PROGRAMMATION PHP-8 – MYSQL-8
MYSQL - PDO
L2S – 7-9 AVRIL 2025
3 JOURNEES

PHP – MYSQL

http://php.net/manual/fr/langref.php
http://www.w3schools.com/php/
https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql

SOMMAIRE

RAPPELS DE BD ET USAGES AVEC PHPMYADMIN (WAMP, XAMPP, OU AUTRES)

Architecture Client-Serveur

- **Une BD, c'est un serveur de BD** : on le démarre avec WAMP ou XAMPP ou autre. C'est le programme « mysqld » pour la BD mysql.
- **Une BD, c'est un client pour communiquer avec le serveur de BD**.

2 types de client de BD : GUI et CLI Client-Serveur

- **Un client GUI** : c'est un client avec Graphic User Interface.
  ➔ phpMyAdmin est un client GUI-WEB.
- **Un client CLI** : c'est un client avec Command Line Interface. C'est un client en ligne de commande : écran noir et on tape des commandes. Avec WAMP, on fait : W-bouton gauche-MySQL-Console MySQL. On tape ok pour l'utilisateur root. Pour enter password, on tape « entrée » : il n'y a pas de password (sauf si vous en avez mis un à l'installation).
  ➔ Le client CLI correspond à la commande : mysql -u root -p

Créer et utiliser une Base de données

GUI : créer une Base de données – version 1

- Démarrer phpMyAdmin : W-bouton droit-phpMyAdmin- phpMyAdmin-Connexion
- Cliquez sur « nouvelle base de données » dans la colonne de gauche.
  - ➔ Saisissez un nom + créer : c'est fait.
  - ➔ On peut voir la BD dans la colonne de gauche.

GUI : créer une Base de données – version 2

- Démarrer phpMyAdmin : W-bouton droit-phpMyAdmin- phpMyAdmin-Connexion
- Cliquez sur l'onglet SQL
  - ➔ Dans la fenêtre, saisissez :

```
DROP DATABASE if exists maBD; -- on supprime avant de recréer
CREATE DATABASE maBD;
Use maBD;
```

  - ➔ Avec WAMP sur PC, on a 3 réponses vertes pour nos 3 commandes.

CLI : créer une Base de données

- Démarrer un client : W-bouton gauche-MySQL-Console MySQL - ok - entrée
  - ➔ On obtien un « prompt » mysql : mysql>
  - ➔ On va taper ces commandes :

```
DROP DATABASE if exists maBD; -- on supprime avant de recréer
CREATE DATABASE maBD;
Use maBD;
```

  - ➔ On a 3 « ok » : la BD est créée.

Créer et consulter une table dans la BD

phpMyAdmin : que pour faire du SQL !

- On n'écrit jamais du code directement avec l'interface graphique de phpMyAdmin
  ➔ Sauf pour créer la BD

GUI : phpMyAdmin – SQL

- Dans la colonne de gauche, on commence par cliquer sur la BD dans laquelle on veut travailler.
- Ensuite on clique sur l'onglet SQL
- Ensuite on écrit le code de création des tables.
- Par exemple pour une table ELEVES avec id, nom, dateDeNaissance, email

```
CREATE TABLE ELEVES (
    id INT AUTO_INCREMENT,
    nom VARCHAR(100) NOT NULL,
    dateDeNaissance DATE NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE, -- clé secondaire
    telephone VARCHAR(15), -- pas obligatoire
    PRIMARY KEY(id)
);
```

  ➔ Dans la colonne de gauche, on voit la table

  ➔ On peut cliquer dessus et voir son contenu (onglet parcourir), sa structure (onglet structure)

CLI

- On démarre un client en mode CLI.
- On fait un **use maBD ;**
- On tape le code ci-dessus.
- On fait un **select * from eleves ;** pour voir le contenu de la table eleves.

Créer et consulter des lignes dans une table de la BD

phpMyAdmin : que pour faire du SQL !

- On n'écrit jamais du code directement avec l'interface graphique de phpMyAdmin
  ➔ Sauf pour créer la BD

GUI : phpMyAdmin - SQL

- Dans la colonne de gauche, on commence par cliquer sur la BD dans laquelle on veut travailler.
- Ensuite on clique sur l'onglet SQL
- Ensuite on écrit le code de création d'un élève, par exemple.

```
INSERT INTO ELEVES VALUES
(NULL, 'Toto', '2003-05-15', 'toto@example.com', NULL);
```

  ➔ Pour voir l'élève, on clique sur la table élèves dans la colonne de gauche.

