Compte rendue

Formation MVC via OpenClassrooms

Comment professionnaliser sont code ?

Il est modulaire :	Coder avec de nombreux fichier mais qui chacun leurs rôles spécifiques.
Il est découpé :	Développer de façon a ce que chaque fichier fonctionne indépendamment des autres.
Avoir une documentation :	Il est important d'avoir une documentation avant les méthodes afin de savoir le fonctionnement des méthodes et leur objectif.
Utiliser l'Anglais :	Utiliser l'anglais le plus possible car l'anglais est la langue qu'utilise les développeurs.
Être claire :	Utiliser un style de langage utiliser par tout développeurs.

Ce style de développement apporte de nombreux avantages :

Il est réutilisable :	Permet à toute développeurs de reprendre le projet sans avoir a perdre du temps a comprendre ou recommencer de zéro.
Facilité de travail :	Permet de travailler sur le projet avec une équipes sans soucis de compréhension.
Il est évolutif :	Cela permet en cas d'évolution de simplifier l'intégration de nouveau contenue sans risque de détruire le code et de le rendre non fonctionnel.

Afin de réaliser des codes qui fonctionne mais aussi professionnel on utilise des Frameworks qui vas aidez à mettre cela en place plus facilement.

Les limite du développement débutant :

Lorsque l'on est débutant le principal souci est de réaliser l'entièreté de sont projet dans un seul est unique fichier et comme dit dans la formation « Plus le projet est grand, plus les problèmes de code sont impactant. ».

Cela indique que le travaille en équipe est très difficile voire irréalisable.

L'lorsque les conventions de codage ne sont pas respecter, cela apporte des difficultés pour la lecture du document, utilisation de multiples dans langages pour des utilisations différentes et encore lors de la reprise du code qui seras très compliquer.

Isoler l'affichage du traitement PHP :

Une première étape pour rendre son code professionnel est d'isoler la partie du code responsable de générer l'affichage pour l'utilisateur.

Les variables fournies aux templates doivent être le plus simple possible, pour rendre plus facile le travail sur l'interface utilisateur.

Les gabarits d'affichage peuvent contenir du code – et c'est d'ailleurs ce qui fait toute leur puissance – mais celui-ci n'a pour rôle que d'afficher des informations.

Isolez l'accès aux données :

Le code qui accède aux données fait partie d'un second lot de code à isoler, à côté du code gérant l'affichage.

Il y a des morceaux de code en PHP qui ne font rien par eux-mêmes et qui se mettent simplement à disposition des autres morceaux de code.

Il y a un troisième lot de code à isoler. Il est responsable de gérer la requête HTTP de l'utilisateur, d'orchestrer la lecture des données et de renvoyer la réponse HTML.

Soigné la cosmétique du code :

Il est recommandé de garder uniquement la balise ouvrante`<?php` dans les fichiers qui contiennent seulement du PHP. Ceux qui contiennent aussi du HTML ne changent pas.

La syntaxe des "short open tags" peut être utilisée pour avoir des gabarits d'affichage plus concis.

En tant que développeur PHP, vous êtes aussi responsable de la bonne gestion de la base de données. Vous pouvez la traiter comme votre code : des noms clairs, de l'anglais…

