PDO : PHP Data Object

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

PHP

Plusieurs extensions possibles pour se connecter à une BD

- `mysql_`
- `mysqli_`
- `PDO`

PHP

Plusieurs extensions possibles pour se connecter à une BD

- `mysql_`
- `mysqli_`
- `PDO`

Quel choix ?

- `mysql_` et `mysqli_` sont utilisables seulement pour se connecter à une base de données **MySQL**
- `PDO` : une seule manière pour se connecter à n'importe quelle base de données (**Oracle**, **PostgreSQL**, **SQLITE**...)

PHP

PDO : PHP Data Object

- est une extension **PHP** pour accéder à n'importe quelle base de données
- fournit une interface d'abstraction pour l'accès aux données
- est une interface orientée objet

PHP

Avant de commencer, voici le script SQL qui permet de créer la base de données utilisée dans ce cours

```
CREATE DATABASE courspdo;

USE courspdo;

CREATE TABLE personne(
num INT PRIMARY KEY AUTO_INCREMENT,
nom VARCHAR(30),
prenom VARCHAR(30)
);

SHOW TABLES;

INSERT INTO personne (nom, prenom) VALUES ("Wick", "John"),
 ("Dalton", "Jack");

SELECT * FROM personne;
```

PHP

Pour activer **PDO** avec **WAMP**, il faut

- cliquer sur **WAMP**
- aller dans le menu **PHP** et choisir **Extensions PHP**
- cocher **pdo_mysql**
- attendre le redémarrage des services

PHP

Trois étapes

- Établir la connexion avec la base de données
- Créer et préparer des requêtes **SQL**
- Exécuter et récupérer le résultat de la requête

PHP

Syntaxe

```
<?php
maBase = new PDO (DSN, nomUtilisateur, motDePasse, [options]);
?>
```

PHP

Syntaxe

```
<?php
maBase = new PDO (DSN, nomUtilisateur, motDePasse, [options]);
?>
```


DSN : **D**ata **S**ource **N**ame

```
mysql:host=adresse;dbname=nomBD;port=numPort;charset=encodage
```

- host : localhost
- dbname : courspdo
- port : 3306 ou 3308
- charset : utf8

PHP

Exemple

```
$username = "root";
$password = '';
$dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
$db = new PDO($dsn, $username, $password);
```

PHP

Exemple

```
$username = "root";
$password = '';
$dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
$db = new PDO($dsn, $username, $password);
```


Une exception à capturer

Le constructeur de la classe `PDO` peut lancer une exception s'il ne peut se connecter à la base

PHP

Exemple

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Pour avoir des messages d'erreur précis, on ajoute un quatrième paramètre au constructeur de PDO

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password, array(PDO::ATTR_ERRMODE =>
    PDO::ERRMODE_EXCEPTION));
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Ou on peut l'ajouter avec le setter

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Pour exécuter une requête

- se connecter à la base de données
- utiliser l'objet connexion pour effectuer les requêtes

PHP

Pour exécuter une requête

- se connecter à la base de données
- utiliser l'objet connexion pour effectuer les requêtes


```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=coursphdo;port=3306;charset=utf8'
     ;
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

Afficher le résultat de la requête

- Dans `$req`, on a le résultat de l'exécution de la requête SQL
- Avec `fetch()` ou `fetchAll()`, on obtient le résultat sous forme d'un tableau

Afficher le résultat de la requête

- Dans `$req`, on a le résultat de l'exécution de la requête SQL
- Avec `fetch()` ou `fetchAll()`, on obtient le résultat sous forme d'un tableau

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  $res = $req->fetchAll();
  foreach ($res as $key => $value) {
    echo "$value[0] $value[1] $value[2] <br>";
  }
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Exemple avec `fetch`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  while ($tuple = $req->fetch()) {
    echo $tuple['num'] . " " . $tuple['nom'] . " ". $tuple['prenom']. "
      <br>";
  }
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Fermer le curseur pour exécuter une autre requête

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  while ($tuple = $req->fetch()) {
    echo $tuple['num'] . " " . $tuple['nom'] . " ". $tuple['prenom']. "
      <br>";
  }
  $req->closeCursor();
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

Pour terminer, on peut fermer la connexion

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  while ($tuple = $req->fetch()) {
     echo $tuple['num'] . " " . $tuple['nom'] . " ". $tuple['prenom'].
        "<br>";
  }
  $req->closeCursor();
  // pour fermer la connexion
  $db = null;
  // ou ainsi
  unset($db);
} catch (PDOException $e) {
    echo 'Erreur : ' . $e->getMessage();
}
```

