L’architecture MVC

Ou comment découper son code en lui attribuant des rôles

Référence pour les citations et même un peu de contenu très bien écrit, je vous invite à le lire même si ce document va synthétiser ce cours : https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php

1. Pourquoi adopter une autre architecture ?

Jusqu’ici nous avons mis en pratique différents langages dans ce que l’on appelle une « stack technique ». Une stack technique consiste en une liste de technologies mises en commun pour un objectif de développement, dans notre cas, nous jonglons entre SQL, PHP, HTML, CSS, et Javascript.

Avec cette même stack technique, on peut développer d’une multitude de façons différentes. Jusqu’ici nous avons développé principalement autour de PHP, qui nous sert de pont entre la base de données et l’affichage en HTML, dans ce que l’on va appeler du **PHP Procédural**, c’est-à-dire que chaque fichier exécute son code dans l’ordre, de haut en bas, en appelant potentiellement des fonctions mais qui n’utilise pas de programmation orientée objet.

Cela présente plusieurs limites :

- Le code est parfois difficile à lire. En effet, si dans un seul fichier on peut se retrouver avec tous ces langages en même temps, il peut être compliqué dans savoir quel bout de code fait quoi.
- Il fonctionne, certes. Mais si vous commencez à ajouter de nouvelles fonctionnalités, vous allez avoir besoin d'ajouter de nouvelles requêtes SQL au milieu. Très vite, le fichier va grossir et faire plusieurs centaines de lignes. Ça marchera toujours. Ça ne sera même pas forcément plus lent. Par contre, à chaque fois que vous allez ouvrir votre fichier, vous allez pousser un **grand soupir** :

"Pfff, qu'est-ce que j'avais fait déjà ? Hmm, là c'était à peu près l'endroit qui s'occupait de ça. Bon je modifie ici... ah merde ça plante. Ah oui, c'est vrai, il y avait aussi un truc à modifier à un autre endroit du fichier pour que ça marche. Oh là là, ça fait déjà 4 heures que j'essaye de faire ça, je suis perdu, c'est trop compliqué la programmation !"

- Et bien sûr, n'oubliez pas que la plupart du temps les projets se font en équipe. Jenny, qui fait l'intégration HTML et CSS du site, va devoir naviguer entre vos fonctions PHP. De votre côté, vous vous perdrez au milieu de ses imbrications de balises. Et au moment de rassembler votre travail, ce sera la fête des conflits Git !

C’est ici qu’on va avoir le choix dans notre façon de travailler : Soit on continue en procédural : il est important de comprendre que ça n’est pas **mal** de travailler comme on le fait ici, soit on va se diriger vers du code plus normé, plus standardisé comme dans un framework par exemple. C’est ici qu’arrive le paradigme MVC.

2. C’est quoi ?

MVC est un acronyme désignant les trois principaux rôles que l’on va attribuer à nos fichiers :

Le MODÈLE :

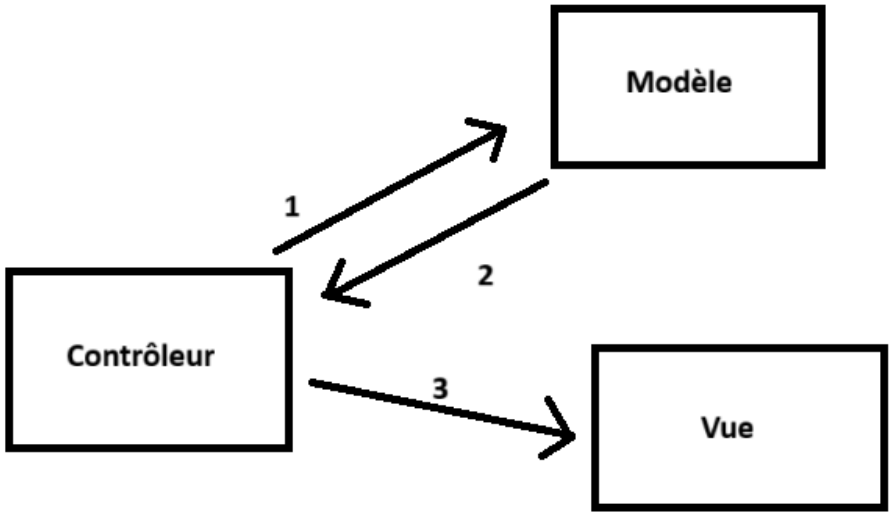
Il va contenir la structure de nos entités ainsi que toutes les interactions avec la base de données

La VUE :

Elle va servir de réceptacle au contenu du site : il faut visualiser ça comme une série de différentes balises HTML prévues pour contenir des données.

Le CONTRÔLEUR :

Il va servir de passerelle entre le modèle et la vue. Le contrôleur va aller chercher les données du modèle, puis récupérer ces données et les transférer à la vue qui accueillera ces données dans les emplacements prévus.



Exemple avec le blog : L’index est le contrôleur, il va chercher d’abord le modèle, puis il va afficher la vue « homepage » qui prendra les données du modèle.

Restrictions :

L’intérêt de découper son code comme ceci consiste à organiser et à structurer les rôles de chacun de ces fichiers. Le contrôleur est le chef d’orchestre qui lie les modèles à des vues.

Le modèle est permissif, mais il faut cependant garder deux choses à l’esprit :

- Il n’y est pas censé avoir de requêtes SQL dans la vue, c’est le rôle du modèle
- Il n’y est pas censé avoir de HTML dans le modèle, c’est le rôle de la vue.