



PDO : Interface d'accès aux BDD





Introduction

1. Présentation
2. mysqli_
3. PDO
 - a. connexion
 - b. requête
4. requête vs requête préparée

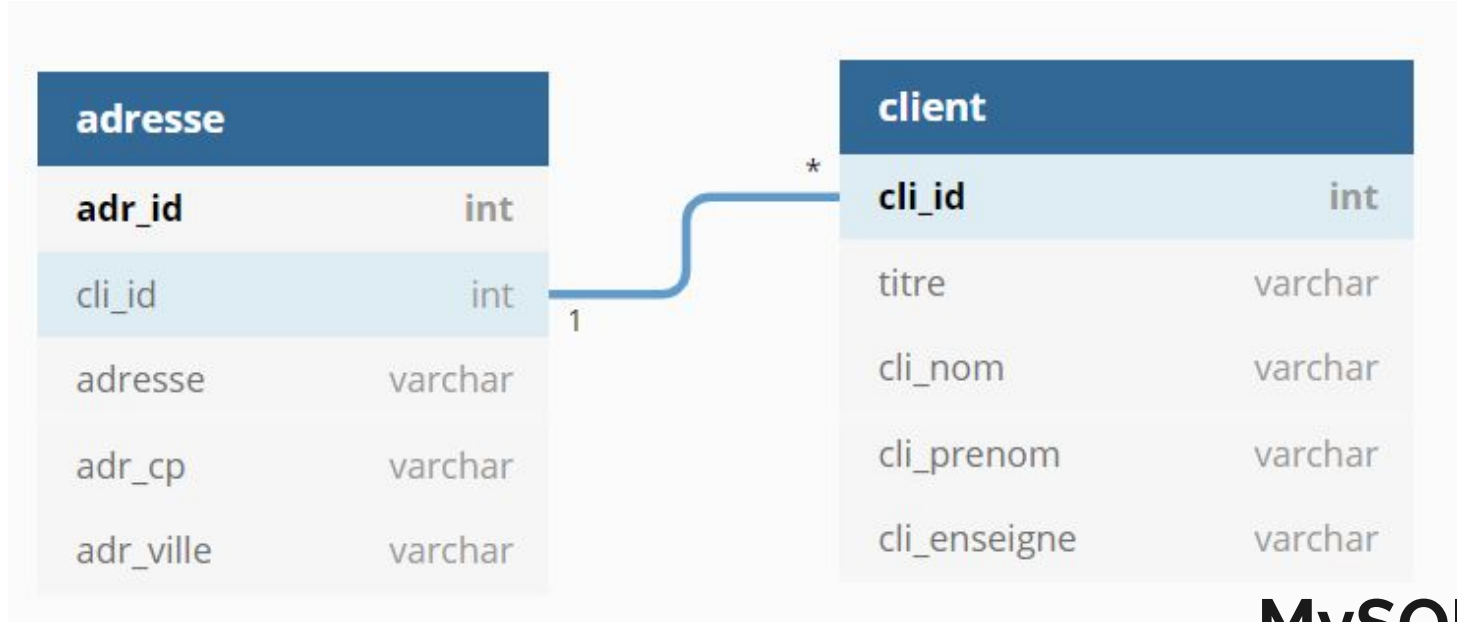


Présentation

PDO : (PHP Data Objects) l'accès aux BDD en faisant abstraction du moteur de SGBD utilisé

- Abstraction du moteur SGBD utilisé
- Orienté Objet (modification du comportement initiale)
- Gestion Exception


Schéma utilisé pour la suite



MySQL



Connexion (sans gestion d'erreur)



```
define('DB_HOST', 'localhost'); //define('DB_HOST', 'mysql:host=localhost;dbname=store;charset=utf8');
define('DB_PORT', '3306');
define('DB_DATABASE', 'bd_client');
define('DB_USERNAME', 'u_db_client');
define('DB_PASSWORD', '123456');

$database = new PDO('mysql:host=' . DB_HOST . ';port' . DB_PORT . ';dbname=' . DB_DATABASE, DB_USERNAME,
DB_PASSWORD);
```

Création d'une instance de la classe PDO avec les paramètres de connexion définis, le new appelle le constructeur de la classe PDO.

