Application en couches et modèle MVC

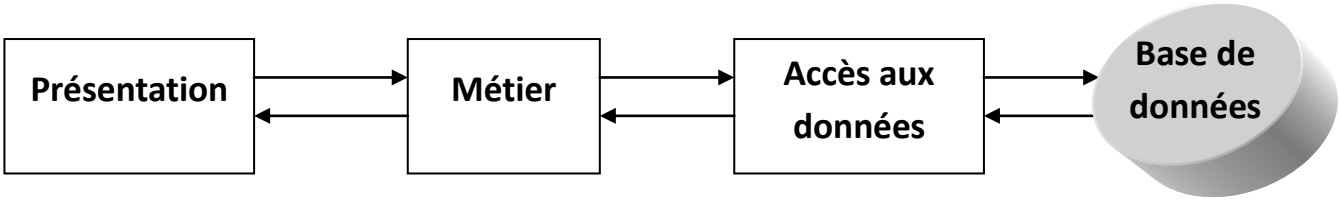
F. Lemainque
Version 1.3
Septembre 2017

1. Architecture en couches

1.1 Introduction

L'architecture en couches est souvent appelée architecture en 3 *tiers*, d'après le terme anglais *tier* signifiant étage ou niveau. C'est un modèle logique d'architecture qui modélise une application comme un empilement de trois couches logicielles au rôle clairement défini :

- **Présentation des données** : c'est l'affichage sur le poste de travail de l'utilisateur. Aussi nommée " couche graphique ";
- **Traitement des données** : c'est la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative. Aussi appelée " couche métier " ;
- **Accès aux données** : c'est le mécanisme d'accès aux données devant être conservées. Aussi nommée " couche d'accès aux données " ou DAO (*Data Access Objet*).



Les couches ne communiquent qu'avec leurs voisines immédiates. Le rôle de chaque couche étant parfaitement défini, les fonctionnalités de chacune d'entre elles peuvent évoluer sans conséquence aucune pour les autres couches.

1.2 Détails des couches

Couche de présentation

Elle correspond à la partie de l'application qui sert à l'interaction avec les utilisateurs. Elle peut par exemple être représentée en HTML/CSS pour être exploitée par un navigateur web ou un téléphone portable. Elle peut revêtir de nombreuses formes pour une même finalité d'application.

La couche de présentation relaie les requêtes de l'utilisateur à destination de la couche métier, et en retour lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.

Couche métier

C’est la partie fonctionnelle de l'application, celle qui comprend la “ logique ” et qui décrit les opérations appliquées aux données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.

Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche. La couche métier offre des services applicatifs et métier à la couche présentation. Pour fournir ces services, elle s'appuie, le cas échéant, sur les données du système, accessibles au travers des services de la couche d’accès aux données. En retour, elle renvoie à la couche présentation les résultats calculés.

Couche accès aux données

Elle gére l'accès aux données du système. Ces données peuvent être propres au système, ou gérées par un autre système, peu importe : son rôle est uniquement (!) de fournir à la couche métier les données nécessaires, et de recevoir de celle-ci les données qui doivent être pérennisées.

1.3 Intérêts de l’architecture en couches

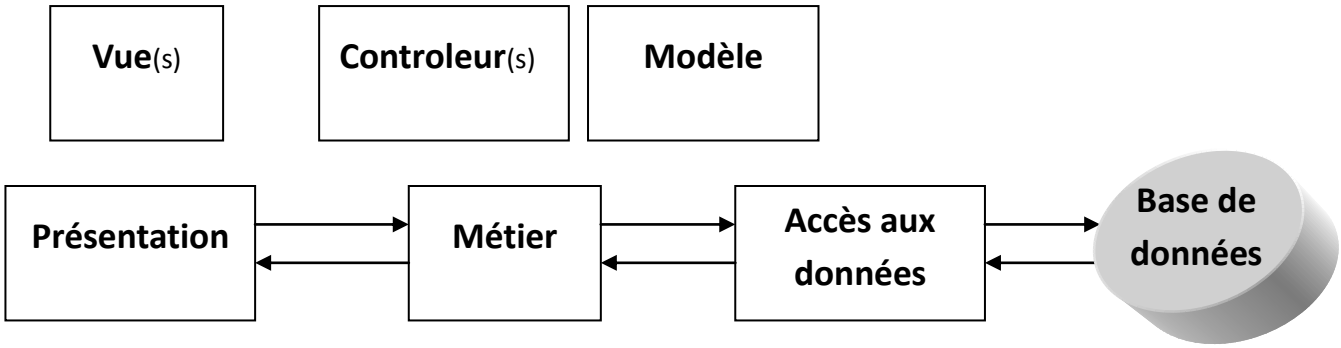
Ce modèle d'architecture a pour objectif de répondre aux préoccupations suivantes :