CLI

- On démarre un client en mode CLI.
- On fait un **use maBD ;**
- On tape le code ci-dessus.
- On fait un **select * from eleves ;** pour voir le contenu de la table eleves.

Code complet de création de la BD

On peut aussi avoir un fichier SQL avec tout le code de création de la BD :

```
DROP DATABASE if exists maBD; -- on supprime avant de recréer
CREATE DATABASE maBD;
Use maBD;

CREATE TABLE ELEVES (
    id INT AUTO_INCREMENT,
    nom VARCHAR(100) NOT NULL,
    dateDeNaissance DATE NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE, -- clé secondaire
    telephone VARCHAR(15), -- pas obligatoire
    PRIMARY KEY(id)
);

INSERT INTO ELEVES VALUES
(NULL, 'Toto', '2003-05-15', 'toto@example.com', NULL);

SELECT * FROM ELEVES;
```

On peut coller ce code dans un client GUI, onglet SQL, on dans un client CLI.

Autres usages

Exécuter un SELECT dans phpMyAdmin

En allant dans l'onglet SQL ou dans un client CLI, on peut saisir une requête. Par exemple :

```
SELECT * FROM `utilisateur` WHERE id = 3
```

Exporter avec phpMyAdmin

- On peut exporter la BD dans différents formats :
  - ➔ **SQL** : Ca génère un script SQL qui permet de reconstituer la BD à l'identique, donc une sauvegarde.
    - ➔ Exportez la BD générée et regardez le script.
  - ➔ **CSV** : Ca génère un fichier texte qu'on pourra importer sur Excel.
    - ➔ Exportez la BD en format CSV et importez le fichier CSV dans Excel.

Importer avec phpMyAdmin

- On commence par créer une BD. Ensuite, on peut importer le contenu de la BD dans différents formats :
  - ➔ **SQL** : on choisir le fichier à importer. C'est un fichier avec du code SQL (du DDL et du DML : des CREATE TABLE et des INSERT INTO). Par exemple, c'est un fichier exporté de sauvegarde.
  - ➔ CSV : on peut importer un fichier Excel qui aura été enregistré au format CSV. Il faut préciser le séparateur de colonne (plutôt un « ; » en CSV) et préciser si les colonnes ont une première ligne avec le nom de colonne. On peut dire de ne pas s'arrêter en cas d'erreur d'INSERT. Il faudra ensuite ajouter des contraintes d'intégrité dans la table. Particulièrement mettre les clés primaires et étrangères et donnez un nom à la table.

BD du cours : Posts

Présentation

On utilise une BD_posts permettra de gérer :

- ✅ **Utilisateurs** (création de compte, mise à jour, suppression)
- ✅ **Articles (posts)** rédigés par des utilisateurs
- ✅ **Commentaires** sur les articles
- ✅ **Catégories** d'articles

Code de création de la BD

```
-- Schéma de la BD :

--      comments
--      |    \
--      |     \
--      |    posts
--      |    /   \
--      |  /      \
--     users      categories


-- Création de la BD
DROP DATABASE if exists BD_posts;
CREATE DATABASE BD_posts;
USE BD_posts;


-- Création des tables
-- Table des utilisateurs
CREATE TABLE users (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id)
);

-- Table des catégories d'articles
CREATE TABLE categories (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL UNIQUE,
    PRIMARY KEY(id)
);

-- Table des articles (posts)
CREATE TABLE posts (
    id INT AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_id INT,
    category_id INT,
```

```
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (category_id) REFERENCES categories(id),
    PRIMARY KEY(id)
);

-- Table des commentaires
CREATE TABLE comments (
    id INT AUTO_INCREMENT,
    post_id INT,
    user_id INT,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (post_id) REFERENCES posts(id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    UNIQUE(post_id, user_id), -- clé secondaire
    PRIMARY KEY(id)
);


-- Insérer des utilisateurs
INSERT INTO users (name, email, password) VALUES
('Alice', 'alice@example.com', 'hashedpassword1'),
('Bob', 'bob@example.com', 'hashedpassword2');

-- Insérer des catégories
INSERT INTO categories (name) VALUES
('Tech'),
('Lifestyle'),
('Science');

-- Insérer des articles
INSERT INTO posts (user_id, category_id, title, content) VALUES
(1, 1, 'Les nouveautés en IA', 'Blabla sur l\'intelligence
artificielle...'),
(2, 2, '10 astuces pour mieux dormir', 'Blabla sur le
sommeil...');

-- Insérer des commentaires
INSERT INTO comments (post_id, user_id, content) VALUES
(1, 2, 'Super article, merci !'),
(2, 1, 'Intéressant, merci pour les conseils.');


SELECT * FROM users; -- Liste des utilisateurs
SELECT * FROM posts; -- Liste des articles
SELECT * FROM comments; -- Liste des commentaires
SELECT * FROM categories; -- Liste des commentaires
```

