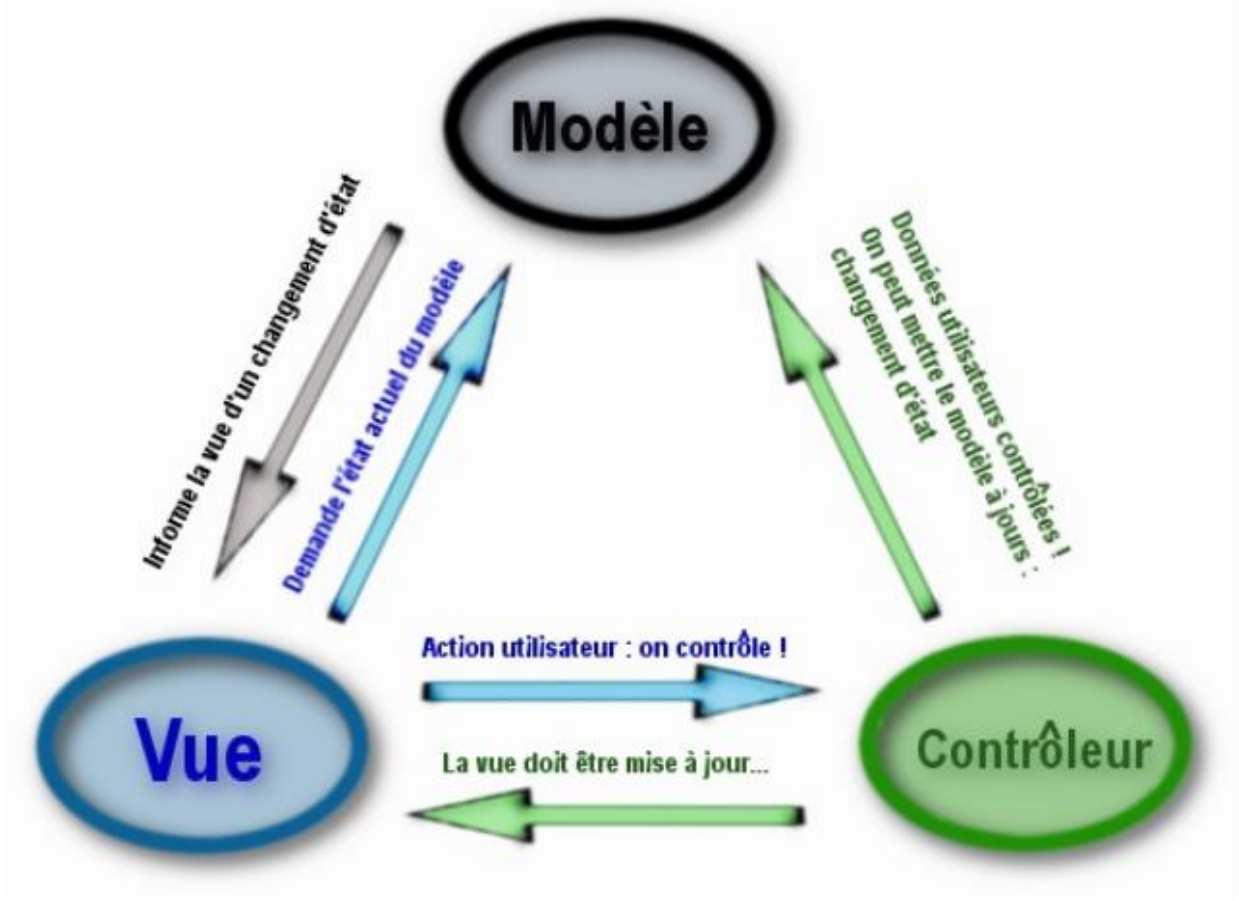
MVC



La vue

Ce qu'on nomme la vue est en fait une Interface Humain-Machine.

Celle-ci est en fait ce qu'a l'utilisateur sous les yeux. La vue peut donc être : une application graphique swing, awt, swt pour Java , une page web, un terminal Linux ou une console Windows, etc.