Pour terminer, on peut fermer la connexion

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  while ($tuple = $req->fetch()) {
     echo $tuple['num'] . " " . $tuple['nom'] . " ". $tuple['prenom'].
        "<br>";
  }
  $req->closeCursor();
  // pour fermer la connexion
  $db = null;
  // ou ainsi
  unset($db);
} catch (PDOException $e) {
    echo 'Erreur : ' . $e->getMessage();
}
```

Si on ne le fait pas explicitement, **PHP** fermera automatiquement la connexion lorsque le script arrivera à la fin

PHP

Mode de récupération des données (paramètres de `fetch()`)

- `PDO::FETCH_BOTH` (par défaut) : retourne chaque ligne dans un tableau indexé par les noms des colonnes ainsi que leurs numéros, en commençant à 0.
- `PDO::FETCH_ASSOC` : retourne chaque ligne dans un tableau indexé par les noms des colonnes comme elles sont retournées dans le jeu de résultats correspondant.
- `PDO::FETCH_NUM` : retourne chaque ligne dans un tableau indexé par le numéro des colonnes en commençant de 0.
- `PDO::FETCH_OBJ` : retourne chaque ligne dans un objet avec les noms de propriétés correspondant aux noms des colonnes comme elles sont retournées dans le jeu de résultats.
- `PDO::FETCH_CLASS` ou `PDO::FETCH_CLASSTYPE` : retourne une nouvelle instance de la classe demandée, liant les colonnes aux propriétés nommées dans la classe.

PHP

Exemple avec `PDO::FETCH_NUM`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  $tuple = $req->setFetchMode(PDO::FETCH_NUM);
  while ($tuple = $req->fetch()) {
    echo $tuple[0] . " " . $tuple[1] . " ". $tuple[2]. "<br>"; $tuple['
      prenom']. "<br>";
  }
  $req->closeCursor();
  $db = null;
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

On peut préciser le mode directement dans `fetch`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  // $tuple = $req->setFetchMode(PDO::FETCH_NUM);
  while ($tuple = $req->fetch(PDO::FETCH_NUM)) {
    echo $tuple[0] . " " . $tuple[1] . " ". $tuple[2]. "<br>";
  }
  $req->closeCursor();
  $db = null;
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Exemple avec `PDO::FETCH_OBJ`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM personne');
  while ($tuple = $req->fetch(PDO::FETCH_OBJ)) {
    echo $tuple->num . " " . $tuple->nom . " " . $tuple->prenom . "<br>
      ";
  }
  $req->closeCursor();
  $db = null;
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Considérons la page `formulaire.php`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Formulaire</title>
</head>
<body>
        <form method='post' action='test.php'>
                <div>
                        Nom : <input type='text' name='nom'>
                </div>
                <div>
                        Prénom : <input type='text' name='prenom'>
                </div>
                <div>
                        <input type='submit' value='connexion'>
                </div>
        </form>
</body>
</html>
```

Le corps de la page `test.php`

```
try
{
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('SELECT * FROM Personne WHERE nom = "' . $_POST['
     nom'] . '" AND prenom = "' . $_POST['prenom'] . '"');
  if ($req->fetch()) {
    echo "Bonjour";
  } else {
    echo "Vous n'êtes pas inscrit";
  }
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
?>
```

Injection **SQL** : faille **XSS** (**C**ross-**S**ite **S**cripting)

- En saisissant un nom et un prénom qui existent dans la base, on peux se connecter.
- Mais on peut aussi se connecter en saisissant des valeurs inexistantes

Injection **SQL** : faille **XSS** (**C**ross-**S**ite **S**cripting)

- En saisissant un nom et un prénom qui existent dans la base, on peux se connecter.
- Mais on peut aussi se connecter en saisissant des valeurs inexistantes

Exemple

- Nom : un nom qui existe (`Wick`)
- Prénom : `xxx ?" or true !="`

Injection **SQL** : faille **XSS** (**C**ross-**S**ite **S**cripting)

- En saisissant un nom et un prénom qui existent dans la base, on peux se connecter.
- Mais on peut aussi se connecter en saisissant des valeurs inexistantes