!! Il manque un "=" après ";port"

Connexion (levage d'une exception)

Si la base n'existe pas ou si nous n'avons pas les bonnes données de connexion à la SGBD, nous collectons une erreur tout en **ORANGE DE LA MORT** !!!!!!!!

! Fatal error: Uncaught PDOException: SQLSTATE[HY000] [1045] Accès refusé pour l'utilisateur: 'u_db_client'@'localhost' (mot de passe: OUI) in C:\wamp64\www\class\personnage.class.php on line 77				
! PDOException: SQLSTATE[HY000] [1045] Accès refusé pour l'utilisateur: 'u_db_client'@'localhost' (mot de passe: OUI) in C:\wamp64\www\class\personnage.class.php on line 77				
Call Stack				
#	Time	Memory	Function	Location
1	0.0301	417648	{main}()	...personnage.class.php:0
2	0.0318	418400	<u>__construct</u> ()	...personnage.class.php:77

Cette erreur peut, n'est pas très graphique, nous indique l'utilisateur utiliser, les lignes où les erreurs sont présentes ...




Connexion (avec gestion d'erreur)

C'est pour celà, que PHP 5 à créer une façon de gérer les erreurs, et nous nous retrouvons avec ceci.

```
ERREUR : SQLSTATE[HY000] [1045] Accès refusé pour l'utilisateur: 'u_db_client'@'localhost' (mot de passe: OUI)  
ERREUR : 1045
```

Connexion (avec gestion d'erreur)



```
try
{
    $database = new PDO('mysql:host='.DB_HOST.';port'.DB_PORT.';dbname='.DB_DATABASE, DB_USERNAME,
DB_PASSWORD);
    // $database = new PDO('mysql:host='.DB_HOST.';port'.DB_PORT.';dbname='.DB_DATABASE, DB_USERNAME,
DB_PASSWORD, array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));

    // Non obligatoire
    $database->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); // rapport d'erreur :emet des
exceptions
    $database->exec("SET NAMES 'utf8'"); // Codage utilisé
}
catch(Exception $e) //interception de l'erreur
{
    echo 'Erreur : '.$e->getMessage(). '<br />';
    echo 'Numéro : '.$e->getCode();
    exit();
}
```




Connexion

La gestion des exceptions doit être gérée avec **try** et **catch**, tout code qui peut générer des exceptions doit être placé entre dans le **try**, une exception sera attrapée et traitée dans le **catch**.



```
try
{
    // instruction ...
}
catch(Exception $e) //interception de l'erreur
{
    // gestion de l'exception
}
```

[Pour en savoir plus sur la gestion des exceptions de PHP](#)



2 types de requête

- **query()**

La méthode `query()` pour l'interrogation de données, dès que vous utilisez le `SELECT`.

- **exec()**

La méthode `exec()` pour la manipulation de données, dès que vous utilisez le `UPDATE`, `INSERT` ou `DELETE`

Exécuter une requête

Exec(), on souhaite modifier l'adresse du client 13

```
include 'config.BD.inc.php'; // Connexion à la BD

try
{
    $iNbLigne = $database->exec("UPDATE adresse SET adresse = '13 rue Voltaire' WHERE cli_id=13");
}
catch(Exception $e)
{
    die('Erreur : '.$e->getMessage(). '<br />');
}

echo $iNbLigne.' ligne(s) modifiée(s)';
```

Exécuter une requête



Exécution de la requête de modification; La méthode `exec()` retourne le nombre de lignes affectées par les requêtes de type DELETE, UPDATE ou INSERT.