PHP – MYSQL

1. Bibliothèques utilisées

Utilisation de la base de données en PHP

- 3 jeux de fonctions (API) permettent de se connecter à la BD et de l'utiliser :
  - mysql
  - mysqli
  - PDO
    - ➔ http://php.net/manual/fr/mysqlinfo.api.choosing.php
- Le jeu mysql est le plus ancien : mieux vaut l'éviter.
- On peut utiliser mysqli ou PDO (PHP Data Object), et particulièrement PDO_MYSQL.
- Mysqli permet d'utiliser MySQL et est plus facile à mettre en œuvre.
- PDO permet d'utiliser n'importe quel SGBD et est plus complexe à mettre en œuvre.

PDO ou mysqli ? PDO !!!

- https://websitebeaver.com/php-pdo-vs-mysqli
- https://openclassrooms.com/forum/sujet/debat-pdo-vs-mysqli
- Dans les framework modernes, c'est PDO qui est utilisé.
- L'intérêt du PDO est que c'est une interface d'abstraction permettant l'utilisation de n'importe quelle BD. De plus elle est « orienté objet ».
  - ➔ On utilisera plutôt PDO_MYSQL.
  - ➔ http://php.net/manual/en/ref.pdo-mysql.php

PHP – MYSQL – PDO

00. Présentation du document

Exemples et exercices

- Les exemples sont présentés dans un chapitre en vert avec le mot clé : exemple-
- Les dossiers d'exemples sont fournis dans l'article qui contient ce fichier de cours.
  - ➔ http://bliaudet.free.fr/IMG/zip/PHP-03-PHP-MySQL.zip
  - ➔ Chargez ce fichier et mettez-le dans le dossier « php » du répertoire web « www » du serveur WEB. Vous pouvez aussi structurer les choses avec des dossiers J1, J2, etc. correspondant aux journées de travail.

- Les exercices à faire sont présentés dans un chapitre en jaune avec le mot-clé « exercice : »
  - ➔ Les sources pour les exercices, quand il y en a, sont fournis dans le dossier des exemples.
- Un TP final est en fin de document

Remarque

- Certains fichiers d'exemple commencent par ces trois lignes :

```
echo '<h1>CODE PHP</h1>';
highlight_file('fichier.php');
echo '<h1>RESULTATS</h1>';
```

- Ce code affiche deux balises h1 avec CODE PHP puis RESULTATS
- La fonction « highlight_file » permet d'afficher le contenu du fichier proposé. Quand on teste le code, on commence par affiche le code. Ca permet de voir le code en même temps que les résultats.
- Pour généraliser le code, on écrit : highlight_file(basename(__FILE__));
- basename(__FILE__) permet de récupérer le nom du fichier en cours de traitement.

Document « slidé »

- Ce document Word est en partie « slidé » : chaque page tient, en général, sur un écran, comme un slide.

Objectifs généraux de ce document

- Utilisation de la bibliothèque PDO
- Usage de la programmation objet en PHP
- Select et DML en PDO
- Organisation du code non MVC

Exemples

1. Connexion à la BD : new PDO
2. Afficher le contenu d'une table : SELECT (R du CRUD)
3. Afficher le contenu d'une table en MVC : SELECT (R du CRUD)
4. Afficher le contenu d'une table en MVC : SELECT (R du CRUD) + order by et autres.
5. Afficher le contenu d'une table en MVC : SELECT (R du CRUD) + Where variable avec saisie d'une condition (GET ou POST).

PHP-MySQL :

- OCR :
  ➔ https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/presentation-des-bases-de-donnees-2
- Manuel de Référence du PHP :
  ➔ http://php.net/manual/fr/langref.php
- Manuel de Référence du SQL :
  ➔ http://www.w3schools.com/sql/ : à noter que la représentation ensembliste pour expliquer les jointures n'a aucun sens !

