
    header('Location: index.php');
} catch (PDOException $ex) {
    header('Location: error_page.html');
}
```

Fin

NAVIGATION

Accueil
- Ma page
  - Pages du site
  - Mon profil
  - Cours actuel
    - POO PHP
      - Participants
      - Généralités
      - La programmation orientée objet : premiers pas
      - L'héritage
      - Les interfaces
      - Le typage
      - Les namespaces
      - Les exceptions
      - Les bases de données avec PDO
      - **PDO : PHP Data Object**
      - personne.sql
      - La classe MysqlDAO
      - Les requêtes préparées
      - T.P. MysqlDao
      - Les tests avec PHPUnit
      - Petite application version 2
      - Petitie application version 3
  - Mes cours

ADMINISTRATION

Administration du cours

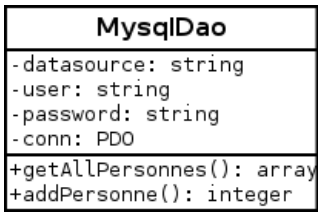
Réglages de mon profil

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP

Programmation orientée objet en PHP

La classe MysqlDAO

Nous allons implémenter une classe MysqlDAO afin d'implémenter le motif de conception DAO évoqué dans les leçons précedentes. Le rôle de cette classe va être de fournir des méthodes qui effectueront le travail de persistance. Cette classe aura pour responsabilité d'interagir avec le SGBD.
Cette classe possédera quatre attributs privés : la datasource, l'utilisateur et le mot de passe et la connexion à la base (instance de PDO), ainsi que deux méthodes publiques : getAllPersonnes() et addPersonne().




```php
<?php

// MysqlDao.php
class MysqlDao {

    private $datasource;
    private $user;
    private $password;
    private $conn;

    function __construct($datasource, $user, $password) {
        $this->datasource = $datasource;
        $this->user = $user;
        $this->password = $password;
        $this->conn = new PDO($datasource, $user, $password,
                [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    }

    public function getAllPersonnes() {
        $query = 'SELECT * FROM personne';
        $stmt = $this->conn->query($query);
        $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $result;
    }

    public function addPersonne($nom, $prenom) {
        $query = "INSERT INTO personne (nom, prenom) VALUES "
                . "('$nom', '$prenom')";
        $result = $this->conn->exec($query);
        return $result;
    }

}
```

Le constructeur ainsi que les deux méthodes sont susceptibles de propager une exception de type PDOException. Il sera donc nécessaire de gérer ces exceptions dans un bloc try/catch.
Le script index.php et add_personne.php vont donc maintenant utiliser la classe MysqlDAO :

```php
<?php
// index.php
include './MysqlDao.php';
$dsn = 'mysql::host=localhost;dbname=test_pdo';
$username = 'root';
$passwd = 'root';
try {
    $dao = new MysqlDao($dsn, $username, $passwd);
    $result = $dao->getAllPersonnes();
} catch (PDOException $ex) {
    header('Location: error_page.html');
}
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Les personnes</title>
    </head>
    <body>
        <table>
            <tr>
                <th>Id</th>
                <th>Nom</th>
                <th>Prénom</th>
            </tr>
            <?php foreach ($result as $personne): ?>
                <tr>
                    <td><?php echo $personne['id'] ?></td>
                    <td><?php echo $personne['nom'] ?></td>
                    <td><?php echo $personne['prenom'] ?></td>
                </tr>
            <?php endforeach; ?>
        </table>
    </body>
</html>
```

```php
<?php

// add_personne.php
include './MysqlDao.php';
$dsn = 'mysql::host=localhost;dbname=test_pdo';
$username = 'root';
$passwd = 'root';
try {
    $dao = new MysqlDao($dsn, $username, $passwd);
    $nom = $_POST['nom'];
    $prenom = $_POST['prenom'];
    $dao->addPersonne($nom, $prenom);
    header('Location: index.php');
} catch (PDOException $ex) {
    header('Location: error_page.html');
}
```

Fin

NAVIGATION

Accueil
- Ma page

Pages du site
Mon profil
Cours actuel
POO PHP

ADMINISTRATION

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP

Programmation orientée objet en PHP

Les requêtes préparées

Les requêtes préparées sont des requêtes paramétrables, pré-compilées et stockées par le SGBD en vue d'une utilisation ultérieure. La requête peut ainsi être exécutée plusieurs fois, avec des paramètres différents, sans être re-compilée. Les paramètres passés à la requête sont "sécurisés", assurant ainsi une protection contre les injections SQL.
La méthode prepare() de PDO permet de créer une requête préparée. Cette méthode renvoie un objet de type PDOStatement.
La méthode prepare() prendra en paramètre une chaîne de caractères représentant le requête SQL. Cette requête pourra contenir des paramètres nommés sous la forme :param. Ces paramètres pourront être valorisés lors de l'exécution de la requête ou, avant l'exécution de la requête, via les méthode bindParam() ou bindValue(). La méthode bindParam() permet de lier une variable à un paramètre de la requête, tandis que la méthode bindValue() permet de lier une valeur à une variable de la requête. Enfin, la méthode execute() de PDOStatement permet d'exécuter la requête.

Nous allons modifier les méthodes de la classe MysqlDao pour qu'elles utilisent des requêtes préparées au lieu de simples requêtes.

