



# MVC Cinéma Structuration



**ELAN**

14 rue du Rhône - 67100 STRASBOURG

☎ 03 88 30 78 30 ✉ [elan@elan-formation.fr](mailto:elan@elan-formation.fr)

[www.elan-formation.fr](http://www.elan-formation.fr)

SAS ELAN au capital de 37 000 € -

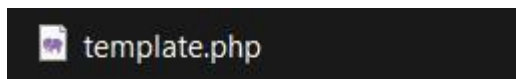
RCS Strasbourg B 390758241 – SIRET 39075824100041 – Code APE : 8559A

N° déclaration DRTEFP 42670182967 - Cet enregistrement ne vaut pas agrément de l'Etat

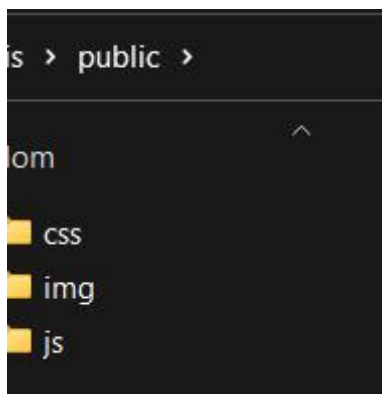
## I. Création d'un nouveau dossier de projet pour exploiter la base de données Cinéma

Nom	Modifié le	Type	Taille
controller	30/03/2022 08:11	Dossier de fichiers	
model	30/03/2022 08:11	Dossier de fichiers	
public	19/05/2022 15:49	Dossier de fichiers	
view	30/03/2022 08:11	Dossier de fichiers	
index.php	31/03/2022 11:32	Fichier source PHP	2 Ko

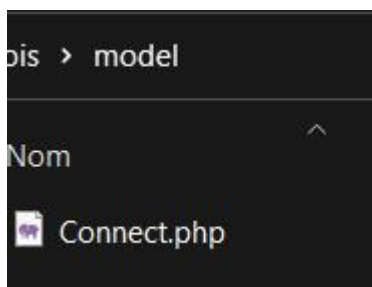
Dans le dossier "view" on créera l'ensemble des vues affichant les résultats de nos requêtes mais aussi un fichier "template.php" qui servira de base / squelette à l'ensemble des vues. Ainsi on aura besoin de déclarer le doctype, les links css / js etc qu'une seule fois dans ce fichier. On exploitera ce qu'on appelle "la temporisation de sortie" en PHP



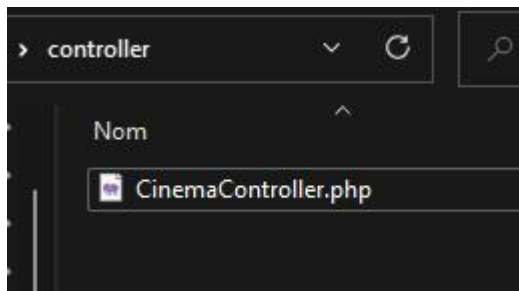
Dans "public" on retrouvera un dossier css / js / img



Dans "model" on y trouvera pour le moment que le fichier "Connect.php" permettant de se connecter à la BDD (grâce à PDO)



Dans "controller", un seul fichier pour le moment également qui contiendra l'ensemble des requêtes dans les fonctions en relation avec les vues



## II. Création des fichiers

Il faudra bien comprendre que l'index.php va servir à accueillir l'**action** transmise par l'URL (en GET) (comme on avait fait dans le projet "Appli PHP" avec le panier)

On aura toujours une URL de la forme

index.php?action=listFilms

Dans index.php :

- On "use" le controller Cinema
- On autocharge les classes du projet
- On instancie le controller Cinema
- En fonction de l'action détectée dans l'URL via la propriété "action" on interagit avec la bonne méthode du controller

```
index.php > ...
1  <?php
2
3  use Controller\CinemaController;
4
5  spl_autoload_register(function ($class_name) {
6      include $class_name . '.php';
7  });
8
9  $ctrlCinema = new CinemaController();
10
11 if(isset($_GET["action"])){
12     switch ($_GET["action"]) {
13
14         case "listFilms" : $ctrlCinema->listFilms(); break;
15         case "listActeurs" : $ctrlCinema->listActeurs(); break;
16
17     }
```

On aura toujours un **namespace** permettant de catégoriser virtuellement (dans un espace de nom la classe en question). Ainsi on pourra "**use**" la classe sans connaître son emplacement physique. On a juste besoin de savoir dans quel namespace elle se trouve, pratique !