- Séparation possible du développement des différentes couches, selon les compétences propres de programmeurs ;
- Allègement du poste de travail client (une évolution du schéma client-serveur, d’un client lourd à un client léger);
- Prise en compte de l'hétérogénéité des plates-formes (serveurs, clients, langages, etc.) ;
- Amélioration de la sécurité des données, en supprimant tout lien entre le client et les données. Le serveur doit vérifier l'intégrité et la validité des données avant de les envoyer dans la couche de données.
- Rupture du lien de propriété exclusive entre application et données. La base de données peut être plus facilement normalisée et intégrée à un entrepôt de données, ou même intégralement modifiée sans nécessiter une refonte complète de l’application : seule la couche d’accès aux données sera impactée.
- Enfin, meilleure répartition de la charge entre différents serveurs d'application.

2. Modele MVC

2.1 Présentation

Le modèle MVC (Modèle-Vue-Contrôleur) est une approche voisine de l'architecture en couches. Comme celle-ci, elle cherche à séparer nettement les couches de présentation, de traitement et d'accès aux données. La couche de traitement est toutefois séparée en deux éléments, le Contrôleur et le Modèle, tandis que la couche d'accès aux données est intégrée au Modèle.



Le traitement d'une demande d'un client se déroule selon les étapes suivantes :

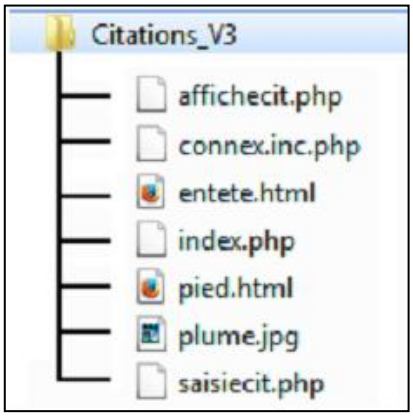
- A partir d'une vue, le client fait une demande à un contrôleur. Ce contrôleur est la porte d'entrée de l'application, le C de MVC.
- Le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin du Modèle (le M de MVC) qui correspond à une partie de la couche métier.
- Le contrôleur reçoit une réponse du modèle. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. La réponse du contrôle s'exprime à l'aide d'une Vue (le V de MVC) envoyée au client.

L'architecture MVC est bien adaptée à des applications web écrites avec des langages orientés objet. C'est d'ailleurs le modèle retenu dans la plupart des charpentes (*framework*) PHP, si bien que se familiariser avec ce modèle peut largement faciliter les choses par la suite.

2.2 Une démarche de développement MVC en Web/PHP

Lors d'un développement " classique ", les fichiers ont tendance à présenter un savant mélange de PHP, de HTML et parfois de Javascript. Le CSS a lui normalement déjà été placé dans une feuille de styles externe.

De ce fait, l'arborescence d'une application simple pourrait se présenter comme suit :



Traditionnellement, chaque fichier correspond plus ou moins à une page de l'application. Certains fichiers sont " inclus " : des en-têtes, des menus, des pieds de page. Il s'agit ici des deux fichiers HTML. De même, une bonne pratique consiste à créer un fichier de variables pour cette application, ce fichier comportant parfois la fonction de connexion à la base de données : c'est ici le fichier `connex.inc.php`. En revanche, les autres fonctions, lorsqu'elles existent, sont réparties dans les différentes pages. Chaque page constitue ainsi à la fois la présentation et la logique de traitement.

Dans le cadre d'une architecture MVC, vous allez créer des répertoires distincts pour les trois types de composants : Modèle, Vues et Contrôleurs. Il est fréquent de créer en outre un répertoire pour les images, et éventuellement un autre pour les divers fichiers (CSS, scripts, etc), mais le principal est de disposer des trois répertoires principaux.

Le principe de développement est assez simple :

- Un fichier de vue, qui sera placé dans le répertoire du même nom, ne devra comporter que des instructions de présentation : du code HTML, plus éventuellement un peu de PHP pour récupérer des informations dynamiques. Des structures `fetchAll/foreach` ou `while/fetch` y sont fréquentes. Une vue ne doit comporter aucun code de traitement de données (sauf éventuellement du code de vérification/prévention).
- Un fichier de contrôle, placé dans le répertoire Contrôleur, effectue le traitement des informations prodiguées par une vue et renvoie les informations traitées à cette vue ou à une autre. Il ne doit contenir toutefois aucune fonction, mais uniquement des appels vers celles-ci : toutes les fonctions doivent être placées dans le (ou les) modèle(s).

