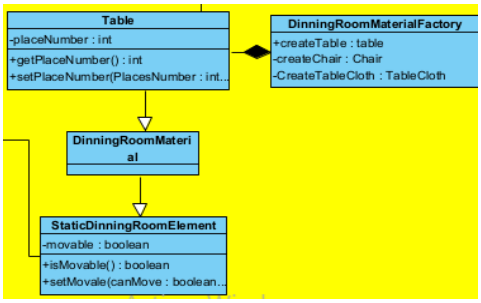
DESIGN PATTERN

Un design pattern est un schéma représentant une solution à un problème récurrent dans le monde de la programmation. Le design pattern est structure en plusieurs modèles distinctifs parmi lesquels nous pouvons citer :

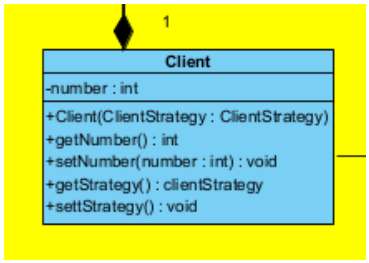
- FABRIQUE (FACTORY)

La fabrique permet de créer un objet dont le type dépend du contexte : cet objet fait partie d'un ensemble de sous-classes. L'objet retourné par la fabrique est donc toujours du type de la classe mère mais grâce au polymorphisme les traitements exécutés sont ceux de l'instance créée.



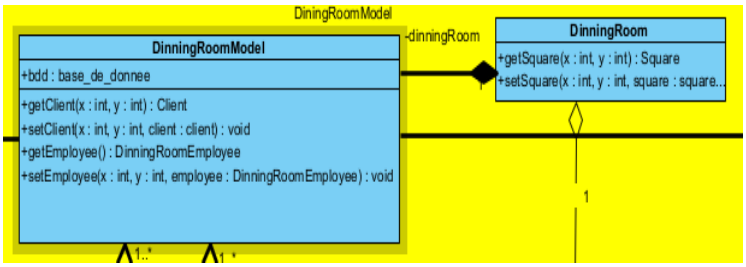
- SINGLETON (SINGLETON)

Il permet de restreindre l'instanciation d'une classe à un seul objet. Il est utilisé lorsqu'on a besoin exactement d'un objet pour coordonner les opérations dans un système. On implémente le singleton en écrivant une classe contenant une méthode qui crée une instance uniquement s'il n'en existe pas encore.



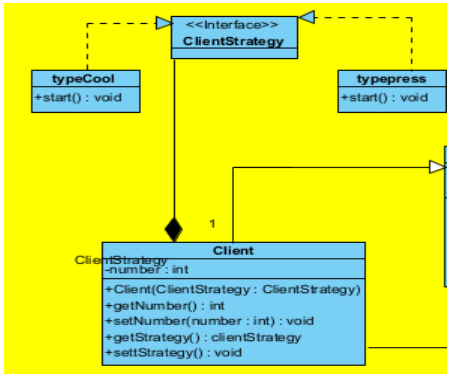
- OBSERVATOR

L'Observateur est un patron de conception comportemental qui permet de mettre en place un mécanisme de souscription pour envoyer des notifications à plusieurs objets, au sujet d'événements concernant les objets qu'ils observent.



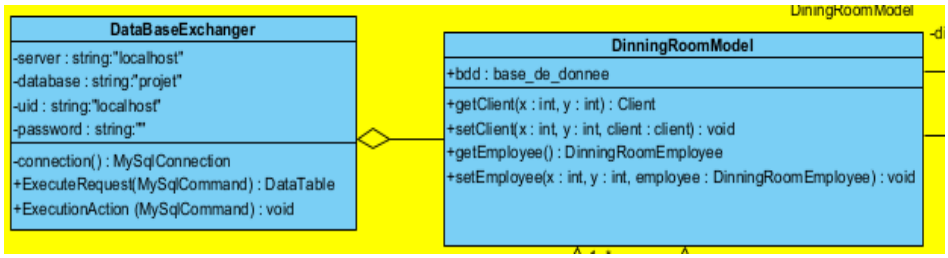
- STRATEGY

Le modèle de stratégie (également appelé modèle de politique) est un modèle de conception de logiciel comportemental qui permet de sélectionner un algorithme au moment de l'exécution. En règle générale, le modèle de stratégie stocke une référence à un code dans une structure de données et la récupère.



- DAO (DATA ACCESS OBJECT)

un objet d'accès aux données (DAO) est un modèle qui fournit une interface abstraite à un certain type de base de données ou à un autre mécanisme de persistance . En mappant les appels d'application à la couche de persistance, le DAO fournit des opérations de données sans exposer les détails de la base de données.



Comment fonctionne une architecture MVC ?

Un des plus célèbres design patterns s'appelle MVC, qui signifie Modèle - Vue - Contrôleur.

Le pattern MVC permet de bien organiser son code source. Il va nous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

• Modèle : cette partie gère les données du site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.

Modèle
(accès à la base de données)

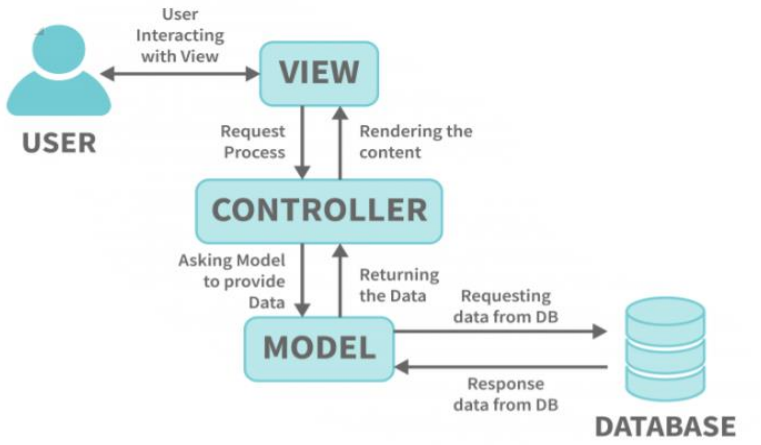
• Vue : cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.

Vue
(affichage de la page)

• Contrôleur : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

Contrôleur
(logique, calculs et décisions)

La figure suivante schématise le rôle de chacun de ces éléments.



Il faut retenir que le contrôleur est le chef d'orchestre : c'est lui qui reçoit la requête du visiteur et qui contacte d'autres fichiers (le modèle et la vue) pour échanger des informations avec eux.

Le fichier du contrôleur demande les données au modèle sans se soucier de la façon dont celui-ci va les récupérer.

Une fois les données récupérées, le contrôleur les transmet à la vue qui se chargera d'afficher la liste des messages.

Concrètement, le visiteur demandera la page au contrôleur et c'est la vue qui lui sera retournée, comme schématisé sur la figure suivante. Bien entendu, tout cela est transparent pour lui, il ne voit pas tout ce qui se passe sur le serveur.