1. Connexion à la BD

Connexion à la BD : new PDO (exemple-1 – connexion)

Code

```php
<?php
function connexionBD($dbname)
{
    // paramètres de la base de donnée
    $sgbdname = 'mysql';
    $host = 'localhost';
    $charset = 'utf8';
    // dsn : data source name
    $dsn =
        $sgbdname .
        ':host=' . $host .
        ';dbname=' . $dbname .
        ';charset=' . $charset;

    // utilisateur connecté à la base de donnée
    $username = 'root';
    $password = 'root';

    // pour avoir des erreurs SQL plus claires
    $erreur = array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    );

    try {
        // connexion à la BD : new PDO
        $bdd = new PDO($dsn, $username, $password, $erreur);
        echo '<p>Connexion réussie</p>';
        return $bdd;
    } catch (PDOException $e) {
        echo 'Connexion échouée : ' . $e->getMessage();
        return NULL;
    }
}
```

new PDO

On fait un new PDO avec 4 paramètres

- PDO est une classe.
  ➔ http://php.net/manual/fr/pdo.construct.php
- On crée un nouvel objet de la classe qu'on appelle $bdd.
- Le new PDO à 4 paramètres :
  ➔ $**dsn** (data source name) : contient des infos sur le SGBD (mysql), le serveur (host, ici : localhost), le nom de la BD, le jeu de caractères utilisé (UTF8 pour que ce soit le plus générique).
  ➔ $**username** : nom de l'utilisateur qui se connecte à la BD.
  ➔ $**password** : password de l'utilisateur qui se connecte à la BD.
  ➔ $**erreur** : pour gérer les messages d'erreur.

On utilise des variables pour rendre le code lisible et facile à paramétrer :

- $host : la machine du serveur de SGBD,
- $sgbdname : le type de SGBD,
- $username : le nom de l'utilisateur qui se connecte sur la BD,
- $password : le mot de passe de cet utilisateur,
- $dbname : le nom de la BD à laquelle on accède sur le SGBD.

$erreur : gestion des erreurs

```
$erreur = array(
   PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
);
```

- En ajoutant le paramètre $erreur tel qu'il est là dans le new PDO, on aura des messages d'erreurs du SGBD, par exemple si le SELECT est mal écrit.
- $erreur est un tableau associatif. On lui fournit un couple key => value.
  - ➔ key : PDO ::ATTR_ERRMODE
  - ➔ value : PDO::ERRMODE_EXCEPTION
- Syntaxe de POO-PHP :
  - ➔ PDO ::ATTR_ERRMODE et PDO::ERRMODE_EXCEPTION sont des constantes de classe définies dans la classe PDO.
  - ➔ https://www.w3schools.com/php/php_oop_constants.asp
  - ➔ Classe::CONSTANTE_DE_CLASSE
  - ➔ PDO ::ATTR_ERRMODE : http://php.net/manual/fr/pdo.constants.php
  - ➔ PDO::ERRMODE_EXCEPTION : http://php.net/manual/fr/pdo.error-handling.php

Gestion des erreurs avec try catch

- La structure « try » « catch » est une structure de programmation objet pour gérer les erreurs : les exceptions.
  - ➔ On essaye un code : try
  - ➔ Si ça « plante », on attrape une exception et on la gère
  - ➔ https://www.w3schools.com/php/php_exception.asp
- Le die permet d'arrêter proprement l'exécution de la page en cas d'erreur.
  - ➔ Il ne s'utilise pas avec PDO.

2. print_r du contenu d'une table

Afficher le contenu d'une table -1 : query(), fetch(), print_r (exemple-2-select)

Exemple

```php
<?php
// Le modèle
include("connexion.php");
$bdd = connexionBD('BD_Posts');

// Le contrôleur

// 1 : on écrit la requête
$reqSQL = 'SELECT * FROM users';

// 2 : on récupère le résultat
$reqPHP = $bdd->query($reqSQL);
echo '<pre>';
print_r($reqPHP);
echo '</pre>';

// 3 : on affiche le résultat ligne par ligne
$users = $reqPHP->fetchAll();
echo '<pre>';
print_r($users);
echo '</pre>';

// 4 : on libère les tables de la requête
$reqPHP->closeCursor(); // pour finir le traitement
?>
```