Le modèle

Le modèle peut être divers et varié. Il s'agit en général d'un ou plusieurs objets Java. Ces objets s'apparentent généralement à ce qu'on appelle souvent "la couche métier" de l'application. Ce sont des objets qui font des traitements absolument transparents pour l'utilisateur. Par exemple, on peut citer des objets dont le rôle est de gérer une ou plusieurs tables d'une base de données. En trois mots, il s'agit du cœur du programme!

Le contrôleur

Cet objet - car il s'agit aussi d'un objet - permet de faire le lien entre la vue et le modèle lorsqu'une interaction utilisateur est survenue sur la vue ! C'est cet objet qui aura pour rôle de contrôler les données et, le cas échéant, ira dire au modèle qu'il peut utiliser les données envoyées par l'utilisateur en toute tranquillité !

***

Pour commencer, vous pouvez constater que les trois composants sont tous en interaction les uns avec les autres. Si nous partons du postulat que votre application, implémentant le pattern MVC, est chargée, voici ce qu'il peut se passer : l'utilisateur fait une action sur votre calculatrice (clic sur un bouton) ; l'action est captée par le contrôleur qui va vérifier la cohérence des données et, le cas échéant, transformer celles-ci afin d'être comprises par le modèle. Le contrôleur peut aussi demander à la vue de changer ; le modèle reçoit les données et change d'état (une variable qui change, par exemple...) ; le modèle notifie à la vue (ou aux vues) qu'elle (qu'elles) doit (doivent) se mettre à jour ; l'affichage dans la vue est modifié en conséquence en allant chercher l'état du modèle

Le modele

Le modèle sera l'objet qui sera chargé de stocker les données nécessaires à un calcul (nombre et opérateur) et d'avoir le résultat ! Afin de prévoir un changement éventuel de modèle, nous allons créer celui-ci à partir d'un super-type de modèle : comme ça, si un changement s'opère, nous pourrons utiliser les différentes classes filles de façon polymorphe. Afin de faire des calculs simples, celui-ci devra pouvoir : récupérer et stocker au moins un nombre ; stocker l'opérateur de calcul ; renvoyer le résultat ; calculer le résultat ; tout remettre à zéro.

Le contrôleur

Celui-ci sera chargé de faire le lien entre notre vue et notre modèle. Pour la même raison que pour le modèle : polymorphisme ! Nous créerons aussi une classe abstraite afin de définir un supertype de variable pour, au cas où, utiliser des contrôleurs de façon polymorphe... C'est le contrôleur qui va intercepter les actions utilisateur, qui va modeler les données et les envoyer au modèle ! Il devra donc : agir lors d'un clic sur un chiffre ; agir aussi lors du clic sur un opérateur ; avertir le modèle pour qu'il se réinitialise dans le cas d'un clic sur le bouton 'reset' ; contrôler les données !

La vue

Voici le plus facile à développer
Nous allons créer celui-ci avec le package javax.swing .