Quiz

2 Fautes sur 8

Découvrez comment fonctionne une architecture MVC

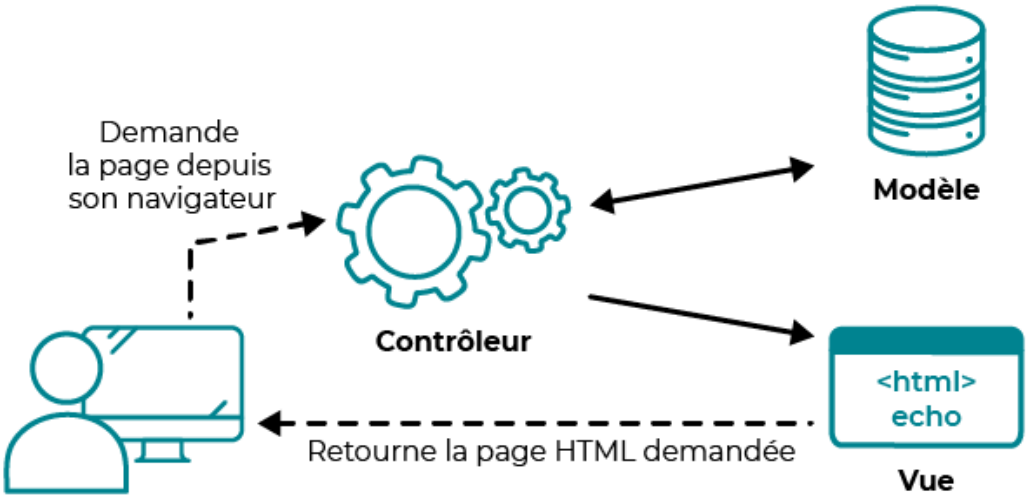
Le modèle est divisé en trois parties :

Modèle : cette partie gère ce qu'on appelle la **logique métier** de votre site. Elle comprend notamment la gestion des données qui sont stockées, mais aussi tout le code qui prend des décisions autour de ces données.

Vue : cette partie se concentre sur l'**affichage**. Elle récupère des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML et quelques boucles et conditions PHP, pour afficher une liste de messages.

Contrôleur : cette partie gère les **échanges** avec l'utilisateur. C'est en quelque sorte l'intermédiaire entre l'utilisateur, le modèle et la vue. Le contrôleur va recevoir des requêtes de l'utilisateur. Les requête envoyer au modèle vont par la suite être interpréter par la vue pour renvoyer un affichage à l'utilisateur.

La communication entre les fichiers :



En résumé :

Les design patterns – ou patrons de conception – sont des méthodes de codage reconnues, qui permettent de résoudre des problèmes récurrents.

MVC signifie "Modèle, Vue, Contrôleur". Le code est segmenté selon ces trois sections :

- Le modèle contient le code qui gère la logique métier
- La vue celui qui gère l'affichage
- Le contrôleur gère le lien avec l'utilisateur.

Affichez des commentaires :

1) Crée la vue :

Avant de commencer à développer, il nous faut crée un dossier qui accueillera tous les fichiers d'affichage.
Une fois cela fait on peut commencer le codage de la vue.
Exemple :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Le blog de l'AVBN</title>
        <link href="style.css" rel="stylesheet" />
    </head>

    <body>
        <h1>Le super blog de l'AVBN !</h1>
        <p><a href="index.php">Retour à la liste des billets</a></p>

        <div class="news">
            <h3>
                <?= htmlspecialchars($post['title']) ?>
                <em>le <?= $post['french_creation_date'] ?></em>
            </h3>

            <p>
                <?= nl2br(htmlspecialchars($post['content'])) ?>
            </p>
        </div>

        <h2>Commentaires</h2>

        <?php
        foreach ($comments as $comment) {
        ?>
        <p><strong><?= htmlspecialchars($comment['author']) ?></strong> le <?= $comment['french_creation_date'] ?></p>
        <p><?= nl2br(htmlspecialchars($comment['comment'])) ?></p>
        <?php
        }
        ?>
    </body>
</html>
```

2) Création du contrôleur

Afin de crée le contrôleur dans la logique du MVC, on va crée un dossier contrôleur tout les fichiers de ce type se retrouveront.

Une fois cela fait, on peut développer notre contrôleur exemple :

```php
<?php
// post.php

require('src/model.php');

if (isset($_GET['id']) && $_GET['id'] > 0) {
    $identifier = $_GET['id'];
} else {
    echo 'Erreur : aucun identifiant de billet envoyé';

    die;
}

$post = getPost($identifier);
$comments = getComments($identifier);

require('templates/post.php');
```

3) Mise à jour du modèle

Si le fichier n'est pas encore réalisé, faite un dossier nommé modèle et commencer a coder.

Exemple :

```php
function getPost($identifier) {
    try {
        $database = new PDO('mysql:host=localhost;dbname=blog;charset=utf8', 'blog', 'password');
    } catch(Exception $e) {
        die('Erreur : '.$e->getMessage());
    }

    $statement = $database->prepare(
        "SELECT id, title, content, DATE_FORMAT(creation_date, '%d/%m/%Y à %Hh%imin%ss') AS
french_creation_date FROM posts WHERE id = ?"
    );
    $statement->execute([$identifier]);

    $row = $statement->fetch();
    $post = [
        'title' => $row['title'],
        'french_creation_date' => $row['french_creation_date'],
        'content' => $row['content'],
    ];

    return $post;
}
```

En résumé :

`src/model.php`: le modèle, qui contient différentes fonctions pour récupérer des informations dans la base.

`index.php`: le contrôleur de la page d'accueil. Il fait le lien entre le modèle et la vue.

`templates/homepage.php`: la vue de la page d'accueil. Elle affiche la page.

`post.php`: le contrôleur d'un billet et ses commentaires. Il fait le lien entre le modèle et la vue.

`templates/post.php`: la vue d'un billet et ses commentaires. Elle affiche la page.

Crée un template de page :

1) Inclure des blocs de pages

Les blocs de pages : Deux fichier comme le header et footer qui seront réutiliser pour chaque vue.

Que l'on fasse appel à chaque vue avec un required('header.php')

2) Créez un layout

Un layout est « L'armature de la vue » c'est l'équivalent d'un texte a trou, il ne suffira uniquement de remplir les trous avec les variables.

Exemple de layout :

```php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title><?= $title ?></title>
        <link href="style.css" rel="stylesheet" />
    </head>

    <body>
        <?= $content ?>
    </body>
</html>
```

En résumé :

Un système de templates est un système où des morceaux de vue sont paramétrables.

On peut utiliser un template de "mise en page", appelé *layout*, pour factoriser le code HTML redondant.

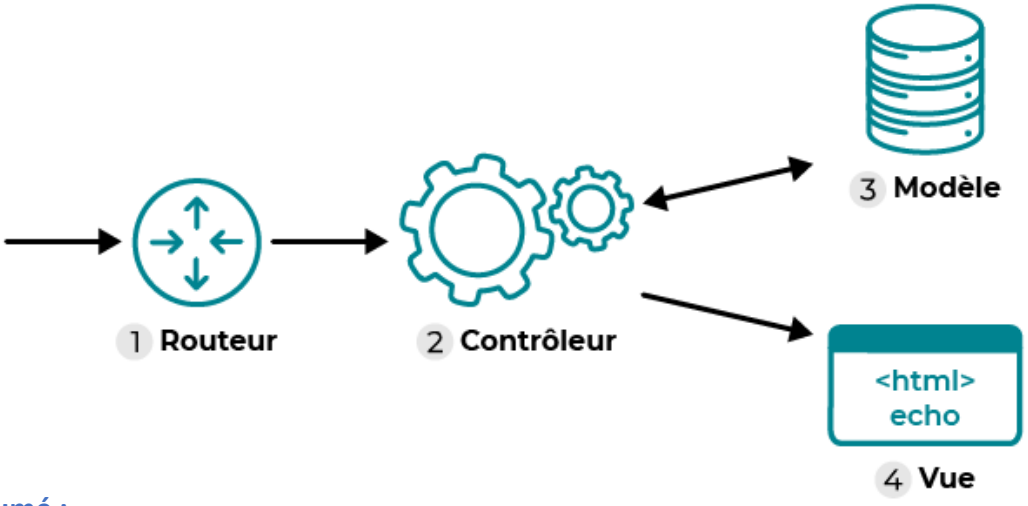
Les fonctions `ob_start()` et `ob_get_clean()` ne sont pas nécessaires, mais aident à implémenter un système de templates.

Crée un routeur :

Routeur = index

Il permet d'avoir la redirection sur chaque fichier utile qui permet de retrouver le plus simplement ces fichiers.

Exemple :



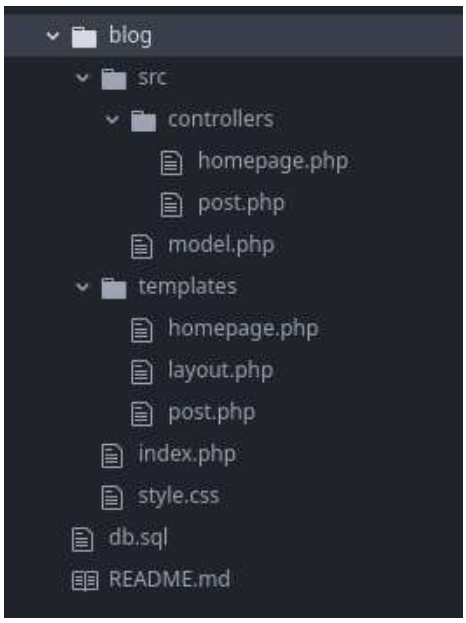
En résumé :

On préfère créer un fichier par contrôleur, tous rassemblés dans un même dossier. À l'intérieur, chaque fichier définit une fonction, qui sera appelée par le routeur.

Quand on fonctionne avec des fichiers PHP de type "bibliothèque de code", il faut utiliser `require_once` pour éviter des plantages.

Organiser en dossier :

Exemple :



Identifié l'utilité du document afin de créer les dossiers en liens avec les fichiers.

Ordre de création et mise a jours lors d'une création de fichier :

- ➔ Création de la vue
- ➔ Création ou mise à jour contrôleur
- ➔ Mise a jour du routeur / index
- ➔ Mise à jour du modèle

Gérer les erreurs :

Utiliser Exception $e qui permet lors de sont utilisation dans catch permet de dire il y a une erreur.

Les exceptions sont un mécanisme qui permet de gérer les remontées d'erreurs en PHP. Elles s'utilisent avec les trois mots clés : `try`, `catch` et `throw`.

`Try` et `catch` permettent de créer des zones de contrôles sur les exceptions.

`Throw` permet de lancer une exception. Elle interrompra le flux classique d'exécution du code, jusqu'à être gérée par une zone de contrôle d'exception.

Les zones de contrôle d'exception peuvent s'imbriquer à l'infini.

Le Throw permet de mieux identifié ou se situe l'erreur que Die.

Quiz

4 Fautes sur 8

Structuré vos données :

La conception MVC utilise 3 couches de code. Les interactions entre chacune des couches doivent être structurées et documentées.

En créant vos propres types grâce aux classes, vous saurez exactement ce qu'ils vont faire.

Utiliser des noms de variable qui signifie exactement à quoi ils servent.

On peut créer des types personnalisés en PHP, grâce aux classes. Chaque classe doit avoir un nom unique. Chaque classe sera composée avec des propriétés ayant d'autres types (simples ou personnalisés).

L'action de créer des variables à partir de ces types personnalisés s'appelle "instancier une classe". On appelle ces instances des "objets".

Les propriétés d'un objet peuvent être obtenues grâce à l'opérateur flèche->. Cet opérateur s'utilise aussi bien pour lire la valeur d'une propriété que pour y écrire une nouvelle valeur.

Conclusion :

Le modèle MVC est utile afin de menée a bien a projet en groupe et de façon organisée en répartissant chaque action a un fichée, cette architecture permet alors aussi une possibilité d’améliorations importante par le biais de la création d’un contrôleur, vue et de nouveaux script dans le modèle pour ajouter une évolutions a une application. Elle permet aussi de mieux répartir sont code avec une sections front avec les vues, Back avec les contrôleur et serveur avec le modèle.