MVVM

**MVVM ? En théorie c’est simple, voici pourquoi**

MVVM est un design pattern ou patron de conception, très souvent utilisé ces derniers temps par des bibliothèques Javascript. On le rencontre également dans Silverlight et WPF. A l’origine, MVVM aurait été introduit par Microsoft.

Dans ce billet, je vais vous expliquer les grands principes de MVVM et comment ce pattern permet de simplifier l’écriture de vos interfaces graphiques. Grâce à ces éléments, vous comprendrez mieux comment utiliser les bibliothèques basées sur MVVM. Dans cet article, je prendrais comment référence le Web. Mais sachez que MVVM peut aussi être utilisé pour des interfaces en client lourd (exemple: WPF ou Silverlight dans .NET).

Habituellement, pour créer des interfaces (Web), le serveur va générer du code HTML. Ce code sera ensuite analysé par un navigateur Web pour afficher une page à l’écran. En interne, la page sera traduite en DOM (Document Object Model). Ce modèle est une représentation objet de l’ensemble des éléments dans notre page. C’est ce modèle qui sera ensuite manipulé par Javascript pour modifier des contrôles visuels dans la page affichée.

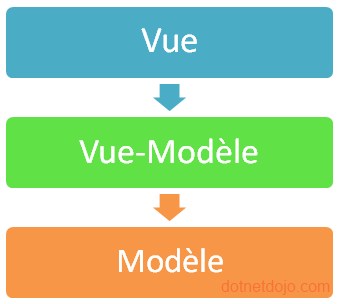
Seulement voilà, très souvent, votre page contient du code HTML mélangé avec du Javascript et finalement elle devient très vite complexe. De plus, la vue chargée d’afficher des données sera fortement dépendante des données qu’elle veut afficher. Tout est codé en dur. Bref, la maintenance n’est pas facile après quelques semaines sur le projet.

Il existe plusieurs solutions pour arranger ça. Vous pouvez par exemple utiliser des templates jQuery (des fragments de code variabilisées que vous pouvez répéter dans la page). Cette solution est sympa mais elle ne règle pas tout.

Pour aller plus loin, il faut réfléchir à un niveau d’abstraction supérieur et c’est là qu’interviennent les design patterns. Les design patterns sont des modèles de conception d’application. Ils décrivent la manière de découper et d’organiser des composants applicatifs.

Dans la série des design patterns, il existe MVVM (Model View ViewModel, ce qui veut dire Modèle Vue Vue-Modèle en français). Ce pattern a spécialement été conçu pour améliorer la séparation entre les données et la vue qui les affichent. Le lien entre la vue et le modèle de données est fait par des mécanismes de binding. Pour rappel, le binding est un mécanisme qui permet de faire des liaisons entre des données de manière dynamiques. Ce qui veut dire que si A et B sont lié, le fait de modifier A va être répercuté sur B et inversement.

**Les composantes de MVVM**



Voyons de plus prêt ce que contient MVVM :

* Model (Modèle en français) : le modèle contient les données. Généralement, ces données proviennent d’une base de données ou d’un service externe comme un API.
* View (Vue en français) : la vue correspond à ce qui est affiché (la page web dans notre cas). La vue contient les différents composants graphiques (boutons, liens, listes) ainsi que le texte.
* ViewModel (Vue-Modèle en français) : ce composant fait le lien entre le modèle et la vue. Il s’occupe de gérer les liaisons de données et les éventuelles conversions. C’est ici qu’intervient le binding.

**A retenir sur MVVM**

L’idée à retenir avec MVVM est simple : la vue ne doit jamais traiter de données. Elle s’occupe uniquement de les afficher. Le View-Model aura en charge les conversions et les accès au modèle de données.

La méthodologie MVVM permet de formaliser le développement avec WPF en garantissant des applications bien architecturées, testables et optimisant le flux de travail entre développeur et designer.

MVVM permet de tirer partie des bénéfices de la plateforme WPF tout en conservant une application correctement architecturée, maintenable et testable.

Le développement logiciel nécessite donc l'intervention de personnes différentes, que l'on peut qualifier à plus haut niveau de développeur (architecte, testeur, codeur, etc.) et designer (artiste, ergonome, animateur, etc.).

La clé de cet échange entre designer et développeur repose sur XAML. C'est au travers de ce langage à balises que les deux mondes partagent leurs travaux pour la réalisation du logiciel.

Les principes décrits dans MVVM ont pour but de maximiser cette collaboration en produisant un logiciel de qualité, maintenable et testable.

En résumé, la méthodologie MVVM peut apporter les bénéfices suivants :

* Une séparation claire entre le code métier et sa représentation graphique
* La possibilité d'utiliser efficacement et simplement les mécanismes de WPF (databinding, commandes, etc.)
* La production d'un code qui est testable
* Une organisation qui facilite et optimise le flux de travail entre développeur et designer
* Un déploiement facilité (réutilisation du travail) en multi-environnement (WPF/Silverlight)

Le but est d'obtenir une architecture la plus faiblement couplée possible.

Puisque les classes Vue-Modèle sont des classes "standard", il devient possible d'écrire des tests unitaires. En quelque sorte, les tests unitaires et la vue (en XAML) deviennent tous deux consommateurs de la classe Vue-Modèle associée.