

# Programmation web II

---



Mohamed Karim Abdmouleh

karim.abdelmoula@iit.ens.tn

1<sup>ère</sup> année Génie Informatique

---

# Chapitre 3 : Connexion à la Base de Données MySQL

---



# Accès aux SGBD

## *Pourquoi ?*

- Stockage permanent des saisies des formulaires
- faciliter l'accès et la manipulation des données
- Fouiller dans de gros volumes de données
- gérer les relations entre les données
- intégration avec systèmes existants

## *Supports*

**MySQL**, SqlLite, ODBC, Oracle, SQL server, MSQL, Adabas, Dbase, DBM, ICAP, Informix, PostgreSQL, Sybase, Firebase, MongoDB....



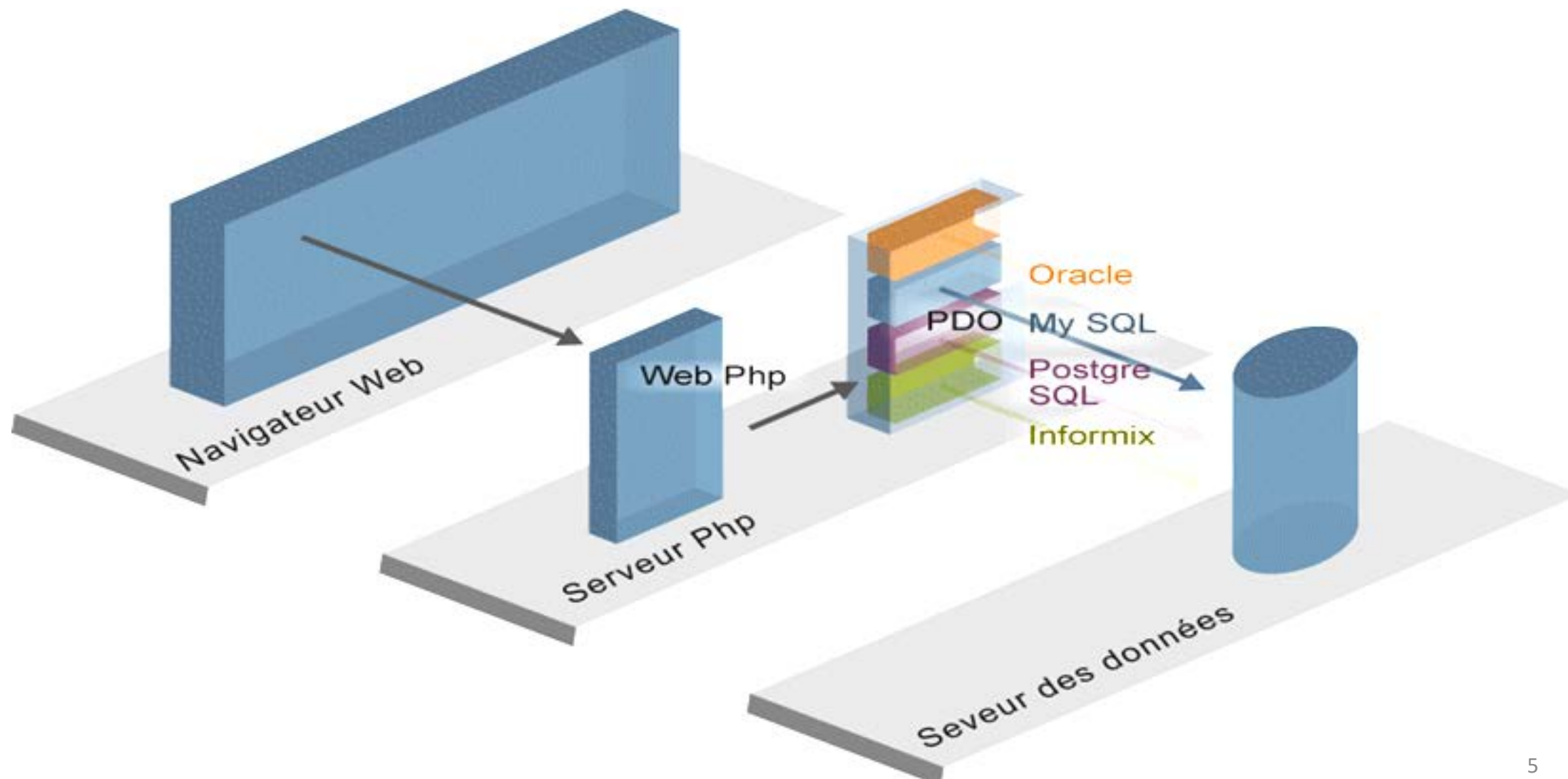
# Connexion avec MySQL

## Présentation de MySQL

- MySQL est un système de gestion de base de données relationnelles (**SGBDR**) open source,
- Il est développé et maintenu par **Oracle**.
- Il est l'un des systèmes de gestion de bases de données **les plus populaires** au monde et est *utilisé* dans **de nombreux projets web**, notamment pour stocker des données structurées telles que des informations d'utilisateurs, des catalogues de produits et des données de commande.
- MySQL utilise une **syntaxe SQL** (Structured Query Language) pour interagir avec la base de données.
