PROJET LICENCE DEVOPS JEE

Application de gestion de portefeuilles clients

- Réalisé par : Sofia LAROUSSI
  Lucas RELAVE
  Siong TCHA
  Yannis RENAUD

SOMMAIRE

INTRODUCTION

Ce rapport décrit le projet effectué en groupe dans le cadre de la fin du module JEE, il s'agit d'une application de gestion de portefeuilles clients qui permettra aux utilisateurs de gérer les ventes, le stock, les références des produits et les clients.

Ce projet nous a permis de mettre en œuvre nos connaissances, se familiariser avec toute l'architecture JEE ainsi que de découvrir les frameworks les plus courant dans le développement web.

Vous trouverez dans ce rapport la présentation de toutes les technologies utilisées pour la réalisation de ce travail.

I. L'ENVIRONNEMENT DE DEVELOPPEMENT

1. Netbeans :

Netbeans est un environnement de développement intégré (IDE), il intègre la plupart des fonctionnalités exigées d'un IDE moderne dont, bien sûr, un éditeur de code gérant l'auto indentation et la colorisation.

Son parcours explique ainsi sa forte orientation pour le Java bien que le logiciel, qui continue à évoluer, permette également de gérer des développements en JavaScript, Python, XML, RubyC et C++ ainsi qu'en PHP et HTML5.

2. Glassfish :

Glassfish server est un serveur internet, qui vous permet de déployer des applications web écrites sur Java. Comme un autre serveur Web : Tomcat, Weblogic, Websphere, JBoss…

Glassfish est développé par Sun et donc il est plus fort que Tomcat. Après que la partie Java a été vendue à Oracle, Glassfish devient le produit d'Oracle. Actuellement, le serveur Glassfish a deux versions: une version gratuite avec code source ouvert et une version commerciale.

3. Bootstrap :

Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option.

II. METHODES UTILISEES :

1. Spring :

Le Framework Spring est un conteneur dit « léger », c'est-à-dire une infrastructure similaire à un serveur d'application J2EE. Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets (IoC – Inversion of Control). Le gros avantage par rapport aux serveurs d'application est qu'avec SPRING, les classes n'ont pas besoin d'implémenter une quelconque interface pour être prises en charge par le Framework. C'est en ce sens que SPRING est qualifié de conteneur « léger ». L'idée du pattern IoC est très simple, elle consiste, lorsqu'un objet A à besoin d'un objet B, à déléguer à un objet C la mise en relation De A avec B.

2. Pourquoi Spring ?

Spring permet principalement de faire la même chose que ce que l'on ferait avec un serveur JEE (Java dans sa version entreprise) complet comme par exemple WebSphere, Weblogic, JBoss, Glassfish, mais dans un serveur Web léger comme Tomcat ou jetty. Au niveau des fonctionnalités, il est en général en avance sur JEE. Il a en effet conçu au départ pour pallier certaines difficultés rencontrées lors de la mise en œuvre des premières versions de JEE et a ensuite progressivement creusé l'écart en évoluant plus rapidement.

Il permet de s'interfacer facilement aux différents Frameworks tires via des API unifiées car il offre des façades qui simplifient le code. Il se distingue également par sa capacité à permettre facilement de tester de façon performante les applications en embarquant dans les tests une version alléger du contexte d'exécution, alors que sur JEE il faut souvent un serveur dans son intégralité.

Le Framework Spring est une solution légère basée sur des briques logicielles indépendantes. Il apporte des solutions techniques unifiées pour la plupart des difficultés standards rencontrées sur les projets. Il offre des implémentations pour un ensemble de design patters. Les librairies d'extension apportent également des facilités pour l'utilisation des bibliothèques logicielles les plus utilisées. Il devient alors très facile de faire des web-services, d'accéder aux données, de faire une application basée sur des enchaînements d'écrans.