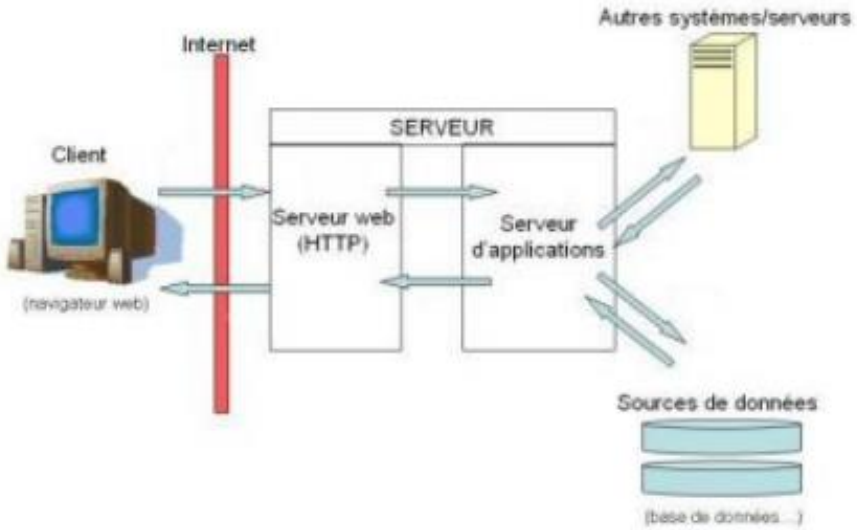
Lorsque vous cliquez sur un chiffre

L'action est envoyée au contrôleur. Celui-ci vérifie si le chiffre est conforme. Il informe le modèle. Celui-ci est mis à jour et informe la vue de ses changements. Celle-ci rafraîchit son affichage.

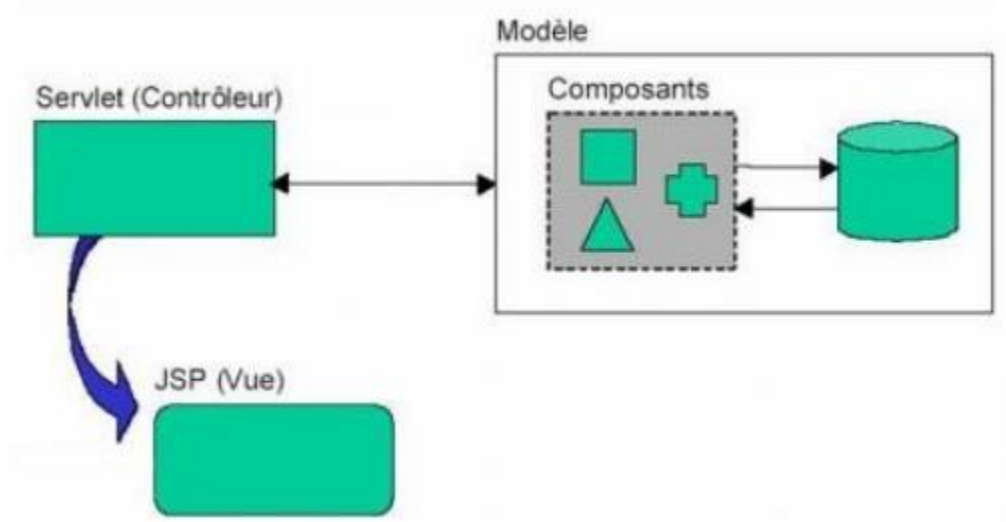
Lorsque vous cliquez sur un opérateur

L'action est toujours envoyée au contrôleur. Celui-ci vérifie si l'opérateur envoyé est dans sa liste. Le cas échéant, il informe le modèle. Celui-ci agit en conséquence et informe la vue de son changement. La vue est mise à jour... La même chose se passe lorsque nous cliquerons sur le bouton "reset"

MVC pour le web : le pattern M2VC



Le principe de fonctionnement pour le couple JSP - Servlet est le suivant :



Le schéma est assez explicite.

Vous pouvez voir que les servlets ont le rôle de contrôleurs, les JSP sont les vues et les objets métiers correspondent à votre modèle ! Cependant, un problème pouvait se poser. En effet, dans certaines boîtes de développement, les tâches sont réparties entre plusieurs développeurs : certains seront chargés de développer les objets métiers ; d'autres s'occuperont des servlets ; et enfin, une troisième équipe sera en charge des vues.

Ce qu'il faut retenir

Le pattern MVC est un pattern composé.
Celui-ci est composé du pattern observer et du pattern strategy.
L'implémentation du pattern observer permet au modèle de tenir informés ses observateurs.
L'implémentation du pattern strategy permet à la vue d'avoir des contrôles différents.
Utiliser ce pattern permet de découpler trois acteurs d'une application, ce qui permet plus de souplesse et de maintenabilité dans la durée de vie de celle-ci.
Pour la conception d'applications web, on utilise le pattern M2VC.