Explications

- On écrit la requête SQL dans $reqSQL : c'est une simple chaine de caractères.
- On utilise la méthode (fonction) query de $bdd (->) en passant la $reqSQL en paramètre.
  - ➔ Le résultat est dans $reqPHP : c'est un objet complexe sur lequel on peut appliquer des méthodes (des fonctions) qui donnera accès au tableau de données du select.
  - ➔ On affiche la $reqPHP pour voir son contenu : seule la requête s'affiche (la requête n'est pas exécutée à ce stade).
- $reqPHP est « fétché » : on fait un « fetchAll() qui récupère toutes les lignes des users.
  - ➔ Chaque $ligne retournée par le fetch est un tableau associatif : on peut faire un print_r de $users. La balise <pre> permet un affichage plus lisible : ça reste du débug.
  - ➔ On constate alors qu'on a deux accès possibles à chaque donnée : par le nom du champs (id par exemple) ou par un numéro (0 pour id) : il y a donc 2 fois plus d'éléments que prévu dans chaque ligne
- Quand on a fini de travailler, on fait un closeCursor, pour libérer les tables de la requête

Résultats

```
Connexion réussie

DEBUT

PDOStatement Object
(
    [queryString] => SELECT * FROM utilisateur
)
while($ligne=$resultat->fetch()){

Array
(
    [id] => 1
    [0] => 1
    [prenomNom] => Sia PEI
    [1] => Sia PEI
    [adMail] => ji@gmail.com
    [2] => ji@gmail.com
    [motDePasse] => jipei
    [3] => jipei
    [annee] => 1995
    [4] => 1995
)
nombre d'éléments de $ligne : 10

Array
(
    [id] => 2
    [0] => 2
    Etc.
```

3. Affichage HTML du contenu d'une table

Affichage HTML :

C'est un simple affichage du tableau associatif de $users.
On a donc traité le R du CRUD des users.
Il va nous rester à le structurer en MVC.

```
<body>
    <h1> Les Users </h1>
    <table border>
        <tr>
            <th>id</th>
            <th>name</th>
            <th>email</th>
            <th>password</th>
            <th>created_at</th>
        </tr>
        <?php foreach($users as $user) { ?>
            <tr>
                <td><?php echo $user['id'] ?></td>
                <td><?php echo $user['name'] ?></td>
                <td><?php echo $user['email'] ?></td>
                <td><?php echo $user['password'] ?></td>
                <td><?php echo $user['created_at'] ?></td>
            </tr>
        <?php } ?>
    </table>
</body>
```

Exercices

Exercice-1 : chargez les exemples 1 et 2 et testez-les

- Il faut parfois charger les BD pour pouvoir tester les exemples.

4. Vocabulaire de Programmation orientée objet

https://www.w3schools.com/php/php_oop_what_is.asp

Classe

https://www.w3schools.com/php/php_oop_classes_objects.asp

- Une classe, c'est un type, comme un entier, un réel, un caractère, une string ou un booléen.
- Une classe va permettre de créer de variables de type « classe » : ce seront des objets.
- En plus, on associe des fonctions à une classe : on les appelle alors « méthode ».

Objet

https://www.w3schools.com/php/php_oop_classes_objects.asp

- Un objet c'est une variable de type Classe.
- En général, un objet correspond à l'équivalent d'un tableau associatif : il contient des couples clé-valeur. Les différentes clés sont appelées « attribut » (comme avec du JSON).
- Une classe va permettre de définir les clés – attributs - de façon générale pour tous les objets de la classe : comme ça, on n'a pas besoin de les redéfinir à chaque fois.
- Quand on crée un objet avec des valeurs pour les couples clé-valeur (pour les attributs), on dit qu'on instancie un objet.
  ➔ Ca passe par la commande « new ».

Méthode

- Les méthodes sont des fonctions qui sont attachées à une classe.
- Elle ne se sont utilisables que par les objets de la classe.
- On écrit : objet->methode() pour appeler la méthode pour l'objet en question : c'est comme si on avait passé l'objet en paramètre de la méthode.

Constructeur

- Le constructeur est la méthode qui permet d'instancier un nouvel objet.
  https://www.w3schools.com/php/php_oop_constructor.asp

Attribut d'instance (attribut d'objet)

- Les attributs d'un objet peuvent avoir des valeurs différentes pour chaque objet.
- Un attribut d'objet est défini au niveau de la classe. Chaque objet instancié portera les attributs de sa classe et pourra avoir des valeurs spécifiques pour chacun des attributs.