Exemple

- Nom : un nom qui existe (`Wick`)
- Prénom : `xxx ?" or true !="`

Solution

Des fonctions spéciales ou les requêtes préparées

PHP

les fonctions spéciales

- `$nom_connexion->quote($nom_chaîne)` : désactive les quotes insérées par l'utilisateur
- `htmlspecialchars($nom_chaîne)` : élimine les < et >. Pour récupérer la chaîne d'origine on peut utiliser `htmlspecialchars_decode()`.
- `strip_tags($nom_chaîne)` : supprime les balises **HTML** et **PHP**
- `stripcslashes` : supprime les antislash d'une chaîne de caractère.
- `htmlentities($nom_chaîne)` : remplace les ', ", <, > par leur code **HTML** (&lt ; ...). Pour récupérer la chaîne d'origine on peut utiliser `html_entity_decode()`.
- `addcslashes()` : ajoute des slashs devant des caractères mentionnés d'une chaîne de caractère.

Une deuxième solution consiste à utiliser les requêtes préparées et le marqueur ?

```
try
{
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->prepare('SELECT * FROM Personne WHERE nom = ? and prenom
     = ?');
  $req->execute(array($_POST['nom'], $_POST['prenom']));
  if ($req->fetch()) {
    echo "Bonjour";
  } else {
    echo "Vous n'êtes pas inscrit";
  }
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

Une deuxième solution consiste à utiliser les requêtes préparées et le marqueur ?

```
try
{
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->prepare('SELECT * FROM Personne WHERE nom = ? and prenom
     = ?');
  $req->execute(array($_POST['nom'], $_POST['prenom']));
  if ($req->fetch()) {
    echo "Bonjour";
  } else {
    echo "Vous n'êtes pas inscrit";
  }
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

Attention à l'ordre des paramètres

Une troisième solution consiste à utiliser le marqueur nominatif :

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->prepare('SELECT * FROM Personne WHERE nom = :nom AND
     prenom = :prenom');
  $req->execute(array(
      'nom' => $_POST['nom'],
      'prenom' => $_POST['prenom']
  ));
  if ($req->fetch()) {
    echo "Bonjour";
  } else  {
    echo "Vous n'êtes pas inscrit";
  }
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

Une quatrième solution consiste à utiliser le marqueur nominatif : et la méthode `bindValue()`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->prepare('SELECT * FROM Personne WHERE nom = :nom AND
    prenom = :prenom');
  $req->bindValue(':nom', $_POST['nom']);
  $req->bindValue(':prenom', $_POST['prenom']);
  $req->execute();
  if ($req->fetch()) {
    echo "Bonjour";
  } else {
    echo "Vous n'êtes pas inscrit";
  }
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

Attention

- `bindValue()` copie la valeur
- `bindParam()` copie la référence

Limite de `bindValue()`

```
$nom = 'wick';
$prenom = 'john'
$req = $db->prepare('SELECT * FROM Personne WHERE nom = :nom AND prenom
    = :prenom');
$req->bindValue(':nom', $nom);
$req->bindValue(':prenom', $prenom);
$nom = 'travolta';
$req->execute();
// la requête sera exécutée avec les valeurs wick et john
```

Limite de `bindValue()`

```
$nom = 'wick';
$prenom = 'john'
$req = $db->prepare('SELECT * FROM Personne WHERE nom = :nom AND prenom
    = :prenom');
$req->bindValue(':nom', $nom);
$req->bindValue(':prenom', $prenom);
$nom = 'travolta';
$req->execute();
// la requête sera exécutée avec les valeurs wick et john
```

Exemple avec `bindParam()`

```
$nom = 'wick';
$prenom = 'john'
$req = $db->prepare('SELECT * FROM Personne WHERE nom = :nom AND prenom
    = :prenom');
$req->bindParam(':nom', $nom);
$req->bindParam(':prenom', $prenom);
$nom = 'travolta';
$req->execute();
// la requête sera exécutée avec les valeurs travolta et john
```

PHP

Le type de la valeur PHP ne dicte pas le type SQL, mais c'est le troisième paramètre (optionnel) de `bindValue`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->prepare('SELECT * FROM Personne WHERE nom = :nom AND
     prenom = :prenom');
  $req->bindValue(':nom', $_POST['nom'], PDO::PARAM_STR);
  $req->bindValue(':prenom', $_POST['prenom'], PDO::PARAM_STR);
  $req->execute();
  if ($req->fetch()) {
    echo "Bonjour";
  } else {
    echo "Vous n'êtes pas inscrit";
  }
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