- Il prend en charge les transactions ACID (Atomicité, Cohérence, Isolation et Durabilité), ce qui signifie que les transactions sont traitées de manière fiable et en toute sécurité, même en cas d'erreurs ou de pannes du système.



# Accès aux SGBD à travers l'API PDO



# Accès aux SGBD à travers PDO

## ■ PDO (*PHP Data Objects*)

- API fournit une interface unifiée pour accéder à différents types de bases de données.
- PDO est apparu avec PHP 5

## ■ Avantages

- PDO offre une couche d'abstraction de base de données, ce qui permet aux développeurs de travailler avec **différents SGBD sans avoir à réécrire leur code pour chaque SGBD.**
- orienté objet
- Les objets de l'interface PDO utilisent des **exceptions**, il est donc tout à fait possible d'intégrer facilement un système de **gestion des erreurs.**





## Rappel

# Qu'est ce qu'un objet ?

**Exemple** : Gérer une boutique en ligne d'articles de sport

### article 1

désignation =	Sac à dos
référence =	S1245
prixHT =	110
calculerPrixTTC() = fonction {}	

**Idée 1** : regrouper ces variables dans une boîte (encapsulation)  
cette boîte on va lui donner un nom **article 1**  
**article 1 c'est l'objet**

### article 2

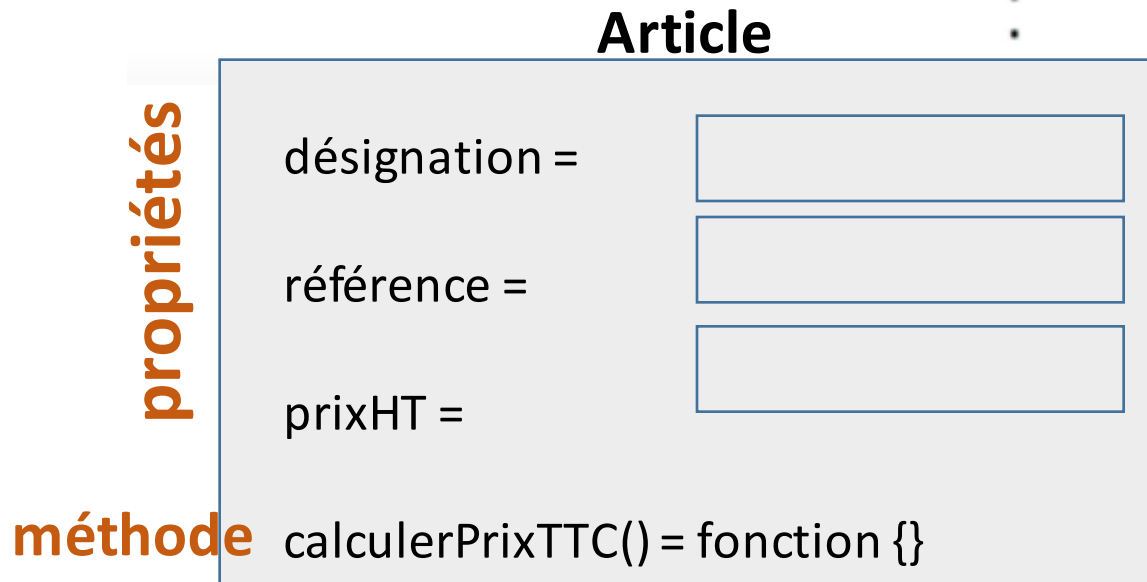
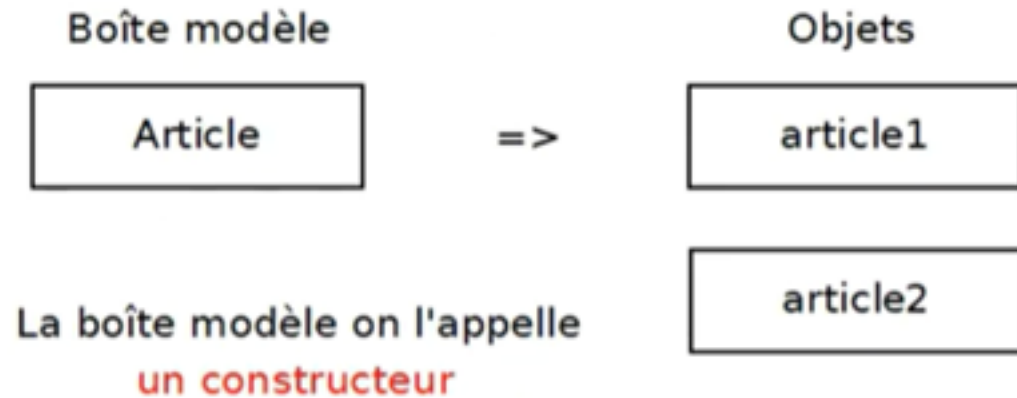
désignation =	survêtement
référence =	S4939
prixHT =	160
calculerPrixTTC() = fonction {}	

