

Construction d'une architecture MVC selon un cahier des charges

- Dans la racine du serveur web **www** créer un répertoire : **projet-mvc**
- Utilisez un éditeur de texte avancé comme **VSCODE**
- Environnement technologique : un serveur **LAMP** opérationnel
- Le projet étant accessible depuis le lien : <http://mvc.dev>

PREMIERE PHASE

1. Construction de la structure des répertoires (**cls**, **controleur**, **modele**, **static**, **vue**)

C:\www\projet-mvc\ ->



2. Un contrôleur principal (**index.php**)
3. Une vue nommée "vue/**intro_vue.php**" avec un titre : "hello world" et un lien :
` page 1 `
4. Un contrôleur nommé "controleur/**intro_ctl.php**" avec un `include()` de
"vue/**intro_vue.php**"
5. Sur **index.php**, une **requête par défaut** sur "controleur/**intro_ctl.php**"
6. Dans un fichier "cls/**config.php**" Définir une constante **RACINE** avec :
`dirname(__DIR__) ;` Chercher la documentation de : `dirname()` et `__DIR__`

DEUXIÈME PHASE

1. Ajouter un autre contrôleur nommé : controleur/**page1_ctl.php**
2. Une autre vue : vue/**page1_vue.php**
3. Un modèle : modele/**page1_bd.php**
4. Dans ce modèle, **importer une donnée** sans SQL, mais avec le format **JSON** :
`$data = file_get_contents(RACINE. "/modele/data.json");`
data.json contient la donnée : "page1" : "Article de la page 1",
5. L'action : **index.php?action=page1** doit afficher la vue depuis : vue/**page1_vue.php**
6. La vue contient une **feuille de style**, tous les textes de la page1 sont **rouges** :
`<link href="static/css/page1.css" rel="stylesheet" />`

TROISIÈME PHASE

1. Sur **index.php**, optimiser le routage avec un **switch** :

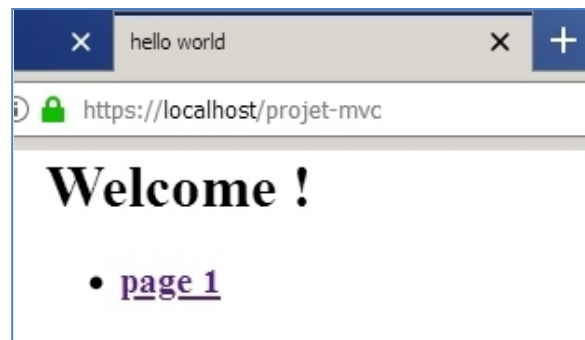
```
switch($action) {
    case "mon_action":
        //action ?
        break;
    default:
        //action ?
}
```

Ajoutez par défaut une **page 404** structurée sur le même exemple de page1.

La donnée dans "modele/**data.json**" étant : "404" : "page introuvable",

2. Ajouter le routage dans une classe nommée « **route.php** ».
 Cette classe dans le répertoire : « **cls** » (cls/**route.php**).

- Vue la page par défaut :



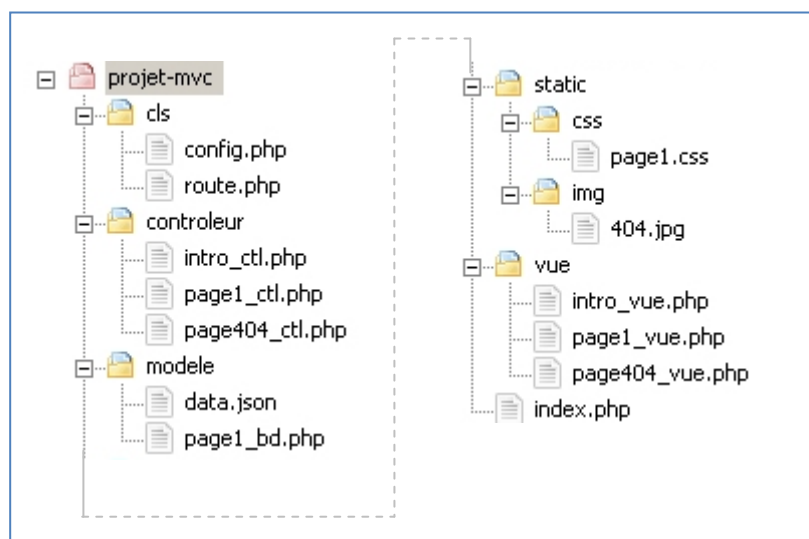
- Vue de la page 1 :



- Vue de la page 404 :



- Arborescence actuelle :



QUATRIÈME PHASE

Adaptation du projet actuel sur une base de données et une connexion PDO

Récupérer les éléments fournis à l'emplacement :

➔ **RESSOURCES : projet-mvc.zip**

- Sur PhpMyAdmin, créer une base de données nommée **projetMVC**
- Importer le fichier nommé : **projetMVC.sql** contenu dans l'archive.