Pour consulter la liste de constantes **PDO** spécifiant le type

https://www.php.net/manual/fr/pdo.constants.php

PHP

Insérer des données

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $nom = $_POST['nom'];
  $prenom = $_POST['prenom'];
  $res = $db->exec("INSERT INTO Personne (nom, prenom) VALUES('$nom','
     $prenom')");
  echo $res ? "tuple inséré" : "problème d'insertion";
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

Insérer des données

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $nom = $_POST['nom'];
  $prenom = $_POST['prenom'];
  $res = $db->exec("INSERT INTO Personne (nom, prenom) VALUES('$nom','
     $prenom')");
  echo $res ? "tuple inséré" : "problème d'insertion";
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

`$res` **contient le nombre de tuples insérés**

PHP

Insérer des données avec une requête préparée

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $nom = $_POST['nom'];
  $prenom = $_POST['prenom'];
  $req = $db->prepare("INSERT INTO Personne (nom, prenom) VALUES(:nom,:
     prenom)");
  $req->bindValue(":nom", $nom);
  $req->bindValue(":prenom", $prenom);
  $res = $req->execute();
  echo $res ? "tuple inséré" : "problème d'insertion";
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

Pour récupérer l'identifiant de la personne ajoutée

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $nom = $_POST['nom'];
  $prenom = $_POST['prenom'];
  $req = $db->prepare("INSERT INTO Personne (nom, prenom) VALUES(:nom,:
     prenom)");
  $req->bindValue(":nom", $nom);
  $req->bindValue(":prenom", $prenom);
  $res = $req->execute();
  $id = $db->lastInsertId();
  echo $res ? "tuple inséré avec identifiant $id" : "problème d'
     insertion";
  $db = null;
} catch (Exception $e) {
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

De même pour la suppression et la modification

PHP

Une transaction

- Un ensemble de requêtes SQL
- Exécutée en bloc (soit tout, soit rien)

PHP

Une transaction

- Un ensemble de requêtes SQL
- Exécutée en bloc (soit tout, soit rien)

Exemple : un virement bancaire

- Un retrait d'un premier compte
- Un versement vers un deuxième

PHP

Comment ça marche ?

- À la fin de chaque transaction, on peut soit valider (`commit`), soit annuler (`rollback`), même en cas de réussite de toutes les requêtes.
- Si une requête échoue, la transaction sera annulée et les changements ne seront pas validés.
- Par défaut, chaque requête est considérée comme une transaction : validé si la requête n'échoue pas.

PHP

Comment ça marche ?

- À la fin de chaque transaction, on peut soit valider (`commit`), soit annuler (`rollback`), même en cas de réussite de toutes les requêtes.
- Si une requête échoue, la transaction sera annulée et les changements ne seront pas validés.
- Par défaut, chaque requête est considérée comme une transaction : validé si la requête n'échoue pas.

Non nécessaire si elle concerne une seule requête

Transaction : syntaxe

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $db->beginTransaction();
  $nom = $_POST['nom'];
  $prenom = $_POST['prenom'];
  $req = $db->prepare("INSERT INTO Personne (nom, prenom) VALUES(:nom,:
    prenom)");
  $req->bindValue(":nom", $nom);
  $req->bindValue(":prenom", $prenom);
  $res = $req->execute();
  $id = $db->lastInsertId();
  $db->commit();
  echo $res ? "tuple inseré avec identifiant $id" : "problème d'
    insertion";
  $db = null;
} catch (Exception $e) {
  $db->rollback();
  echo ('Erreur : ' . $e->getMessage());
}
```

PHP

Créons une classe `Personne` (entité) dans un répertoire `models`

```php
class Personne
{
    private $num;
    private $nom;
    private $prenom;

    public function __construct(int $num, string $nom, string $prenom)
    {
        $this->setNum($num);
        $this->nom = $nom;
        $this->prenom = $prenom;
    }

    // + getters + setters

    public function __toString(): string
    {
        return $this->num . " " . $this->nom . " " . $this->prenom;
    }
}
```

PHP

Utilisation de la classe pour afficher les tuples

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('select * from Personne');
  while($res = $req->fetch(PDO::FETCH_NUM)) {
    $perso = new Personne($res[0], $res[1], $res[2]);
    echo $perso;
  }
  $db = null;
  unset($db);
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Utilisation de la classe pour afficher les tuples

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('select * from Personne');
  while($res = $req->fetch(PDO::FETCH_NUM)) {
    $perso = new Personne($res[0], $res[1], $res[2]);
    echo $perso;
  }
  $db = null;
  unset($db);
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

Il faut faire attention à l'ordre des attributs (et des colonnes)

PHP

Ajoutons la méthode `hydrate($tuple)` dans la classe `Personne`