Attribut et constante de classe

https://www.w3schools.com/php/php_oop_constants.asp

- Un attribut de classe est un attribut qui a une valeur « attachée » à la classe et pas à un objet particulier.
- On utilise souvent les attributs de classe pour définir des constantes de classe : des attributs attachés à une classe avec une valeur fixe.

Exceptions

- Les erreurs en POO sont gérés par une mécanisme de « try » « catch » :
- On essaye (try) un code. Si ça marche on fait certains traitements. Si ça plante on attrape (catch) l'erreur générée et on fait les traitements nécessaires.
- Les erreurs sont appelées des « exceptions ».

Guides de style

- https://fr.wikipedia.org/wiki/CamelCase
- https://eilgin.github.io/php-the-right-way/#code_style_guide

5. Technique de programmation – PDO et PDOStatement - $bdd, $reqSQL, $reqPHP

- Pour manipuler la BD, on utilise principalement 2 classes correspondant à 2 objets :

  ➔ classe **PDO** -> objet **$bdd**

  ➔ classe **PDOStatemenet** -> objet **$reqPHP**

Terminologie : $bdd - $reqSQL - $reqPHP - $ligne

- **$bdd :** un objet de la classe PDO sera appelé $bdd. C'est en quelque sorte l'objet qui permet l'accès concret à la base de données, pour un utilisateur et une base de donnée.
- **$reqSQL :** le texte de la requête sera mis dans un $reqSQL. C'est une simple chaine de caractères. Il ne doit pas être confondu avec le résultat de la requête : $requete.
- **$reqPHP :** un objet de la classe PSOStatement sera appelé $requete (statement peut vouloir dire « requête ». Cette $reqPHP est un objet complexe qui contient à la fois le $reqSQL et permet l'accès au résultat de la requête une fois celle-ci exécutée.
- **$ligne :** le résultat d'un fetch() est une ligne : $ligne =$requete->fetch(). C'est un tuple de la table résultant de la requête SQL.

$bdd (PDO) – query - prepare

- http://php.net/manual/fr/class.pdo.php
- La classe PDO ne contient que des méthodes (elle ne contient pas d'attributs). Notons Particulièrement :
  - ➔ **query($reqSQL)** : renvoie un **$reqPHP** auquel est associé le $reqSQL passé en paramètre et donne accès au résultat de la requête (le résultat du Select) prêt à être fetché (prêt à être parcouru).
  - ➔ **prepare($reqSQL)** : renvoie un **$reqPHP** auquel est associé le $reqSQL passé en paramètre. La requête n'a pas été exécutée. Elle peut contenir des variables.
- PDO contient aussi des méthodes propres à une BD comme la gestion des transactions : commit, rollback, etc., et d'autres choses.

$requete (PDOStatement) - execute - fetch - closeCurseur

- http://php.net/manual/fr/class.pdostatement.php
- La classe PDOStatement contient un attribut : la valeur du $resSQL fourni en paramètre quand il a été créé. Il contient aussi des méthodes. Notons particulièrement :
  - ➔ **execute()** : permet d'exécuter une une requête avec des variables. Il faut fournir en paramètre un tableau de valeurs pour les variables.
  - ➔ **fetch()** : permet de récupérer les lignes du résultat de la requête, une par une.
  - ➔ **fetchAll()** : permet de récupérer les lignes du résultats de la requête, toutes dans un tableau.
  - ➔ **closeCurseur()** : permet de refaire un execute.

Synthèse

$bdd : PDO	$reqPHP : PDOStatement
-> query($reqSQL) : PDOStatement	-> fetch() : ligne
-> prepare($reqSQL) : PDOStatement	-> exectute() : bool
	-> closeCursor() : bool
	-> fetchAll() : toutes les lignes

Déboguer : or die bdd->errorInfo()

Présentation

- On a déjà vu les 2 techniques pour déboguer :
  - ➔ le paramètre $erreur dans le new PDO()
  - ➔ le « or die() »

Créer le $bdd avec la gestion des erreurs

- Pour afficher les détails d'une erreur, on crée un $bdd avec la gestion des erreurs :

```
$erreur = array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION);
$bdd = new PDO($dsn, $username, $password, $erreur);
```

- C'est suffisant pour obtenir des messages d'erreur propres.

