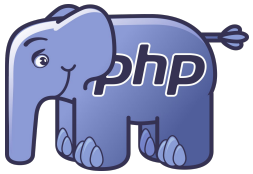


PHP BTS SIO - SQY

Romain Jalabert - 27.01.2023 - IPSSI

Planning de la journée

- Codewar le retour !
- Retour sur les cookies et Sessions
- Exercice sur les session (login et signup) + corrigé

- PDO introduction
- PDO exercices
- PDO TP

P’tit Challenge

Afficher sa propre adresse IP à l’aide d’un script PHP

(n’hésitez pas à aidez vous de Gogole !)

Codewars !!

- Reversed Words (8 kyu)
- Especially Joyful Numbers (7 kyu)
- You’re a square ! (7 kyu)
- Who likes it ? (6 kyu)

PDO : PHP DATA OBJECT

PDO : PHP Data Object

PDO est une interface d'abstraction qui permet de se connecter à une base de données. L'avantage est que PDO fonctionne avec de nombreux SGBD contrairement à mysqli.

On se connecte en créant une instance de la classe PDO avec comme paramètre le data source name :

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
?>
```

Une fois notre objet PDO créé on va pouvoir envoyer des requêtes SQL et communiquer avec la BDD

S'il y a des erreurs de connexion, on peut les récupérer via l'objet **PDOException**

Ressources : https://www.php.net/manual/fr/intro.pdo.php

Utilisation PDO avec try et catch

On essaie dans un premier temps de se connecter à la bdd avec **try** et on attrappe l'erreur s'il y en a une dans le bloc **catch.** On stoppe aussi l'exécution du script dans ce dernier cas à l'aide de **die("erreur")**

```php
try {

    // DSN - database ou data source name - de connexion
    $dsn = "mysql:dbname=test;host=localhost";

    $options = array(PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC);

    // On vient instancier PDO
    $db = new PDO($dsn, "root", "", $options);

    echo "Nous sommes bien connectés";

