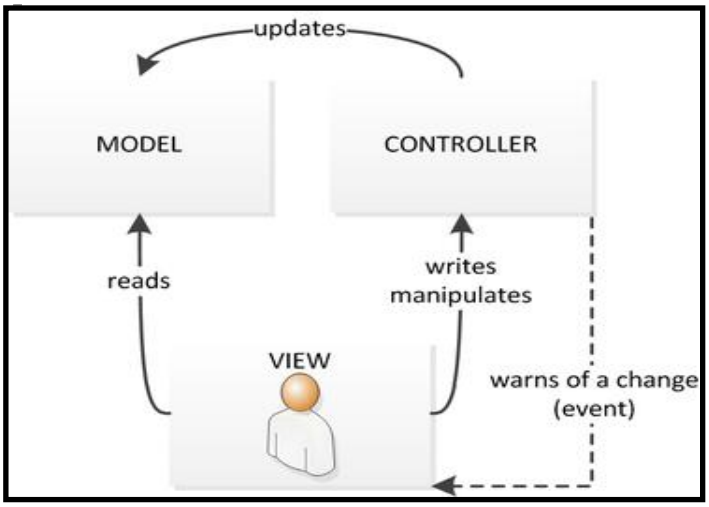
MVC

Définition

- MVC = Modèle-Vue-Contrôleur
- MVC est un patron de conception, son principe est de séparer la partie de présentation des informations (Vue), les actions de l'utilisateur (Contrôleur) et l'accès aux données (Modèle).
- MVC n'est pas lié au type d'application, ni au langage de programmation, mais il est utilisé très souvent dans la réalisation des application web (Client léger).



Modèle

- La couche Modèle représente la partie de l'application qui **exécute la logique métier**. Cela signifie qu'elle est responsable de récupérer les données, de les convertir selon des concepts chargés de sens pour votre application, tels que le traitement, la validation, l'association et d'autres tâches concernant la **manipulation des données**.
- Pratiquement, on stocke dans le modèle les attributs des données avec les méthodes qui permettent d'y accéder, de les modifier et de les sauvegarder.

Vue

- La Vue retourne une **présentation des données** venant du modèle. Etant séparée par les Objets Modèle, elle est responsable de l'utilisation des informations dont elle dispose pour produire une interface de présentation de votre application.
- La Vue utilise les données pour **fournir une page HTML** les contenant, **ou un résultat XML** formaté pour que d'autres l'utilisent. Elle peut aussi être utilisée pour offrir une grande variété de formats en fonction de vos besoins, comme les vidéos, la musique, les documents, etc.

Contrôleur

- La couche Contrôleur gère les **requêtes des utilisateurs**. Elle est responsable de **retourner une réponse** avec l'aide mutuelle des couches Modèle et Vue.

- D’une manière générale, le contrôleur va utiliser les données du modèle, les traiter en fonction de l’action de l’utilisateur, et les envoyer à la vue afin qu’elle les affiche.
- On peut créer un seul contrôleur pour toute l'application, mais en créant plusieurs (un pour chaque classe du modèle) on facilite la conception, et on rend le code plus réutilisable.

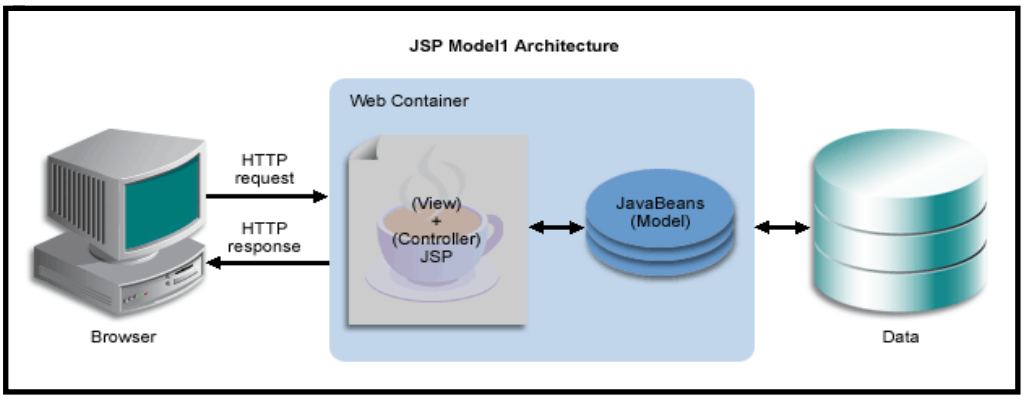
Avantages

- Concevoir des applications de manière claire et efficace grâce à la séparation des intentions.
- Simplifier la maintenance et les mises à jour.
- Répartir les tâches entre développeurs d’une façon facile et organisée.
- Faciliter les tests de l’application développée.

Inconvénients

- Couplage fort entre les composants (en MVC 1).

MVC 1



MVC 2