PDO - Accès à une base de donnée MySQL en PHP

PDO est un API de PHP qui permet d'accèder à des base de donnée MySQL, PostGre, Oracle ou encore SQLite.
https://www.php.net/manual/en/book.pdo.php

Pré-requis

- le module php `mysqli` installé et activé dans le php.ini
- le module php `pdo_mysql` installé et activé dans le php.ini
- le php.ini se trouve au même endroit que php.exe

Connexion

```php
$dbname = "bdd_name";
$user = "root";
$pass = "root";
$bdd = new PDO('mysql:host=localhost;dbname=$dbname', $user, $pass);
```

> Si rien de s'affiche sur la page, tout fonctionne bien.

Les 3 classes de PDO

Dans PDO il existe trois classes :

- `PDO` , la classe qui permet d'instancier une connexion à la BDD, `new PDO()` . Elle permet de lancer des requête au serveur et renvoi des objet de la classe PDOStatement.
- `PDOStatement` , la classe qui contient le résultat et contient des méthodes pour naviguer dans les lignes du tableau de résultats renvoyé par mysql. La fonction la plsu importante est `PDOStatement::fetchAll()` qui permet de récupérer tout les résultats dans un array PHP.
- `PDOExeception` , la classe qui représente les erreurs de PDO (status code 500).

Envoyer une requête et récupérer un array

La fonction `PDOStatement::fetchAll()` permet de récpuérer un array.

```
$name = "Jérome";
$stmt = $bdd->query("SELECT * FROM Product");
$products = $stmt->fetchAll();
```

Envoyer une requête et récupérer un élement

La fonction `PDOStatement::fetch()` permet de récpuérer le première element de l'array. A chaque appel de la fonction on récupére l'élement suivant. Très pratique dans une boucle for lorsque l'on ne veux pas lire tout les élements.

```
$name = "Jérome";
$stmt = $bdd->query("SELECT * FROM Product WHERE name=$name");
$user = $stmt->fetch();
```

La gestion d'erreur avec try..catch

En PHP si une erreur arrive on veut parfois la récupérer pour y réagir. Pour faire ceci on utilise `try{}catch{}` .

Syntaxe:

```
try{
    // Code risqué
}catch(Exeception $error){
    echo $error->getMessage();
}
```

Avec PDO on veut toujouts tester si la connexion à la PDO est réussi.

```php
try{
    $dbname = "bdd_name";
    $user = "root";
    $pass = "root";
    $bdd = new PDO('mysql:host=localhost;dbname=$dbname', $user, $pass);
}catch(PDOExeception $error){
    echo $error->getMessage();
}
```

Les requêtes SQL préparées

Jusqu'à maintenant quand on écriait une requête on utilisait la fonction query et si besoin on concatènait une variable pour faire un WHERE par exemple.
Seulement voilà si cette variable provient d'une autre source comme le client (variables POST ou GET) lors de la concaténation je risque une injection de code malicieux dans ma BDD.
Voilà pourquoi lorsque j'ai des paramètres dans ma requête j'utilise une requête dite "préparée".

Etapes d'envoi d'une requête préparée.

L'envoi d'une requête préparé se fait en 2 étapes :

- `PDO::prepare()` qui va préparer la requête SQL et lui fournir les paramètres à utiliser pour les WHERE, etc ...
- `PDOStatement::execute` qui execute la requête et demande les paramètres.

Préparation de la requête

Syntaxe :
Il faut placer des points d'intérogations là où souhaitez mettre vos paramètres.

```php
$requete = $dbh->prepare('REQUETE ? SUITE ? SUITE > ?');
$requete->execute([param1,param2,param3 ]);
```

Puis executer les requêtes en fournissant les paramètres dans un `array` , dans l'ordre l'apparitions des points d'intérogation.
Exemple :

```php
$request = $dbh->prepare('SELECT * FROM Produit WHERE price < ? AND stock > ?');
$request->execute([99.99, 20]);
// La requete finale est
// SELECT * FROM Produit WHERE price < 99.99 AND stock > 20;
```

> **Note : Requête réutilisable**
> L'avantage des requêtes préparées, c'est la réutilisation de l'objet $request. La requête se relancera à chaque appel de `PDOStatement::execute` .

Les transactions SQL

Les transactions SQL permettent d'annuler les requêtes appliquées à la base de données. Les transactions sont incluses dans l'objet `PDO` et il est très pratique de couplées avec un `try catch` .

Les transactions sont des méthodes de l'objet `PDO` :

- `PDO::beginTransaction` : https://www.php.net/manual/en/pdo.begintransaction.php
- `PDO::commit` : https://www.php.net/manual/en/pdo.commit.php
- `PDO::rollBack` : https://www.php.net/manual/en/pdo.rollback.php
- `PDO::inTransaction` : https://www.php.net/manual/en/pdo.intransaction.php , cette méthode renvoie vrai si une transaction est en cours.

Exemple :

```php
try {
    // Je démarre ma transaction...
    $dbh->beginTransaction();

    $dbh->query("INSERT INTO employe (name, lastname) VALUES ('Jeff', 'Bezos')");
    // Si tout c'est bien passé...je peux commit
    $dbh->commit();

} catch (Exception $e) {
    // Un problème ! Vite j'annule avec rollback !
    $dbh->rollBack();
    echo "Erreur: " . $e->getMessage();
}
```