

Mission : Adapter une application au modèle MVC

- ✓ **Compétence visée : Répondre aux incidents et aux demandes d'assistance et d'évolution**
- ✓ **Sous compétence visée : Traiter des demandes concernant les applications**

Objectifs du TP : A partir d'une application de gestion d'une bibliothèque développée en PHP Orientée objet, on a demandé de la réorganiser et ajouter les instructions nécessaires pour l'adapter au modèle MVC.

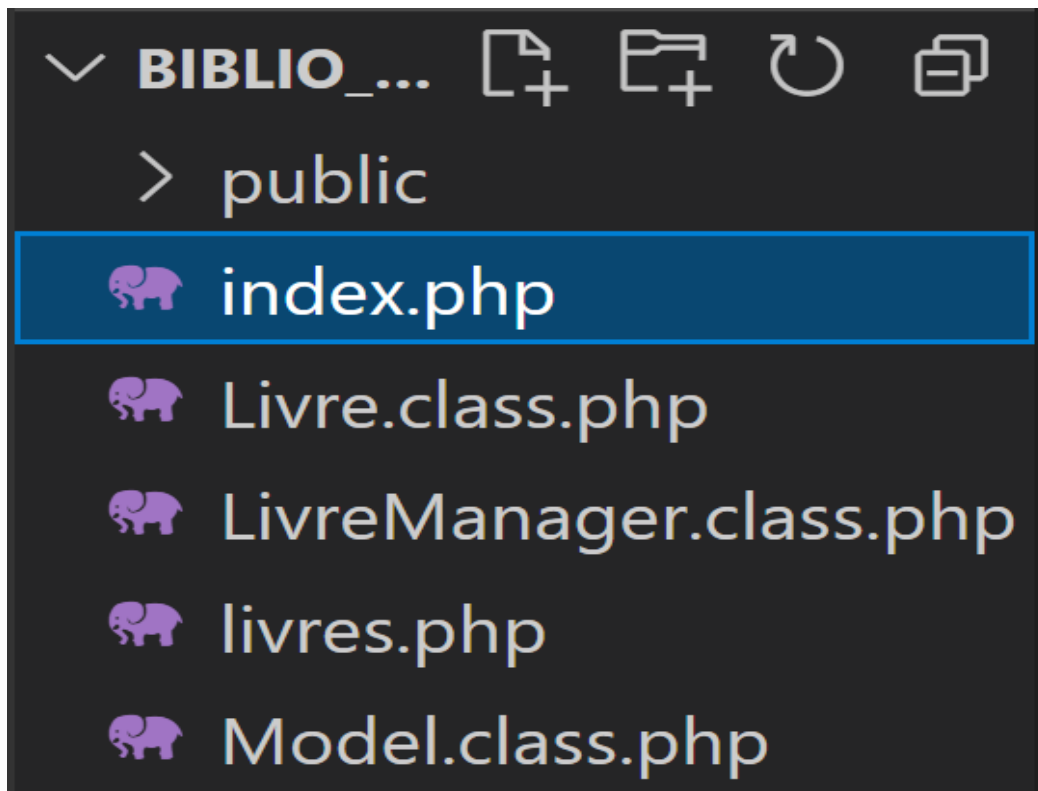
1) Le MVC

Le modèle MVC (Modèle – Vue - Contrôleur) permet de bien organiser le code source. Son but est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts :

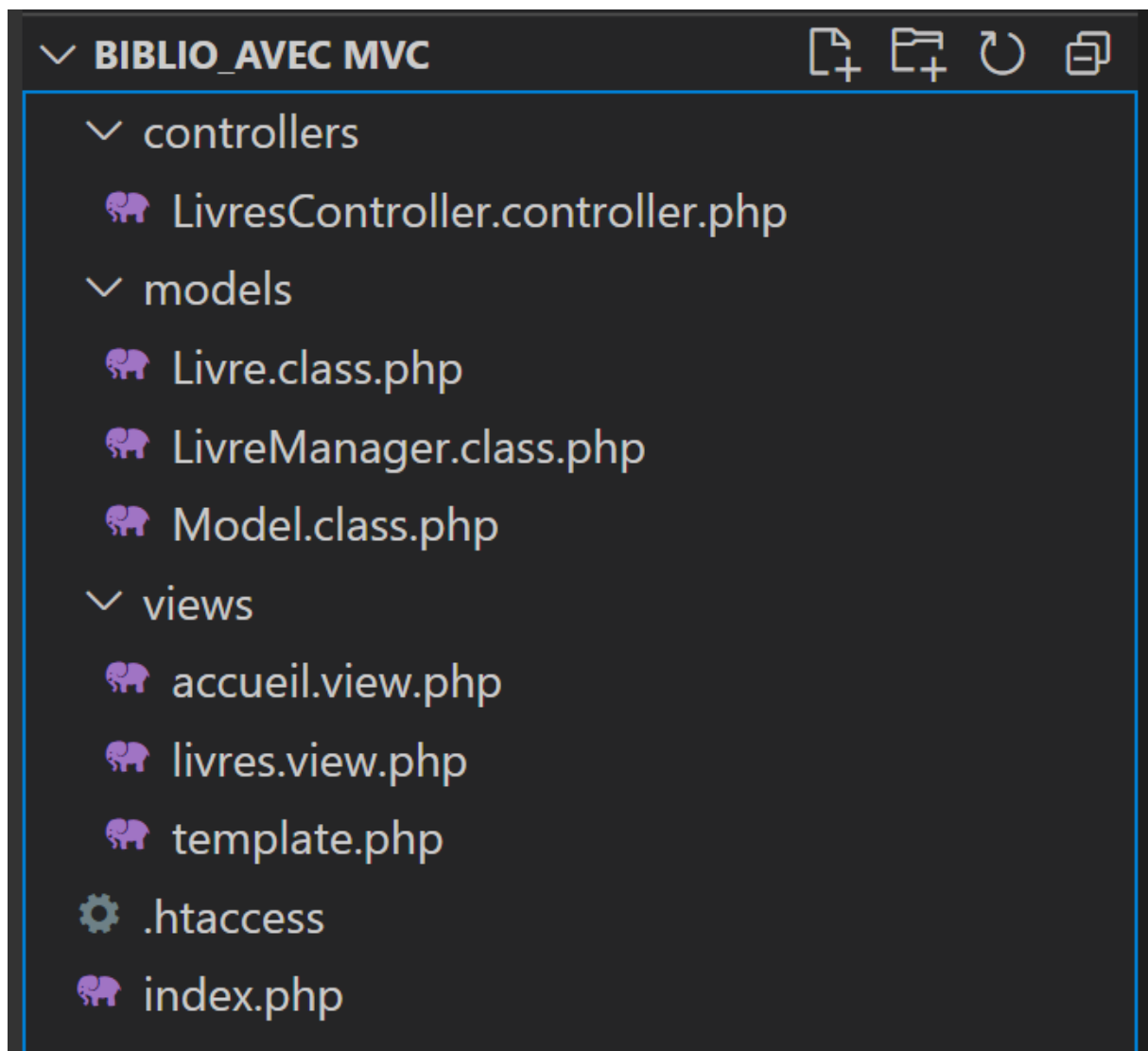
- Le modèle (M) : qui contient les données (\simeq base de données).
- Le contrôleur (C) qui prend les décisions (\simeq la partie « algorithmique » : chef d'orchestre entre modèle et vue)
- La vue (V) (template) qui organise la présentation (affichage).

2) Le travail demandé

- La structure de l'application fournie est le suivant :
- Le fichier index.php contient la page d'accueil
- La classe Livre contient l'entité livre
- La classe LivreManager est la classe qui permet de gérer la liste des livres
- Le fichier livres.php permet d'afficher un tableau HTML de la liste des livres
- La classe model.php permet la gestion de la BD : connexion



Etape1 : créer 3 dossiers : models-views-controllers



Etape2 : Le routeur

Le fichier index.php va jouer le rôle de routeur, il va diriger la demande de l'utilisateur vers le contrôleur.

Code source avant routage

```
index.php > ...  
1  <?php  
2  ob_start();  
3  ?>  
4  
5  Ici la page d'accueil  
6  
7  <?php  
8  $content = ob_get_clean();  
9  $titre = "Bibliothèque MGA";  
10 require "template.php";  
11 ?>
```

Le code source après routage

```
<?php
require_once "controllers/LivresController.controller.php";
$livreController = new LivresController;

if(empty($_GET['page'])){
    require "views/accueil.view.php";
} else {
    switch($_GET['page']){
        case "accueil" : require "views/accueil.view.php";
        break;
        case "livres" : $livreController->afficherLivres();
        break;
    }
}
```

Etape 3 : Fichier .htaccess

Pour réécrire et simplifier l'affichage de l'url.

Toutes les pages de l'application partent de la page index.php qui auront les url de la forme suivante : **index.php?page=nom_page**

Exemple pour la page d'accueil : <http://localhost/biblio/index.php?page=accueil>

on veut la rendre : <http://localhost/biblio/accueil>

```
⚙ .htaccess ×
⚙ .htaccess
1 RewriteEngine On
2
3 RewriteCond %{REQUEST_FILENAME} !-f
4 RewriteCond %{REQUEST_FILENAME} !-d
5
6 RewriteRule ^(.*)$ index.php?page=$1
```

Etape 4 : Déplacer les fichiers dans les dossiers models-views- controllers selon leurs rôles

- Déplacer le fichier livres.php dans le dossier views et le rendre accessible par le routeur et le renommer livres.views.php
- Séparer le fichier dans les différentes parties MVC
- Déplacer le fichier template dans le dossier view et mettre à jour le contenu

Ajouter la route vers la view livres

```
index.php X
index.php
1  <?php
2
3  if(empty($_GET['page'])){
4      require "views/accueil.view.php";
5  } else {
6      switch($_GET['page']){
7          case "accueil" : require "views/accueil.view.php";
8          break;
9          case "livres" : require "views/livres.view.php";
10         break;
11     }
12 }
13
```

Changer le lien vers la pages livres

```
template.php X
views > template.php > html > head
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7      <link rel="stylesheet" href="https://bootswatch.com/4/sketchy/bootstrap.min.css">
8  </head>
9  <body>
10     <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
11         <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-label="Toggle navigation">
12             <span class="navbar-toggler-icon"></span>
13         </button>
14
15         <div class="collapse navbar-collapse" id="navbarColor01">
16             <ul class="navbar-nav mr-auto">
17                 <li class="nav-item">
18                     <a class="nav-link" href="accueil">Accueil</a>
19                 </li>
20                 <li class="nav-item">
21                     <a class="nav-link" href="livres">Livres</a>
22                 </li>
23             </ul>
24         </div>
25     </nav>
26
27     <div class="container">
28         <h1 class="rounded border border-dark p-2 m-2 text-center text-white bg-info">?<? $titre ?></h1>
29         <?> $content ?>
30     </div>
31
32     <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4891E2+poT44NhyKhv5vZF5SrPo01EjwBvKU7imGFAV0wwj1yYfRSJoZ+n" crossorigin="anonymous"></script>
33     <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvox4fooAo" crossorigin="anonymous"></script>
34     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdJ003uMBJnJnuUD41h7YwaYd1qfktj0Uod8GCEX130g81fwB6" crossorigin="anonymous"></script>
35 </body>
36 </html>
```

Le contenu de livres.view.php

```
livres.view.php X
ews > livres.view.php > ...
1  <?php
2
3  require_once "LivreManager.class.php";
4  $livreManager = new LivreManager;
5  $livreManager->chargementLivres();
6
7  ob_start();
8  ?>
9
10 <table class="table text-center">
11     <tr class="table-dark">
12         <th>Image</th>
13         <th>Titre</th>
14         <th>Nombre de pages</th>
15         <th colspan="2">Actions</th>
16     </tr>
17     <?php
18     $livres = $livreManager->getLivres();
19     for($i=0; $i < count($livres);$i++ :
20     ?>
21     <tr>
22         <td class="align-middle"></td>
23         <td class="align-middle"><?= $livres[$i]->getTitre(); ?></td>
24         <td class="align-middle"><?= $livres[$i]->getNbPages(); ?></td>
25         <td class="align-middle"><a href="" class="btn btn-warning">Modifier</a></td>
26         <td class="align-middle"><a href="" class="btn btn-danger">Supprimer</a></td>
27     </tr>
28     <?php endfor; ?>
29 </table>
30 <a href="" class="btn btn-success d-block">Ajouter</a>
31
32 <?php
33 $content = ob_get_clean();
34 $titre = "Les livres de la bibliothèque";
35 require "template.php";
36 ?>
```

Etape 5 : créer le contrôleur

Le fichier livres.view.php contient le code suivant(récupération des données, exécution des fonctions,...) qui ne correspond pas à l’affichage.

```
<?php

require_once "LivreManager.class.php";
$livreManager = new LivreManager;
$livreManager->chargementLivres();

ob_start();
?>
```

On doit créer un fichier contrôleur appelé livres.controller.php qui contient la partie pilotage



```
1  <?php
2  require_once "models/LivreManager.class.php";
3
4  class LivresController{
5      private $livreManager;
6
7      public function __construct(){
8          $this->livreManager = new LivreManager;
9          $this->livreManager->chargementLivres();
10     }
11
12     public function afficherLivres(){
13         $livres = $this->livreManager->getLivres();
14         require "views/livres.view.php";
15     }
16 }
```

Etape 6 : déplacer les fichiers model dans le dossier models

- Le fichier model.class.php permet la connexion à la BD
- Livre.class.php contient la classe livre
- LivreManager.class.php contient la classe LivreManager