Interception de l'éventuelle erreur est affiche de celle-ci, le die affiche le message et stoppe l'exécution du script Exemple: si la table invoquée dans la requête n'existe pas

Exécuter une requête

Query()

- Il existe plusieurs méthodes de récupération du résultat définies par les constantes PDO::FETCH_* est plusieurs méthodes pour le parcourir
- Dans notre exemple on utilise la méthode PDO::FETCH_ASSOC qui spécifie qu'une ligne du résultat sera traitée comme un tableau associatif;
- La méthode -> fetchAll() de la classe PDOStatement retourne l'intégralité du résultat dans un tableau

```
include 'config.BD.inc.php'; // Connexion à la BD

$query = 'SELECT cli_nom, cli_prenom, adr_ville
FROM client INNER JOIN adresse
ON client.cli_id = adresse.cli_id';

try {
    $results = $database->query($query);
}
catch(Exception $e)
{
    die('Erreur : '.$e->getMessage().'\n');
}

// Fetch
while($row = $results->fetch())
{
    echo $row['cli_nom'].'\n';
}

//FetchAll
$array = $results->fetchAll(PDO::FETCH_ASSOC);
foreach($array as $row)
    foreach($row as $value)
        echo $value.\n';
```

Exécuter une requête

Query()

- Il existe plusieurs méthodes de récupération du résultat définies par les constantes PDO::FETCH_* est plusieurs méthodes pour le parcourir
- Dans notre exemple on utilise la méthode PDO::FETCH_ASSOC qui spécifie qu'une ligne du résultat sera traitée comme un tableau associatif;
- La méthode -> fetchAll() de la classe PDOStatement retourne l'intégralité du résultat dans un tableau

```
Array
(
    [0] => Array
        (
            [cli_nom] => DREYFUS
            [cli_prenom] => Jean
            [adr_ville] => PARIS
        )
    [1] => Array
        (
            [cli_nom] => DUVAL
            [cli_prenom] => Arsène
            [adr_ville] => PARIS
        )
)
```

Il aura autant d'éléments que de lignes retournées par la requête

Exécuter une requête “Préparer”

Prepare()

La méthode `prepare()` offre la possibilité de préparer la requête avec des marqueurs qui seront substitués lors de l'exécution.

Il existe deux types de marqueurs :

- Le marqueur Interrogatif
- Le marqueur Nominatif



```
SELECT cli_nom FROM client WHERE cli_id = ? ;
```



```
SELECT cli_nom FROM client WHERE cli_id = :id ;
```

Exécuter une requête “Préparer”

Avantages

- Optimisation des perfs pour des requêtes appelées plusieurs fois
- Limité la bande passante utilisée entre le client et le serveur
- Protection contre les injections SQL

! Une attaque par injection SQL, consiste à injecter une requête SQL dans un paramètre. Si ce paramètre n'est pas contrôlé par l'application alors des actions non prévues peuvent être pilotées à distance par un pirate.



```
<?php
```

```
mysql_query("SELECT * FROM utilisateurs WHERE login = '$login'");
```

```
?>
```

```
// Code Injecté : toto'; DROP TABLE utilisateurs; #
```

```
SELECT * FROM utilisateurs WHERE login = 'toto'; DROP TABLE utilisateurs; #
```


Exécuter une requête “Préparer”

```
$firstName = 'Hector';  
$lastName = 'Sinapi';  
  
try {  
    $sql = $database->prepare('SELECT cli_nom FROM client WHERE cli_prenom = ? AND cli_enseigne = ?');  
    $sql->execute(array($firstName, $lastName));  
}  
catch (Exception $e) {  
    die('ERREUR : '.$e->getMessage());  
}  
  
while($iNbLigne = $sql->fetch(PDO::FETCH_NUM))  
    echo $iNbLigne[0].' ' '.$iNbLigne[1].'<br />';
```

Les ? seront substitués par les valeurs passées lors de l'appelle de la méthode execute()

Exécuter une requête “Préparer”

```

$firstName = 'Hector';
$lastName = 'Sinapi';

try {

    $sql = $database->prepare('SELECT cli_nom, cli_prenom FROM client WHERE cli_prenom = :prenom AND cli_nom
= :nom');
    $sql->execute(array('nom'=>$lastName, 'prenom'=>$firstName));

}
catch (Exception $e) {
    die('ERREUR : '.$e->getMessage());
}

while($iNbLigne = $sql->fetch(PDO::FETCH_OBJ))
    echo $iNbLigne->cli_nom.' '.$iNbLigne->cli_prenom.'<br />';

```

Les **.****** seront substitués par les valeurs passées lors de l'appelle de la méthode execute()