➔ Adapter le projet actuel avec les fichiers fournis : **page1_bd.php** et **connect.php**

1. Remplacer les deux fichiers dans le répertoire correspondant
2. Modifier le fichier : "cls/**config.php**" et adapter le code suivant :

```
//SGBDR :  
  
const DB_USERNAME = 'root'; // nom d'utilisateur  
const DB_PASSWORD = '???'; // mot de passe de l'utilisateur  
const DB_DATABASE = '???'; // nom de la base de données  
const DB_HOST = 'localhost'; //adresse du SGBDR
```

3. Modifier la vue : " vue/**page1_vue.php** " afin d'afficher tous les articles contenus dans la base de données. Aidez-vous du modèle ci-dessous :

```
<?php  
  
if( ! is_array($article) ) {  
    echo "<h3>Aucun article !</h3>";  
} else {  
    foreach($article as $current_article) {  
  
?>  
  
<!-- ICI LES ARTICLES A AFFICHER -->  
  
<?php } } ?>
```

➔ L'itération (*la boucle foreach*) doit afficher :

- Le numéro de l'article
- L'image de l'article (contenu dans l'archive **projet-mvc.zip**)
- Le contenu de l'article
- Le nom de son auteur

4. Adapter le style de la page 1 dans : " static/css/**page1.css** "

- Résultat final du projet - les articles sont collectés depuis la base de données :



CINQUIÈME PHASE

1. Importer la librairie **Bootstrap** afin de rendre responsive au niveau front l'ensemble du projet MVC.

SIXIÈME PHASE

1. Sur la base de données, créez une table nommée "user". Ajouter les colonnes suivantes :
id | nom | prénom | email | password | avatar
Note : l'id doit être auto-incrémentée
2. Créer trois lignes de données suivantes (par défaut le mot de passe est vide) :
1, 'Léponge', 'Bob', 'robert@sponge.us', '', 'd3b8a1e9dc7e3d904ddb9f4434283602'
2, 'Curry', 'Marie', 'marie@spice.in', '', 'b190f20c3e873e62a5d46e6f621932c3'
3, 'Paulo', 'Marco', 'paulo@discover.it', '', '0b9717d93f90b7e687cb394e6a9ee2e4'
5. Adapter votre MVC afin d'ajouter un nouveau contrôleur **page2**
6. Cette page doit afficher l'ensemble de la table "user", sur le même format de la **page 1**.
7. Une image de l'avatar (par exemple d3b8a1e9dc7e3d904ddb9f4434283602.jpg) doit être adaptée pour chacun des utilisateurs.

SEPTIÈME PHASE *(note : passer cette phase en cas de difficultés majeures)*

1. Créer un nouveau contrôleur nommé **rechercher**
2. Sur la vue, ajouter un champ de recherche sur ce modèle :

Requête AJAX :

Rechercher un utilisateur

Résultat

NOTE : Utiliser une requête de type AJAX en JavaScript (avec *FETCH* ou le plugin *JQUERY*).

3. Lors de la frappe (dans le champ de recherche) le prénom de l'utilisateur doit s'afficher dans la partie **Résultat**.

HUITIÈME PHASE

1. Créer un nouveau contrôleur nommé : **login**
2. Rechercher dans la documentation PHP les méthodes : **password_hash()** et **password_verify()**
3. Ajouter dans la base de données **un mot de passe** pour chaque utilisateur
 - Ajouter sur la table "user" une colonne **paswd**
 - Utiliser la méthode suivante afin de générer un mot passe haché :
 - `password_hash("superSecret", PASSWORD_DEFAULT);`
 - Insérer manuellement un mot de passe haché pour chaque utilisateur.
4. Ajouter une vue **login** avec deux champs et un bouton :
 - Un champ : email
 - Un champ : mot de passe
 - Un bouton : valider
5. Lors de l'action "**valider**" vous devez :
 - Valider la présence de l'email utilisateur dans la BD.
 - Valider le mot de passe saisi. Utiliser la méthode suivante :

```
// $hash représente le mot de passe haché stocké dans la BD,
// soit précisément la colonne password :
if( password_verify( $_POST['password'], $hash ) ) {
    //mot de passe valide
} else {
    //mode de passe invalide
}
```

6. Si l'email et le mot de passe sont validés, **activer une session** :
 - `$_SESSION['user'] = ['nom' => $nom, 'prenom', $prenom];`
7. Afficher sur l'ensemble des vues de votre MVC, le prénom et le nom de l'utilisateur connecté grâce à la session **user**.