Dans le fichier "Connect.php" on se contente de déclarer la connexion à la base de données

```
<?php

namespace Model;

abstract class Connect {

    const HOST = "localhost";
    const DB = "cinema";
    const USER = "root";
    const PASS = "";

    public static function seConnector() {
        try {
            return new \PDO(
                "mysql:host=".self::HOST.";dbname=".self::DB.";charset=utf8", self::USER, self::PASS);
        } catch(\PDOException $ex) {
            return $ex->getMessage();
        }
    }
}
```

La classe est abstraite car on n'instanciera jamais la classe Connect puisqu'on aura seulement besoin d'accéder à la méthode "**seConnector**"

On remarquera au passage le namespace de la classe **Connect** --> "**Model**", ainsi que la présence d'un "\" devant PDO indiquant au framework que **PDO est une classe native** et non une classe du projet

Dans notre CinemaController, on va pouvoir créer la requête qui nous intéresse

Pour la liste des films par exemple

```
<?php

namespace Controller;
use Model\Connect;

class CinemaController {

    /**
     * Lister les films
     */
    public function listFilms() {

        $pdo = Connect::seConnecter();
        $requete = $pdo->query("
            SELECT titre, annee_sortie
            FROM film
        ");

        require "view/listFilms.php";
    }
}
```

- On se connecte
- On exécute la requête de notre choix
- On relie par un "require" la vue qui nous intéresse (située dans le dossier "view")

On remarquera ici l'utilisation du "use" pour accéder à la classe Connect située dans le namespace "Model"

```
<?php

namespace Controller;
use Model\Connect;
```

Chaque vue aura la structure suivante :

```
1  <?php ob_start(); ?>
2
3  <p class="uk-label uk-label-warning">Il y a <?= $requete->rowCount() ?> films</p>
4
5  <table class="uk-table uk-table-striped">
6      <thead>
7          <tr>
8              <th>TITRE</th>
9              <th>ANNEE SORTIE</th>
10         </tr>
11     </thead>
12     <tbody>
13         <?php
14         foreach($requete->fetchAll() as $film) { ?>
15             <tr>
16                 <td><?= $film["titre"] ?></td>
17                 <td><?= $film["annee_sortie"] ?></td>
18             </tr>
19         <?php } ?>
20     </tbody>
21 </table>
22
23 <?php
24
25 $titre = "Liste des films";
26 $titre_secondaire = "Liste des films";
27 $contenu = ob_get_clean();
28 require "view/template.php";
```

On commence et on termine la vue par "**ob\_start()**" et "**ob\_get\_clean()**"

On va donc "aspirer" tout ce qui se trouve entre ces 2 fonctions (temporisation de sortie) pour **stocker** le contenu dans une variable **\$contenu**

Le require de fin permet d'injecter le contenu dans le template "squelette" > template.php

Du coup dans notre "template.php" on aura des variables qui vont accueillir les éléments provenant des vues

Donc **DANS CHAQUE VUE**, il faudra toujours donner une valeur à **\$titre**, **\$contenu** et **\$titre\_secondaire**

Le titre

```

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="public/css/style.css">
  <title><?= $titre ?></title>
</head>
<body>
  <nav class="uk-navbar-container" uk-navbar uk-sticky>