**Idée 2** : créer une boîte **qui va nous servir de modèle** pour construire chacun de nos articles  
**Boite de modèle = Article (avec des variables non initialisées)**

### Article

désignation =	
référence =	
prixHT =	
calculerPrixTTC() = fonction {}	

# Qu'est ce qu'un objet ?



- Qu'est ce qu'il doit faire un **constructeur** pour construire un nouvel objet ?

# Allouer de la mémoire

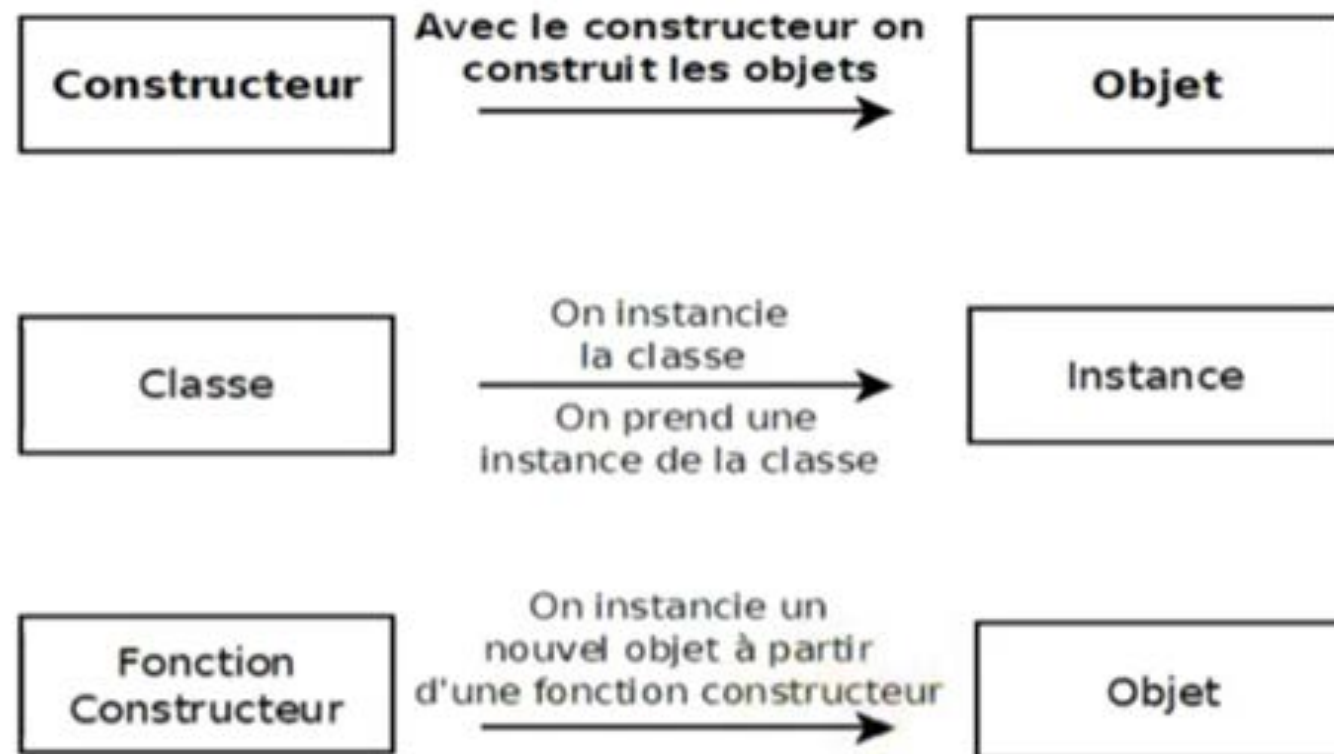
1. Il va regarder la liste des variables qu'on lui a donné dans le modèle
2. implanter la même liste de variables
3. initialise ces variables avec les valeurs spécifiques à ce nouvel objet





# Un point de vocabulaire

- pour mettre en œuvre notre constructeur en **PHP**, il faut :
- utiliser une **classe** ou
- utiliser une fonction **constructeur**





# Comment accéder à la BD ?

- Créer une instance de type PDO :

Pour créer un objet de la classe, il faut utiliser l'opérateur **new** suivi du nom de la classe :

```
$pdo= new PDO($dsn,$login,$password);
```

```
$dsn = 'mysql:host=localhost;dbname=mydatabase';
```

```
$login = 'username';
```

```
$password = 'password';
```



- si vous avez changé le port MySQL par défaut (3306), vous devez spécifier le nouveau port lors de la connexion.

```
<?php
$host = 'localhost';
$db = 'nom_de_la_base_de_donnees';
$user = 'nom_utilisateur';
$pass = 'mot_de_passe';
$port = '3307'; // Le port que vous avez configuré pour MySQL
try {
    $dsn = "mysql:host=$host;dbname=$db;port=$port";
    $pdo = new PDO($dsn, $user, $pass);
    // Configurer PDO pour afficher les erreurs
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // Exécuter vos requêtes ici...
} catch (PDOException $e) {
    echo 'Erreur de connexion : ' . $e->getMessage();
}??>
```

# Les paramètres d'accès

- le **DSN** (*Data Source Name*)

*c'est une chaîne de caractères contenant les informations de connexion à la base : **nom du SGBD**, **nom du serveur**, nom de **la base de données***

**Ex :** 'mysql:host=localhost;dbname=mydatabase';

- le **login** (optionnel)

**Ex :** root

- le **mot de passe** (optionnel)

**Exemple d'instance de la classe PDO :**

**\$pdo=new PDO('mysql:host=localhost;dbname=basebiblio','root','');**





# Avantage PDO : Gestion des exceptions

La gestion des exceptions est importante car elle permet

- d'identifier rapidement et de manière fiable les erreurs,
- de fournir des messages d'erreur utiles pour aider à diagnostiquer et résoudre les problèmes,
- et de garantir que le programme peut continuer à fonctionner même en cas d'erreur

```
try {  
    $pdo=new PDO('mysql:host=localhost;dbname=basebiblio','root','');  
}  
catch(PDOException $e)  
{ printf("Echec de la connexion : %s\n", $e->getMessage()); exit();  
}
```

**printf** est utilisée pour afficher un message d'erreur spécifique

# Quelques méthodes fournies par la classe PDO

- **errorInfo()** : Retourne les informations liées à l'erreur rencontrée dans la dernière opération sur la base. Le retour est de type tableau.
- **exec()** : Exécute une requête SQL de type **insert, update ou delete** et retourne le nombre de lignes affectées. Le retour est de type entier.
- **query()** : Exécute une requête SQL de type **select**, retourne un jeu de résultats en tant qu'objet PDOStatement.
- **lastInsertId()** : Retourne l'identifiant de la dernière ligne insérée. Le retour est de type string.
- **FETCH\_ASSOC** : Retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats.



# Quelques méthodes fournies par la classe PDO

- **rowCount()** : retourne le nombre de lignes affectées par le dernier appel à la fonction. Elle ne fonctionne qu'avec des requêtes de type INSERT, UPDATE ou DELETE.
- **columnCount()** : Retourne le nombre de colonnes dans le jeu de résultats.
- **fetch()** : Récupère la ligne suivante d'un jeu de résultat PDO.
- **fetchAll()** : Retourne un tableau contenant toutes les lignes du jeu de résultat,
- **fetchColumn()** : Retourne une colonne depuis la ligne suivante d'un jeu de résultats PDO.
- **fetchObject()** : Récupère la ligne suivante et la retourne en tant qu'objet.





### Exemple :

```
$pdo =new PDO('mysql:host=localhost;dbname=basebiblio','root','');
```

```
$req= "SELECT count(*) FROM livre WHERE liv_id='$id' " ;
```

```
$res=$pdo->query($req);
```

```
$n= $res->fetchColumn(0) ;
```

```
// 0 est relatif à la première colonne càd le résultat de count(*)
```





# MySQL

- Le nom de la table et le nom des champs ne doivent pas comporter un espace
- Pour le type varchar, on doit spécifier la taille e.g., VARCHAR(20)

# Etude de cas : Gestion d'utilisateurs

- Soit une table utilisateur qui contient la liste des utilisateurs.

- Un utilisateur est caractérisé par :

- Son cin
- Son nom

En HTML

Numéro cin utilisateur	<input type="text"/>
Nom utilisateur	<input type="text"/>
	<input type="button" value="Enregistrer"/>

**inscriptionForm.html**

Numero cin utilisateur	Nom utilisateur	Modifier	Supprimer
1471456	Aziz	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
12345678	Wassim	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
44444	Alia	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
74185	charles	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
748596	Oumaima	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
788945	Alia	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
12021	Malek	<a href="#">Modifier</a>	<a href="#">Supprimer</a>

**liste.php**

code	<input type="text" value="12345678"/>
nom	<input type="text" value="Wassim"/>
	<input type="button" value="modifier"/>

**modifForm.php**



# Etude de cas : Gestion d'utilisateurs

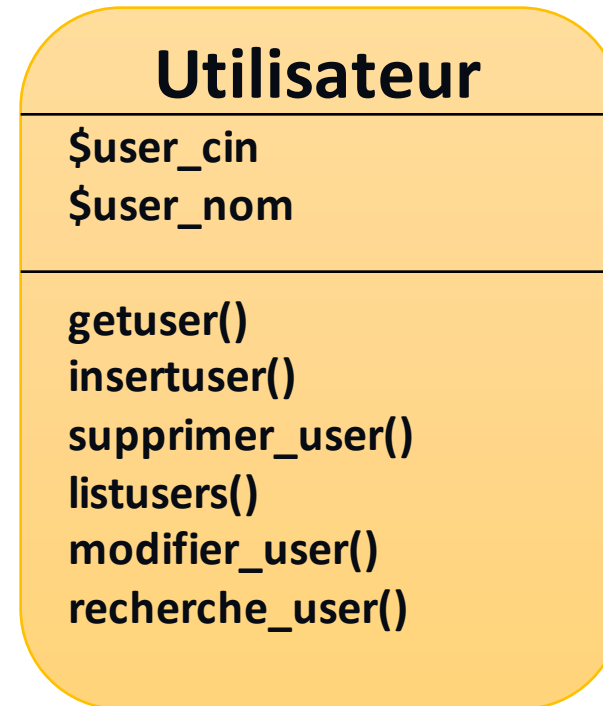
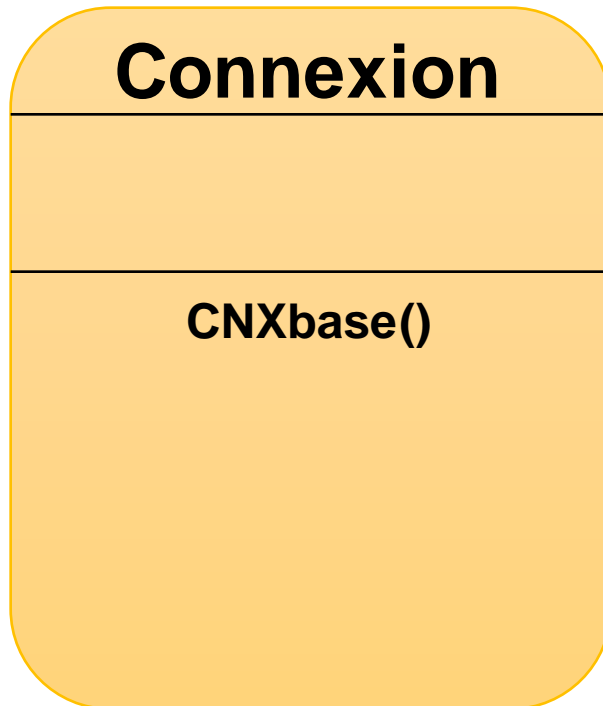
## Remarque

- Nous avons choisi le paradigme Orienté Objet pour réaliser cette application, cependant, il est également possible de la résoudre en utilisant le paradigme procédural










# Etude de cas : Gestion d'utilisateurs

- Pour cela on va créer, en PHP, une classe **Utilisateur** pour la gestion des utilisateurs de la BD et une classe **Connexion** pour la configuration de la BD à travers le PDO.



# Projet arborescence (8 fichiers)

- 1  config.php
- 5  inscription.php
- 4  inscriptionForm.html
- 3  liste.php
- 7  modifForm.php
- 8  modification.php
- 6  sup.php
- 2  user.class.php



## Paramètres de connexion MySQL : config.php

```
<?php
class connexion
{
public function CNXbase()
{
    $dbc=new PDO('mysql:host=localhost;dbname=personnel','root','');
    return $dbc;
}
}
?>
```



## Création de classes en php (user.class.php)

```
<?php
class utilisateur
{
/* attributs de la classe utilisateur*/
    public $user_cin;
    public $user_nom;
```

### Utilisateur

**\$user\_cin**  
**\$user\_nom**

getuser()  
insertuser()  
supprimer\_user()  
listusers()  
modifier\_user()  
recherche\_user()



## Insertion d'un enregistrement

```
function insertuser()  
{  
    require_once('config.php');  
    $cnx=new connexion();  
    $pdo=$cnx->CNXbase();  
    $req="insert into utilisateur (user_cin, user_nom) values  
        ('$this->user_cin','$this->user_nom')";  
  
    $pdo->exec($req) or print_r($pdo->errorInfo());  
}
```

### Utilisateur

user\_cin  
user\_nom

getuser()  
**insertuser()**  
supprimer\_user()  
listusers()  
modifier\_user()  
recherche\_user()





## Visualiser les données d'une table

```
function listusers()  
{  
    require_once('config.php');  
    $cnx=new connexion();  
    $pdo=$cnx->CNXbase();  
  
    $req="SELECT * FROM utilisateur";  
    $res=$pdo->query($req) or print_r($pdo->errorInfo());  
    return $res;  
}
```

### Utilisateur

user\_cin  
user\_nom

getuser()  
insertuser()  
supprimer\_user()  
**listusers()**  
modifier\_user()  
recherche\_user()



## Afficher un utilisateur

```
function getuser($id)
{
    require_once('config.php');
    $cnx=new connexion();
    $pdo=$cnx->CNXbase();
    $req="SELECT * FROM utilisateur where user_cin=$id";
    $res=$pdo->query($req) or print_r($pdo->errorInfo());
    return $res;
}
```

Remarque

Si user\_cin est de type varchar  
alors il faut ajouter les quotes  
au \$id

```
.... where user_cin='$id'";
```

### Utilisateur

user\_cin  
user\_nom

#### getuser()

insertuser()  
supprimer\_user()  
listusers()  
modifier\_user()  
recherche\_user()



## *Mettre à jour des données dans une table*

```
function modifier_user($id)
{
    require_once('config.php');
    $cnx=new connexion();
    $pdo=$cnx->CNXbase();
    $req="UPDATE utilisateur SET user_nom='$this->user_nom' WHERE
    user_cin=$id";
    $pdo->exec($req) or print_r($pdo->errorInfo());
}
```

### Utilisateur

user\_cin  
user\_nom

getuser()  
insertuser()  
supprimer\_user()  
listusers()  
**modifier\_user()**  
recherche\_user()



## Supprimer des données d'une table

```
function supprimer_user($id)
{
    require_once('config.php');
    $cnx=new connexion();
    $pdo=$cnx->CNXbase();

    $req="DELETE FROM utilisateur WHERE user_cin=$id";
    $pdo->exec($req) or print_r($pdo->errorInfo());
}
```

Remarque

Si user\_cin est de type varchar  
alors il faut ajouter les quotes  
au \$id

```
.... where user_cin='$id';
```

### Utilisateur

user\_cin  
user\_nom

getuser()  
insertuser()  
**supprimer\_user()**  
listusers()  
modifier\_user()  
recherche\_user()



## Vérifier l'existence d'un enregistrement

```
function recherche_user()  
{  
    require_once('config.php');  
    $cnx=new connexion();  
    $pdo=$cnx->CNXbase();  
    $req= "SELECT count(*) FROM utilisateur WHERE user_cin='$this->user_cin' " ;  
    $res=$pdo->query($req) or print_r($pdo->errorInfo());  
    return $res;  
}  
  
//fin de la classe  
  
} ?>
```