Vous comprenez immédiatement que cela aboutit à une multiplication des fichiers. C'est le prix à payer pour disposer d'une meilleure souplesse et adaptabilité.

Les vues

Il faut commencer par définir toutes les vues de l'application, autrement dit les pages web présentées à l'utilisateur. Il existe trois types de vues :

- **Vue formulaire de saisie** : il sert à obtenir des informations de l'utilisateur. Celui-ci dispose en général d'un bouton pour envoyer au serveur les informations saisies.
- **Vue page de réponse** : elle procure des informations à l'utilisateur. Des liens (souvent un menu) permettent à celui-ci d'atteindre d'autres pages de l'application.
- **Vue mixte** : la page envoyée par le contrôleur contient des informations, mais permet au client de fournir au contrôleur d'autres informations.

Chaque vue doit constituer une page PHP distincte. D'après la maquette de la page, il faut déterminer quelles sont les parties dynamiques de la page. Il peut s'agit d'informations fournies par le contrôleur en paramètres à la vue PHP ou de données de saisie à transmettre au contrôleur de traitement. Il est donc intéressant de noter les E/S de chaque vue.

La page finale envoyée au client n'est presque jamais une vue mais une composition de vues : des vues élémentaires assemblées en une unique page HTML.

Chaque vue doit être testée, ainsi que toutes les vues élémentaires. C'est aussi le moment d'élaborer les premiers éléments des feuilles de style utilisées.

Les contrôleurs

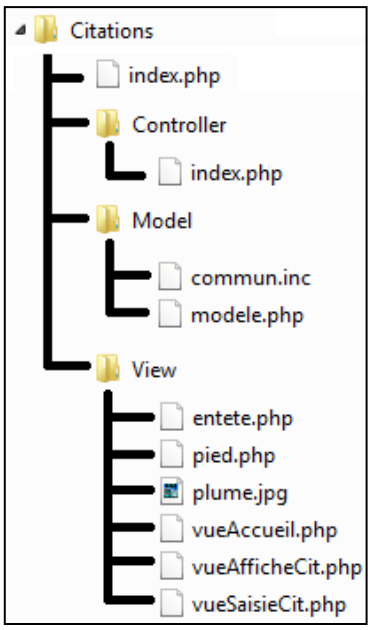
Un contrôleur gère une ou plusieurs actions. Un contrôleur traite généralement une action en appelant une ou plusieurs fonctions, situées dans un (ou le) modèle.

Il est fréquent de vouloir disposer d'un contrôleur " unique " : il faut alors disposer d'un moyen de lui transmettre l'action à exécuter. Cela est généralement effectué à l'aide de la méthode HTML `Get`, puis de l'exploitation de la variable PHP superglobale `$_GET[]` à l'aide d'une instruction `switch`. Le code des actions est soit présent dans le contrôleur, soit chargé spécifiquement avec un `include`. Dans tous les cas, le développement de traitement d'une action peut être effectué de façon relativement indépendante.

Il est capital de placer le code métier ou le code d'accès aux données (dans les deux cas en les transformant en fonctions) dans un module distinct. Le contrôleur doit être le chef d'orchestre : il reçoit des demandes des clients et les fait exécuter par les fonctions métier.

Le modèle

Le modèle regroupe les fonctions ou les classes métier dont se sert le contrôleur. En POO, c'est le développement classique de classes PHP, le plus souvent totalement indépendant de toute application web. Il est bien sûr important de tester chaque fonction pour s'assurer de son bon fonctionnement.



Parvenu à ce stade, l'arborescence de votre projet est bien différent : on peut y retrouver aisément les différents composants.
Remarquez toutefois la présence de deux fichiers portant le même nom : un à la racine de l'application, l'autre dans le répertoire `Controller`. Le fichier dans `Controller` pourrait se nommer `main` ou `accueil`. Il s'agit du contrôleur de la page d'accueil du site (de l'application), donc effectivement traditionnellement nommé `index.php`. Mais pour masquer auprès de l'utilisateur la complexité de l'arborescence, mieux vaut créer un nouveau contrôleur, placé à la racine du site et nommé `index.php` dont le rôle se limite à appeler le vrai premier contrôleur à l'aide d'un `include`.

2.3 Conclusion

L'architecture MVC présente le grand intérêt de matérialiser les principes de l'architecture en couches, en séparant au maximum les différentes composantes d'une application. Ce principe étant employé par la plupart des *frameworks* PHP, il est capital de bien le comprendre.
La pratique la plus courante, que l'on retrouve dans les charpentes, est la création d'un contrôleur générique unique réalisant ses `include` à partir du fichier de paramètres de l'application, et chargeant éventuellement (là aussi par des `include`) de façon dynamique des contrôleurs dédiés selon l'action qui doit être effectuée.