```php
<?php

// MysqlDao.php
class MysqlDao {

    private $datasource;
    private $user;
    private $password;
    private $conn;

    function __construct($datasource, $user, $password) {
        $this->datasource = $datasource;
        $this->user = $user;
        $this->password = $password;
        $this->conn = new PDO($datasource, $user, $password,
                [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    }

    public function getAllPersonnes() {
        $query = 'SELECT * FROM personne';
        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $result;
    }

    public function addPersonne($nom, $prenom) {
        $query = 'INSERT INTO personne (nom, prenom) VALUES '
                . '(:nom, :prenom)';
        $stmt = $this->conn->prepare($query);
        $result = $stmt->execute([':nom' => $nom, ':prenom' => $prenom]);
        return $result;
    }

}
```

Autre version de la méthode addPersonne() :

```php
<?php

// MysqlDao.php
class MysqlDao {

    private $datasource;
    private $user;
    private $password;
    private $conn;

    function __construct($datasource, $user, $password) {
        $this->datasource = $datasource;
        $this->user = $user;
        $this->password = $password;
        $this->conn = new PDO($datasource, $user, $password,
                [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    }

    public function getAllPersonnes() {
        $query = 'SELECT * FROM personne';
        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $result;
    }

    public function addPersonne($nom, $prenom) {
        $query = 'INSERT INTO personne (nom, prenom) VALUES '
                . '(:nom, :prenom)';
        $stmt = $this->conn->prepare($query);
        $stmt->bindValue(':nom', $nom);
        $stmt->bindValue(':prenom', $prenom);
        $result = $stmt->execute();
        return $result;
    }

}
```

Fin

NAVIGATION

Accueil
- Ma page
  - Pages du site
  - Mon profil
  - Cours actuel
    - POO PHP
      - Participants
      - Généralités
      - La programmation orientée objet : premiers pas
      - L'héritage
      - Les interfaces
      - Le typage
      - Les namespaces
      - Les exceptions
      - Les bases de données avec PDO
      - PDO : PHP Data Object
      - personne.sql
      - La classe MysqlDAO
      - **Les requêtes préparées**

T.P. MysqlDao
Les tests avec PHPUnit
Petite application version 2
Petitie application version 3
Mes cours

ADMINISTRATION

Administration du cours

Réglages de mon profil

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP

Programmation orientée objet en PHP

T.P. MysqlDao

Modifier la classe MysqlDao pour que la méthode addPersonne() prenne en argument un objet de type Personne (utiliser le type hinting). Si les attributs de l'objet Personne sont incorrects alors une exception avec un message "Personne invalide" sera jetée. Le nom et le prénom de la Personne ne doivent contenir que des lettres majuscules, minuscules ou des tirets. L'age de la Personne doit être un entier inclus entre 1 et 150.

Ajouter à la classe MysqlDao une méthode getPersonne() qui récupérera une Personne selon son id. Cette fonction prendra en argument l'id de la Personne à récupérer en base, et renverra un objet de type Personne.

Enfin créer une méthode updatePersonne() qui effectuera une mise à jour de la Personne. Cette méthode prendra en argument un objet de type Personne dont l'id sera l'id de la Personne à mettre à jour, le nom sera le nouveau nom de la Personne et le prénom le nouveau prénom de la Personne.

Vous testerez votre classe MysqlDao dans un lanceur.

Les fichiers seront à remettre dans une archive zip dont le nom sera de la forme "mysqldao_*nom_prenom*.zip".

État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Aucune tentative
Statut de l'évaluation	Pas évalué
Dernière modification	vendredi 27 mars 2015, 15:42

Ajouter un travail

Modifier votre travail remis

NAVIGATION

Accueil
- Ma page
  - Pages du site
  - Mon profil
  - Cours actuel
    - POO PHP
      - Participants
      - Généralités
      - La programmation orientée objet : premiers pas
      - L'héritage
      - Les interfaces
      - Le typage
      - Les namespaces
      - Les exceptions
      - Les bases de données avec PDO

ADMINISTRATION

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP