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 :

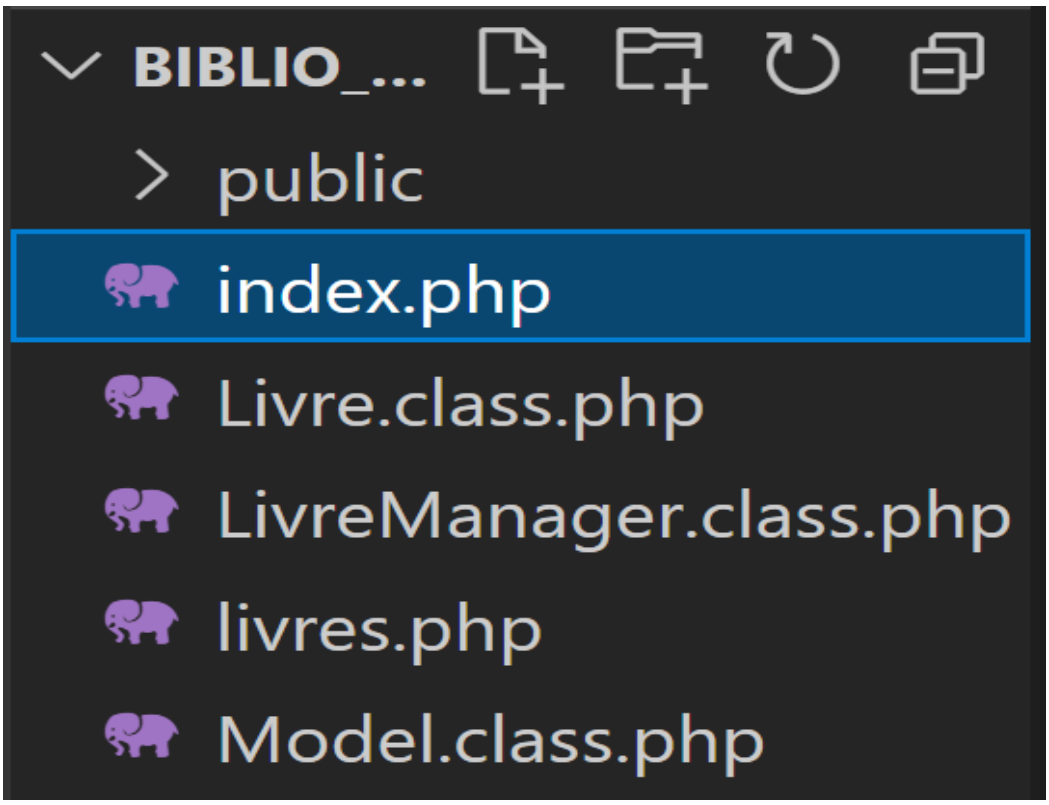
- Le modèle (M) : qui contient les données (≃ base de données).

- Le contrôleur (C) qui prend les décisions (≃ la partie « algorithmique » : chef d'orchestre entre modèle et vue)

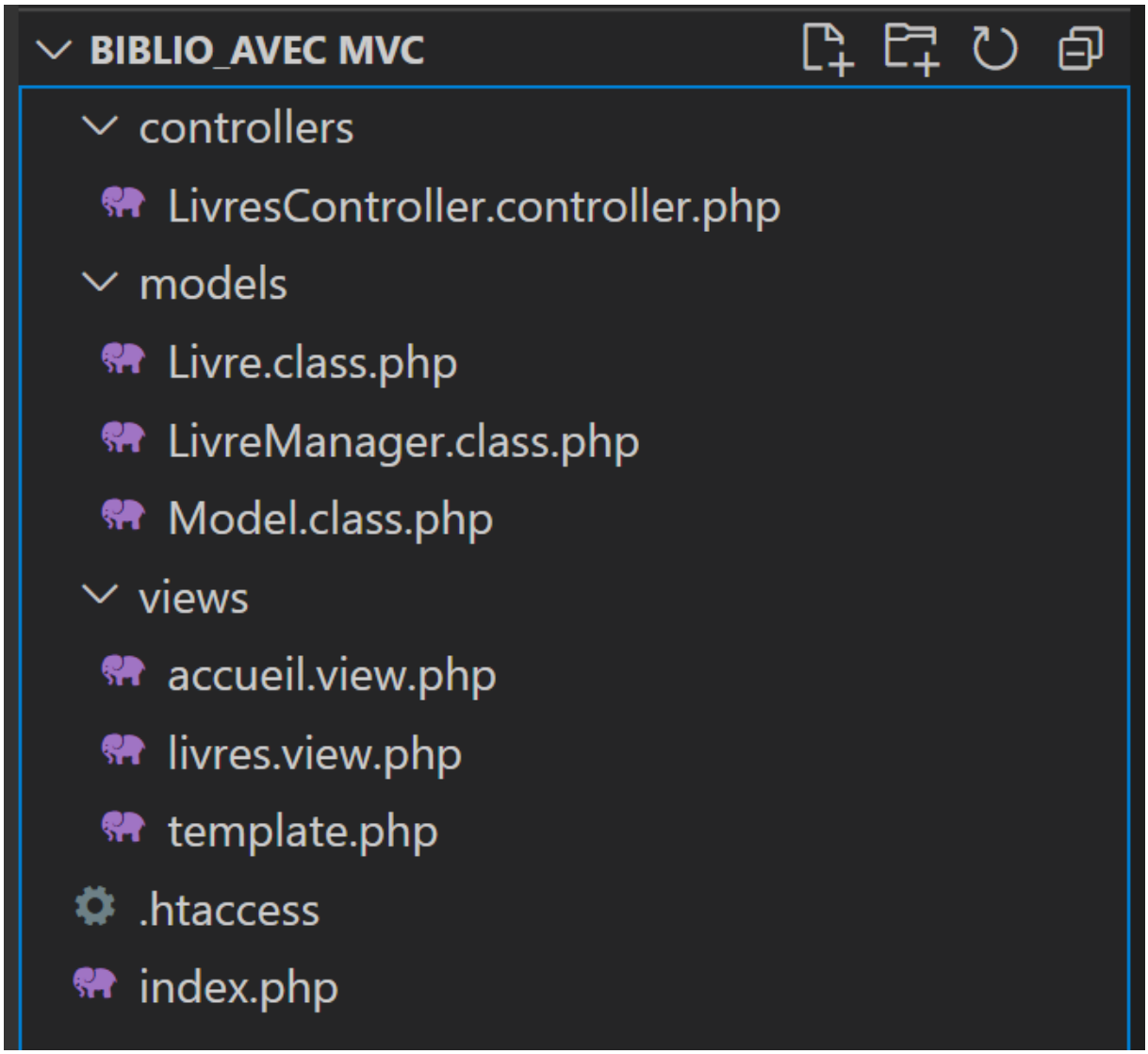
- La vue (V) (template) qui organise la présentation (affichage).

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 > ...

```php
<?php
ob_start();
?>

Ici la page d'accueil

<?php
$content = ob_get_clean();
$titre = "Bibliothèque MGA";
require "template.php";
?>
```

Le code source après routage

```php
<?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

```
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

```php
<?php

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

Changer le lien vers la pages livres

template.php

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="https://bootswatch.com/4/sketchy/bootstrap.min.css">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-label="Toggle navigatio
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarColor01">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item">
                    <a class="nav-link" href="accueil">Accueil</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="livres">Livres</a>
                </li>
            </ul>
        </div>
    </nav>

    <div class="container">
        <h1 class="rounded border border-dark p-2 m-2 text-center text-white bg-info"><?= $titre ?></h1>
        <?= $content ?>
    </div>

    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="a
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin=
</body>
</html>
```

Le contenu de livres.view.php

livres.view.php

```php
<?php

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

ob_start();
?>

<table class="table text-center">
    <tr class="table-dark">
        <th>Image</th>
        <th>Titre</th>
        <th>Nombre de pages</th>
        <th colspan="2">Actions</th>
    </tr>
    <?php
    $livres = $livreManager->getLivres();
    for($i=0; $i < count($livres);$i++) :
    ?>
    <tr>
        <td class="align-middle"><img src="public/images/<?= $livres[$i]->getImage(); ?>" width="60px;"></td>
        <td class="align-middle"><?= $livres[$i]->getTitre(); ?></td>
        <td class="align-middle"><?= $livres[$i]->getNbPages(); ?></td>
        <td class="align-middle"><a href="" class="btn btn-warning">Modifier</a></td>
        <td class="align-middle"><a href="" class="btn btn-danger">Supprimer</a></td>
    </tr>
    <?php endfor; ?>
</table>
<a href="" class="btn btn-success d-block">Ajouter</a>

<?php
$content = ob_get_clean();
$titre = "Les livres de la bibliothèque";
require "template.php";
?>
```

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
<?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

LivresController.controller.php

controllers > LivresController.controller.php > ...

```php
<?php
require_once "models/LivreManager.class.php";

class LivresController{
    private $livreManager;

    public function __construct(){
        $this->livreManager = new LivreManager;
        $this->livreManager->chargementLivres();
    }

    public function afficherLivres(){
        $livres = $this->livreManager->getLivres();
        require "views/livres.view.php";
    }
}
```

Etape 6 : déplacer les fichiers model dans le dossier models

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