Aternative : exécuter la requête (query ou execute) « or die »

- Si on n'utilise pas un $erreur dans le new PDO, on peut ajouter un « or die » au query ou au execute pour afficher les détails d'une erreur.

```
$requete=$bdd->query($reqSQL) or die(print-r($bdd-
>errorInfo())) ;
```

ou

```
$requete->execute(array( … )) or die(print-r($bdd-
>errorInfo())) ;
```

  - ➔ Le « or die » est inutile avec une connexion PDO en ERRMODE.
  - ➔ http://php.net/manual/fr/pdo.errorinfo.php

Les limites du « or die » : confidentialité et user friendly

- Le « or die » peut être pratique en phase de développement.
- En production, il met au jour des informations qui peuvent être confidentielles et n'est pas user-friendly.
- http://www.alsacreations.com/tuto/lire/676-gestion-erreurs-mysql-php-or-die.html

6. Structuration MVC

On reprend le 3) et on le structure en MVC

Création du vue.php

Dans vue.php, on met le code HTML complet

Création de modele_users.php

Dans modele_users.php, on crée une fonction selectUsers qui retourne les $users.
Cette fonction prend le $bdd en paramètre.
Elle reprend le code du SELECT.

```php
<?php
function selectUsers($bdd)
{
    // 1 : on écrit la requête
    $reqSQL = 'SELECT * FROM users';

    // 2 : on récupère le résultat
    $reqPHP = $bdd->query($reqSQL);

    // 3 : on affiche le résultat ligne par ligne
    $users = $reqPHP->fetchAll();

    // 4 : on libère les tables de la requête
    $reqPHP->closeCursor(); // pour finir le traitement

    // on returne les $users
    return $users;
}
```

Création du controleur ctrl_users_select.php

Création du controleur : c'est index_select qu'on renomme ctrl_users_select.
Il include le modèle et la vue :

```php
<?php
// Modèle
include("./connexion.php");
include("./modele_users.php");
$bdd = connexionBD('BD_Posts');

// Contrôleur
$users=selectUsers($bdd);

// Vue
include("./vue.php");
```

Exercice : (1) regardez l'exemple 3 et testez-les

- Il faut parfois charger les BD pour pouvoir tester les exemples.

7. Autres Select et MVC

Where et Order by – exemple 4/order-by

On fait une requête avec un order by sur une nouvelle BD

```
$reqSQL = 'SELECT *
    FROM films
    WHERE realisateur = \'KING VIDOR\'
    ORDER BY annee
';
```

On part du code MVC et on le met à jour.

Exercice : (2) regardez l'exemple 4 et testez-les

- Il faut parfois charger les BD pour pouvoir tester les exemples.

Exercice : (3) créer un code MVC pour la requête suivante :

```
$reqSQL='SELECT *
    FROM films
    WHERE realisateur like \'%manki%\'
    order by annee';
```

```
// like % manki % : n'importe quoi autour de manki
```

Exercice : (4) créer un code MVC qui intègre les 2 SELECT

Il faut 2 contrôleurs : un ctrl_vidor et un ctrl_manki
Dans le modele de film on aura 2 fonctions :
function selectFilmsVidor($bdd)
function selectFilmsManki($bdd)

Exercice : (5) ajouter une 3ème select au code précédent :

```
$reqSQL='SELECT annee, titre, realisateur
    FROM films
    where annee=1960
    order by titre
    LIMIT 0, 10
';
```

```
// on prend les 10 premiers (de 1 à 10)
// limit 10, 10 pour les 10 suivants
// limite 20, 10 pour les 10 suivants, etc.
```

TP

Les exercices du cours

- Tous les exercices du cours, précédés par « Exercice : » et en jaune, peuvent être faits.
- On peut y accéder en chechant « Exercice : »

Application : app_posts_html

- Dans les exemples associés au cours, vous trouvez le dossier app_posts_html.
- C'est du HTML.

1 : Testez-le

- Testez-le : lancez le ficher _index.html.

2 : Factorisez le code

- Il y a trois pages.
- Factorisez le code pour le header, le nav et le footer.
  - Vous devez donc coder du PHP et mettre le projet dans le dossier www/home.

  ➔ Testez le résultat

3 : Le menu « users » va afficher les users

- Utilisez la BD du cours : BD_posts (cherchez BD_posts dans le cours).
- Chargez cette BD.
- Faites-en sortes que le menu « users » permette d'afficher tous les users.

4 : Le menu « post » va afficher les posts

- Comme le 3, mais pour le menu « posts ».