```
public function __construct(array $tuple)
{
    if (count($tuple))
        $this->hydrate($tuple);
}

public function hydrate(array $tuple)
{
    foreach ($tuple as $key => $value) {
        // ucfirst() : uppercase first letter
        $method = 'set' . ucfirst($key);
        if (method_exists($this, $method)) {
            $this->$method($value);
        }
    }
}
```

PHP

Pour tester

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('select * from Personne');
  while($res = $req->fetch(PDO::FETCH_ASSOC)) {
    $perso = new Personne($res);
    echo $perso;
  }
  $db = null;
  unset($db);
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Deuxième solution avec un constructeur sans paramètre

```
public function __construct()
{
}
```

PHP

Pour tester, on utilise le mode `PDO::FETCH_CLASS`

```
try {
  $username = "root";
  $password = '';
  $dsn = 'mysql:host=localhost;dbname=courspdo;port=3306;charset=utf8';
  $db = new PDO($dsn, $username, $password);
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $req = $db->query('select * from Personne');
  $personnes = $req->fetchAll(PDO::FETCH_CLASS, 'Personne');
  foreach ($personnes as $personne) {
    echo $personne->getNom() . "<br>";
  }
  $db = null;
  unset($db);
} catch (PDOException $e) {
  echo 'Erreur : ' . $e->getMessage();
}
```

PHP

Organisation du code

- Il faut mettre toutes les données (url, nomUtilisateur, motDePasse...) relatives à notre connexion dans une classe connexion
- Pour chaque table de la base de données, on crée une entité ayant comme attributs les colonnes de cette table
- Il faut mettre tout le code correspondant à l'accès aux données (de la base de données) dans des nouvelles classes "Manager" (qui constitueront la couche DAO : Data Access Object)

La classe `Connection`

```
class Connection
{
  private static $db = null;
  private static $instance = null;
  const DB_USER = 'root';
  const DB_PASSWORD = '';
  const DB_HOST = 'localhost';
  const DB_NAME = 'courspdo';
  const DBMS_PORT = 3308;

  private function __construct()
  {
     try {
       $dsn = 'mysql:dbname=' . self::DB_NAME . ';host=' . self::
          DB_HOST . ';port=' . self::DBMS_PORT;
       self::$db = new PDO($dsn, self::DB_USER, self::DB_PASSWORD);
       self::$db->setAttribute( PDO::ATTR_ERRMODE, PDO::
          ERRMODE_EXCEPTION);
     } catch (PDOException $e) {
        echo 'Erreur : ' . $e->getMessage();
     }
  }
```

PHP

La classe `Connection` (suite)

```
  public static function getInstance()
  {
    if (self::$instance == null) {
      self::$instance = new Connection();
    }
    return self::$db;
  }
}
```

PHP

La classe `Personne`

```
class Personne
{
    private $num;
    private $nom;
    private $prenom;

    public function __construct()
    {
    }

    // + getters + setters + __toString

}
```

PHP

La classe `PersonneManager` définie dans un répertoire `dao`

```php
class PersonneManager
{
    private $db = null;

    public function __construct()
    {
        $this->db = Connection::getInstance();
    }

    public function save(Personne $personne): int
    {
        $req = $this->db->prepare('INSERT INTO personne (nom, prenom)
           VALUES (:nom, :prenom)');
        $req->bindValue(':nom', $personne->getNom(), PDO::PARAM_STR);
        $req->bindValue(':prenom', $personne->getPrenom(), PDO::
           PARAM_STR);
        $req->execute();
        return $this->db->lastInsertId();
    }
    // + les autres méthodes
}
```

PHP

Pour tester toutes ces classes dans `test.php`

```
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];

$dao = new PersonneManager();

$personne = new Personne();
$personne->setNom($nom):
$personne->setPrenom($prenom):

$res = $dao->save($personne);
echo $res ? "tuple inséré avec identifiant $res" : "
   problème d'insertion";
```

PHP

Remarque

N'oubliez pas d'implémenter les quatre autres méthodes de la classe `PersonneManager`

- `findAll()`
- `findById()`
- `update()`
- `remove()`