```

Le titre secondaire et le contenu principal

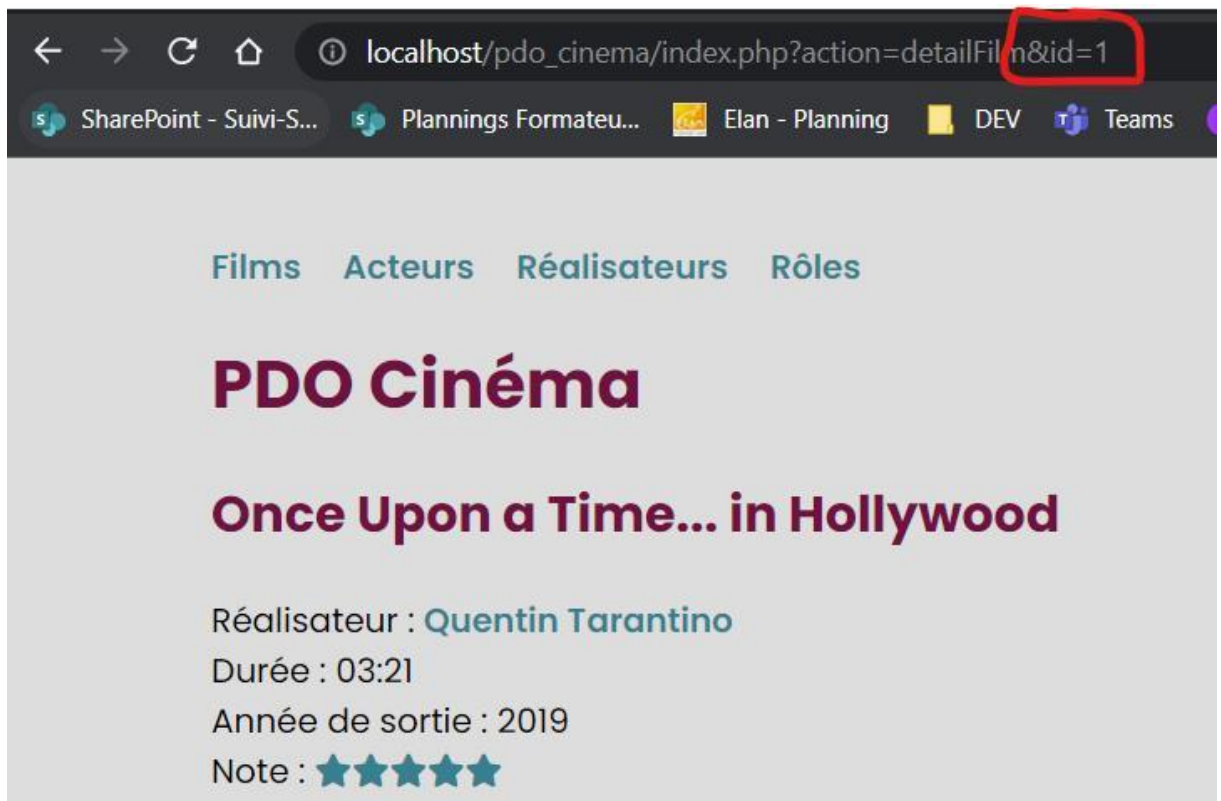
```

</nav>
<div id="wrapper" class="uk-container uk-container-expand">
  <main>
    <div id="contenu">
      <h1 class="uk-heading-divider">PDO Cinema</h1>
      <h2 class="uk-heading-bullet"><?= $titre_secondaire ?></h2>
      <?= $contenu ?>
    </div>
  </main>
</div>

```

Piste dans "index.php" pour afficher le détail d'un film par exemple, c'est encore une fois l'URL qui fera passer un "id" en paramètre





```
$id = (isset($_GET["id"])) ? $_GET["id"] : null;
// $type = (isset($_GET["type"])) ? $_GET["type"] : null;

if(isset($_GET["action"])){
    switch ($_GET["action"]) {
        // Films
        case "listFilms": $ctrlCinema->listFilms(); break;
        X case "detailFilm": $ctrlCinema->detailFilm($id); break;
```

Quand vous faites une requête dans lequel on a un élément variable (comme ici l'id de l'acteur), il faut bien faire un **"prepare"** (et pas un "query") pour ensuite faire un "execute"

Notez que dans le **"execute"** on fait bien passer un tableau associatif qui associe le nom de champ paramétré avec la valeur de l'id (celui passé dans la méthode : \$id)

```
public function detActeur($id) {
    $pdo = Connect::seConnecter();
    $requete = $pdo->prepare("SELECT * FROM acteur WHERE id_acteur = :id");
    $requete->execute(["id" => $id]);
    require "view/acteur/detailActeur.php";
}
```



Pour interpréter les données renvoyées par la requêtes, deux choix s'offrent à nous :

- On **fetch** s'il n'y a qu'un seul résultat attendu (une ligne sur HeidiSQL) --> retourne un tableau associatif
- On **fetchAll** si un ensemble de résultats est attendu (plusieurs lignes sur HeidiSQL) --> retourne un tableau de tableaux associatifs

### Exemple :

Si on veut afficher le détail d'un film (informations de base du film et le casting du film), on doit effectuer 2 requêtes dans la même méthode du **Controller** souhaité :

- Une requête qui va récupérer les informations du film : SELECT

```
$requeteFilm = $pdo->prepare("SELECT ... WHERE id_film = ..."); // ne renvoie qu'une seule ligne

$requeteFilm->execute([  ]);
```

- Une requête qui renvoie le casting du film

```
$requeteCasting = $pdo->prepare("SELECT acteur, role FROM
WHERE f.id_film = ..."); // renvoie potentiellement plusieurs
lignes
$requeteCasting->execute( [ ]);
```

### require view/ ...

Et dans la vue :

```
$film = $requeteFilm->fetch();
echo $film["titre"];
```

```
$casting = $requeteCasting->fetchAll();
foreach($casting as $cast) {
    echo $cast["acteur"]." dans le rôle de ".$cast["role"];
}
```