### Utilisateur

user\_cin  
user\_nom

getuser()  
insertuser()  
supprimer\_user()  
listusers()  
modifier\_user()  
**recherche\_user()**



# Afficher liste des utilisateurs: liste.php

```
<?php
require_once('user.class.php');

$us=new utilisateur();
$res=$us->listusers();

?>

<table border='1'>
<tr><td>Numero cin utilisateur</td>
<td>Nom utilisateur</td>
<td>Modifier</td>
<td>Supprimer</td></tr>
```

Numero cin utilisateur	Nom utilisateur	Modifier	Supprimer
1471456	Aziz	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
12345678	Wassim	<a href="#">Modifier</a>	<a href="#">Supprimer</a>



```
<?php
foreach($res as $row)
{
echo "<tr><td>$row[0]</td>";
echo "<td>$row[1]</td>";
echo "<td><a href ='modifForm.php?id=$row[0] ' > Modifier</a></td>";
echo "<td><a href='sup.php?id=$row[0] ' >Supprimer</a></td> </tr>";
}
echo "</table>";
?>
```

Numero cin utilisateur	Nom utilisateur	Modifier	Supprimer
1471456	Aziz	<a href="#">Modifier</a>	<a href="#">Supprimer</a>
12345678	Wassim	<a href="#">Modifier</a>	<a href="#">Supprimer</a>



## Formulaire : inscriptionForm.html

Numéro cin utilisateur	<input type="text"/>
Nom utilisateur	<input type="text"/>
	<input type="submit" value="Enregistrer"/>

```
<!DOCTYPE html>
<html lang="en">
<body>
  <form method="post" action="inscription.php">
    <table border="1">
      <tr>
        <td>Numero cin utilisateur</td>
        <td><input type="text" name="cinuser"></td>
      </tr>
      <tr>
        <td>Nom utilisateur</td>
        <td><input type="text" name="nomuser" ></td>
      </tr>
      <tr>
        <td></td>
        <td><input type="submit" name="Submit" value="Enregistrer"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```





## Appel des méthodes de la classe : inscription.php

```
<?php
require_once('user.class.php');
$us=new utilisateur();
$us->user_cin=$_POST['cinuser'];
$us->user_nom=$_POST['nomuser'];
$row=$us-> recherche_user();
$n= $row->fetchColumn(0)    ; // fetchColumn(0) retourne la valeur relative à la
première colonne (n° 0).
    if($n==0) { $us->insertuser();
        header('location:liste.php'); }
else {      header('location:inscriptionForm.html');
}??>
```



## Appel des méthodes de la classe : sup.php

```
<?php
require_once('user.class.php');
$us=new utilisateur();
$us-> supprimer_user($_GET['id']);
header('location:liste.php');
?>
```



## Appel des méthodes de la classe : modifForm.php

```
<body>
<?php require_once('user.class.php');
    $us=new utilisateur();
    $res=$us->getuser($_GET['id']);
    $data=$res->fetchAll(PDO::FETCH_ASSOC);
    $cin= $data[0]["user_cin"];
    $nom=$data[0]["user_nom"] ;

