eVOGUE

Liam TARDIEU

www.evogue.fr

SUPPORT DE COURS / MVC

Sommaire

➢ Présentation

Définition

Le Modèle-Vue-Contrôleur (en abrégé MVC, de l'anglais Model-View-Controller) est une architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle. Ce paradigme divise l'IHM en un modèle (modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, synchronisation), chacun ayant un rôle précis dans l'interface.

En informatique, un patron de conception, motif de conception ou modèle de conception est un concept de génie logiciel destiné à résoudre les problèmes récurrents suivant le paradigme objet. Les trois termes sont des traductions de l'expression anglophone design pattern, tentant d'exprimer la réutilisation d'un concept que l'on retrouve dans les patrons et les motifs.
Les patrons de conception décrivent des solutions standard pour répondre à des problèmes d'architecture et de conception des applications.

Historique

L'architecture MVC a été mise au point en 1979 par Trygve Reenskaug, qui travaillait alors sur Smalltalk.
* Smalltalk est un langage de programmation orienté objet, réflexif et dynamiquement typé. Il fut l'un des premiers langages de programmation à disposer d'un environnement de développement intégré complètement graphique.

➢ Architecture Modèle/Vue/Contrôleur

Approche MVC

L'organisation globale d'une interface graphique est souvent délicate. L'architecture MVC ne résout pas tous les problèmes. Elle fournit souvent une première approche qui peut ensuite être adaptée. Elle offre aussi un cadre pour structurer une application.

Ce modèle d'architecture impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.

Modèle

Le modèle représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit ou contient les données manipulées par l'application. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient. Le modèle offre des méthodes pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle sont dénués de toute présentation.

Vue

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, etc.). Ces différents événements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

Plusieurs vues, partielles ou non, peuvent afficher des informations d'un même modèle.
Une page peut être présentée sous différentes formes en fonction d'où l'on vient, d'un choix ou encore d'un événement par exemple.
La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

Elle peut être conçue en HTML, ou tout autre « langage » de présentation.

Contrôleur

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle, ce dernier avertit la vue que les données ont changée pour qu'elle se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier. Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée. Il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.

Par exemple, dans le cas d'une base de données gérant les emplois du temps des professeurs d'une école, une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

➢ Fonctionnement

Traitement

Lorsqu'un client envoie une requête à l'application :

- L'utilisateur émet une requête depuis la vue
- Le contrôleur intercepte et analyse la requête de l'utilisateur
- Le contrôleur détermine quelle partie du modèle est concernée afin d'effectuer les traitements nécessaires
- Le modèle traite les interactions avec les données, applique les règles métier et renvoie les données au contrôleur
- Le contrôleur sélectionne la vue et lui renseigne les données
- La vue présente les données à l'utilisateur

Front Contrôleur / Back Contrôleur : un seul point d'entrée

Les patterns ont une histoire et ne sont pas figés. Avec MVC chaque vue ne communique qu'avec son contrôleur. Ainsi, le pattern MVC a évolué vers le MVC2.
Dans l'architecture MVC 2, il n'existe plus qu'un seul et unique contrôleur réceptionnant toutes les requêtes clientes.
Le contrôleur unique devient le point d'entrée exclusif de l'application. Il devient alors très aisé de centraliser la gestion des accès, des droits, des statistiques ou de toute autre fonctionnalité transverse.
Le point d'entrée unique est ce que l'on appelle un Front Contrôler, il s'agit du contrôleur central, il fait le premier aiguillage vers les différents modules de l'application en appelant les contrôleurs de chaque module (Back Contrôleur). Ensuite les Back Contrôleur classiques « chargent » la vue adéquate.
Cela peut se faire via la demande d'une page dans l'url (exemple : ?page=inscription) ainsi l'internaute reste sur la même page tout au long de la navigation et ce sont les pages qui viennent à lui en fonction de sa demande.

Arborescence

Il est tout à fait correct de mettre en place un fichier index.php faisant office de Front Contrôleur et de mettre trois dossiers composés de sous dossiers portant le nom des modules de votre site :

- Index.php
- Contrôleur /
  - Membre /
  - Boutique /
- Modèle /
  - Membre /
  - Boutique /
- Vues /
  - Membre /
  - Boutique /

➢ Etude de cas

Avantages

Un avantage apporté par ce modèle est la clarté de l'architecture qu'il impose. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet. En effet, la modification des traitements ne change en rien la vue. Par exemple si votre site doit changer de sgbd, nous ne modifierons que la couche modèle et éventuellement la couche contrôleur mais en aucun cas la couche vue (apparence du site).
A l'inverse, si vous souhaitez changer le design de votre site, vous pouvez faire intervenir un intégrateur web (html/css) uniquement dans la couche vue afin qu'il opère les changements, il n'y aura aucun risque d'erreur sur « les lignes de code importantes » du site et d'interrompre malencontreusement son bon fonctionnement car elles ne sont pas contenues dans la couche vue.

Le modèle étant totalement autonome, il peut être modifié beaucoup plus facilement.
Deux équipes peuvent travailler en parallèle. Une équipe d'infographistes peut travailler sur les vues et en même temps une équipe de développeurs peut travailler sur le modèle et le contrôleur. Cet aspect nécessite tout de même une bonne communication entre les deux entités.

Inconvénients

Le MVC se révèle trop complexe pour de petites applications. Le temps accordé à l'architecture peut ne pas être rentable pour le projet.

Même si le code est factorisé, le nombre de microcomposants n'en est pas moins augmenté. C'est le prix à payer pour la séparation des 3 couches. Et le nombre important de fichiers représente une charge non négligeable dans un projet.

Le MVC montre ses limites dans le cadre des applications utilisant les technologies du web, bâties à partir de serveurs d'applications. Des couches supplémentaires sont alors introduites ainsi que les mécanismes d'inversion de contrôle et d'injection de dépendance.

Conclusion

Dans le cadre d'une petite application web peu évolutive dans le temps, la complexité en termes de code et de création de fichiers/dossiers, apportée par MVC, ne sera pas justifiée.
En revanche pour tout autre projet web, la séparation en plusieurs couches permet à différentes équipes de travailler chacune sur une couche, indépendamment des autres.
En séparant les rôles, le code est ainsi plus maintenable dans le temps et gagne sans aucun doute plus de clarté.