} catch (PDOException $error) {
    die("erreur : ".$error->getMessage());
}
```

Ressources : https://www.php.net/manual/fr/language.exceptions.php

PDO query() & exec()

On note de nombreuses méthodes associées à la classe PDO :

- **PDO::query** qui va exécuter les requêtes SQL de type SELECT et retourner le résultat dans l'objet PDOStatement**.**

- **PDO::exec** qui va exécuter les autres types de requêtes SQL. Cette méthode retourne un entier qui correspond aux nombres de lignes affectées par la requête.

Autres type de méthodes :

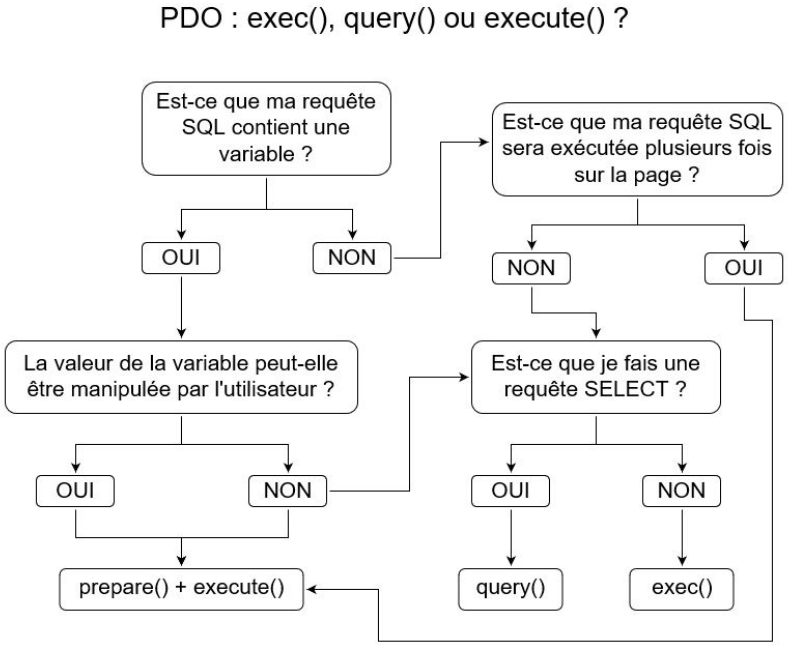
- Avec **PDO::FETCH_ASSOC** on retourne un tableau associatif comme résultat avec comme index le nom des colonnes (alors que par défaut FETCH_BOTH on a un tableau indexé par les noms de colonnes ET les numéros de colonnes)

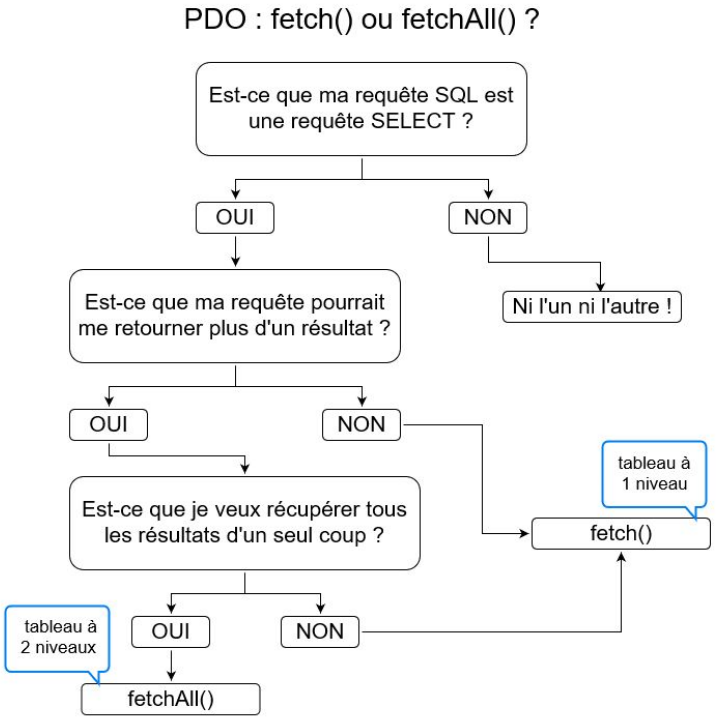
- Concernant PDOStatement, si j'utilise **PDOStatement::fetch** pour les requetes de type SELECT on va me retourner le résultat de ma requête ligne par ligne. Je vais obtenir un résultat à la fois.

- Avec **PDOStatement::fetchAll** je vais obtenir un tableau en retour contenant l'ensemble des résultats de ma requête.

Ressources : https://www.php.net/manual/fr/pdo.query.php

Schémas Récapitulatifs PDO





PDO Premiers Pas

Créer une instance de connexion avec PDO et se connecter à votre base de données dans un fichier db.php. Utiliser la méthode try / catch pour se faire. Afficher l'erreur si besoin. On voudra récupérer les infos sous forme de tableau associatif. On va importer la table authors dans phpmyadmin

Ensuite :

1) Afficher la liste des auteurs (table "authors") de la manière suivante :

   "$first_name $last_name : $email"

2) Afficher maintenant tous les auteurs qui ont moins de 40 ans (on se contentera à l'année près) de la façon suivante :

   "$first_name $last_name : $birthdate"

3) Ajouter un auteur du nom de "John Doe" à la table "authors" avec comme email : john.doe@gmail.com et comme birthdate : 1989-08-18.

   On affichera un message de confirmation si c'est réussi ('Ajout réussi !') sinon l'erreur en question.

4) Supprimer les auteurs avec un ID strictement supérieur à 100

   On affichera également un message de confirmation si c'est réussi ('Suppression réussie !') sinon l'erreur en question (pdo->errorInfo())

Challenge PDO

À l'aide de PDO et Mysql créer une Todo List en PHP.

Je dois pouvoir :

- Ajouter une todo (qui sera créée en BDD)
- Obtenir une erreur si le champ est vide
- Supprimer une Todo

On utilisera également $_POST et $_GET.

Une todo comprend id, titre, created_at (et checked pour le bonus).

BONUS : Un bouton check (ou une checkbox) sur chaque todo qui quand il est coché va rayer le titre (et donc)

Bon courage !

POO : Programmation Orientée objet

POO : Programmation Orientée Objet

La programmation orientée objet se concentre sur la donnée principalement là où la **programmation structurée** (dès les années 60) et la **programmation procédurale** (conditions + fonctions) se concentre sur le code à proprement parler.

C’est dans les années 90 qu'apparaît la **Programmation Orientée Objets** pour répondre aux besoins de projets modernes.

Le principe dans ce type de programmation est de générer des objets à l’aide de classes que l’on instancie.

On peut noter 3 concepts communs à la POO :

- L’héritage
- L’encapsulation du code
- Le polymorphisme

Qu’est ce qu’un objet ?

Un objet est **une entité qui va pouvoir contenir un ensemble de fonctions et de variables**.

Un objet possèdes des **attributs** ou **propriétés** mais aussi des **méthodes**, c'est-à-dire des actions applicables.

Exemple : un objet qui représente un individu aurait un nom, un âge, une adresse … ce sont les propriétés. Mais un individu peut aussi marcher, courir, manger, dormir … ce sont les méthodes.

La **classe** est le modèle qui **définit la structure de l’objet**. Un objet est créé lorsque la classe est instanciée.

Les classes

Les classes en PHP **servent à générer des objets**.

Elles possèdent donc des **propriétés** (équivalentes aux variables) et des **méthodes** (équivalentes aux fonctions).

2 étapes à suivre :

- Définition de la classe
- Instanciation de la classe (l'objet est créé)

Définition de la classe :

Elle se fait avec le mot-clé **class.** Le nom de la classe doit être en **PascalCase**.

Les **propriétés** sont déclarées en premier ainsi que leur portée : public, private ou protected.Les **méthodes** sont définies comme des fonctions, leur nom doit être en **camelCase**.

Instanciation de la classe :

On instancie une classe avec le mot clé **new** puis le nom de la classe.

On peut ensuite afficher les propriétés ou actions de l'objet créé avec une flèche **->**

Définition de la classe :

```php
class Customer {
    // Propriétés
    public $name='John';

    // Méthodes
    function showName() {
        echo $name;
    }
}
```

Instanciation de la classe :

```php
$customer = new Customer();
echo $customer->showName();
```

POO : Utilisation de $this

L'expression **$this** fait référence à l'instance courante. En d'autres termes, $this nous permet d'utiliser les méthodes et les propriétés de notre classe. Cela signifie que nous pouvons, par exemple, appeler une méthode ou une propriété au sein même de la classe actuelle.

Par exemple **$this->name** fait référence à la propriété $name de l'instance en cours, c'est à dire de la classe dans laquelle nous travaillons.

```php
// Définition de la classe Man
class Man{
    public $name;

    public function say(){
        echo 'Bonjour ' . $this->name;
    }
}
```

```php
// Instancier l'objet
$john = new Man();

// Ajouter le nom
$john->name ='john';

// Appeler la méthode hello() - affichera John
echo $john->say();
```