?>
<form method='post' action='modification.php'>
<table>
<tr><td>code </td>
    <td><input type = "text" name = "cin" value = "<?php echo $cin ?>"
readonly/> </td></tr>
<tr><td>nom</td>
    <td><input type = "text" name = "nom" value = "<?php echo $nom ?>"    />
</td></tr>
<tr><td><input type = "submit" value= "modifier" /> </td>
    <td></td> </tr>    </table> </form></body>
```

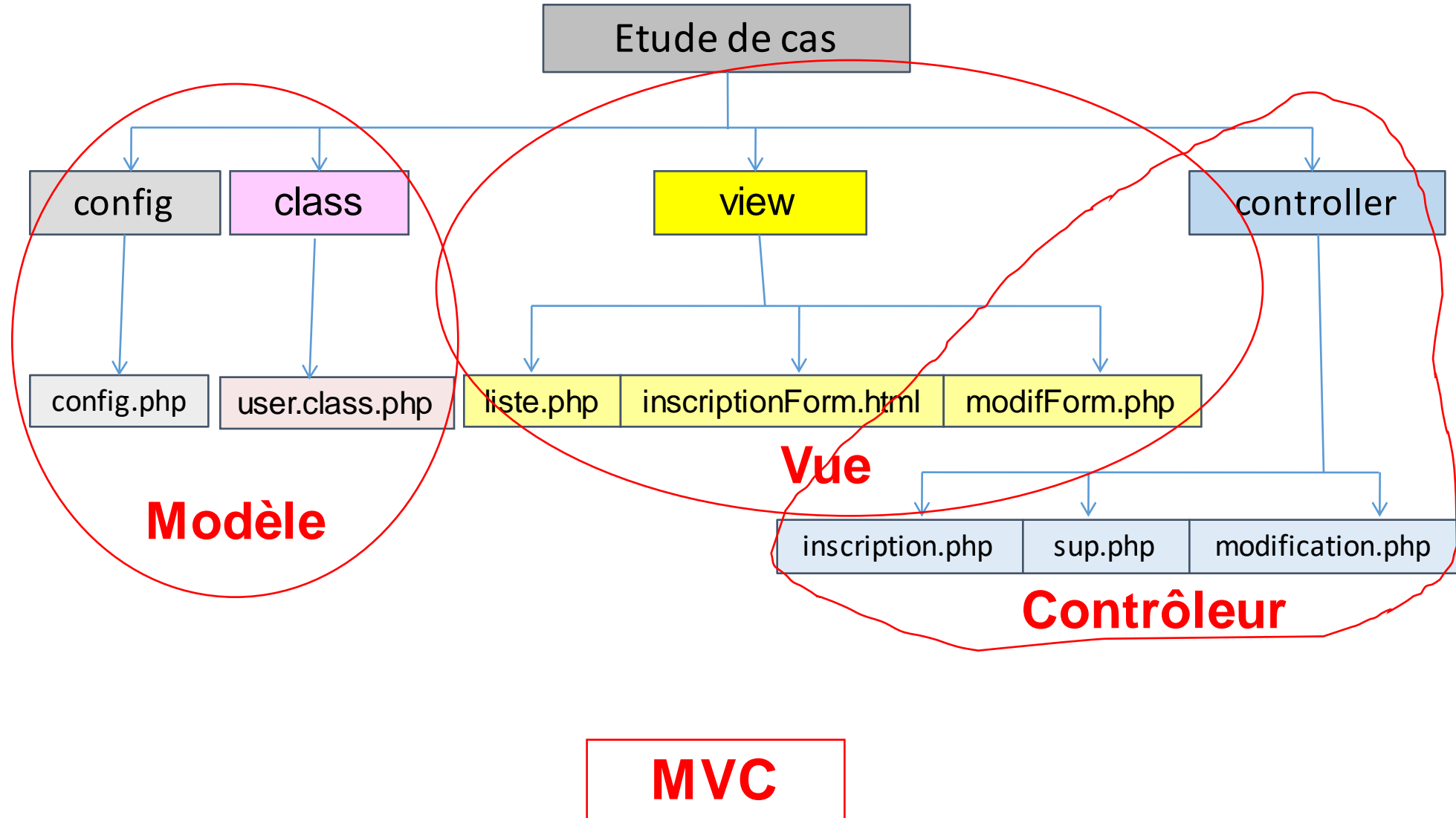
code	<input type="text" value="12345678"/>
nom	<input type="text" value="Wassim"/>
<input type="button" value="modifier"/>	



## modification.php

```
<?php
require_once('user.class.php');
$us=new utilisateur();
$us->user_cin=$_POST['cin'];
$us->user_nom=$_POST['nom'];
$us-> modifier_user($_POST['cin']);
header('location:liste.php');
?>
```

# Arborescence de l'étude de cas





# BOOTSTRAP

- Bootstrap est un framework front-end qui permet de créer des interfaces utilisateur modernes avec moins d'efforts
  - Bootstrap est un framework open-source développé par Twitter.
  - Il est basé sur HTML, CSS et JavaScript, et il offre un ensemble d'outils, de composants et de styles préconçus pour la conception rapide et la création de sites web **réactifs** et esthé
- 
- **réactif (responsive)** : s'adapte de manière fluide à différents appareils et tailles d'écrans tiques.



# Ajouter BOOTSTRAP

- TROIS méthodes pour ajouter bootstrap
  1. Exécuter dans le terminal du VSCODE **npm i bootstrap@5.3.2** :
  2. Télécharger le fichier **bootstrap.min.css** et l'ajouter au fichier html dans la balise LINK du HEAD
  3. Copier les deux lignes **CDN** suivants dans le HEAD

```
<link  
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"  
rel="stylesheet">  
  
    <script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.  
js"></script>
```