Spring est souple et permet l'utilisation progressive de ressources de plus en plus complexes en fonction de la montée en charger et de la nécessité de partager des ressources entre plusieurs applications. Il offre une alternative aux Frameworks historiques tels que Struts, en réutilisant leur force tout en apportant les évolutions

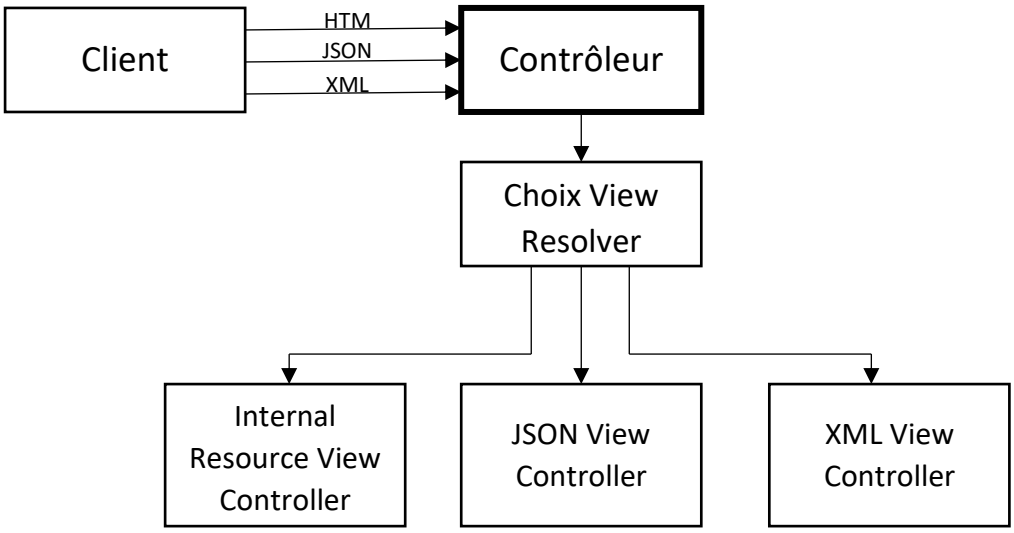
qui rendent la conception d'application MVC plus simple et plus robuste. Il permet également de contrôler le cycle de vie et offre la possibilité d'intercepter les appels des méthodes des objets managés afin d'en prendre le contrôle.

3. Modèle Vue Contrôleur (MVC)

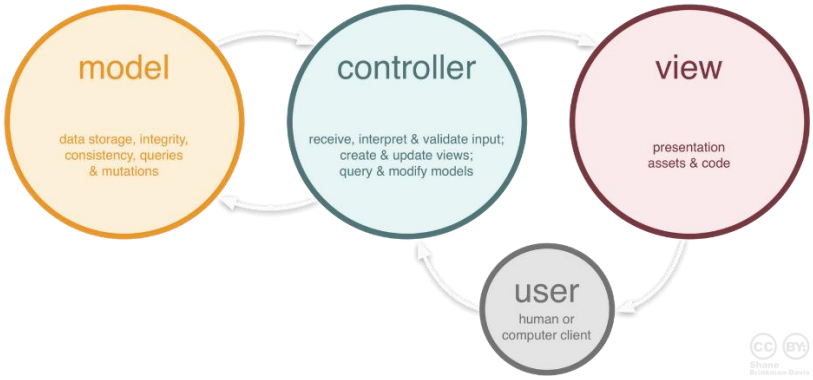
Spring MVC permet de faire simplement des applications web en séparant les trois éléments principaux :

Le modèle	Les données.
La vue	Ce qui est affiché.
Le contrôleur	Le traitement sur les données et l'enchaînement des vues.

Spring délègue le choix de la vue à un ViewResolver qui est, lui-même, un design pattern.



L'architecture MVC prend place dans la couche interface utilisateur lorsque celle-ci est une interface web.



- Modèle : C'est ici que se trouvent les données. Il s'agit en général d'un ou plusieurs objets Java. Ces objets s'apparentent à ce qu'on appelle souvent « la couche métier » de l'application et effectuent 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 !

- Vue : Ce que l'on nomme « la vue » est en fait une IHM. Elle représente ce que l'utilisateur a sous les yeux. La vue s'apparente donc à « la couche présentation » de l'architecture multicouche.

- Contrôleur : Cet objet - car il s'agit aussi d'un objet - permet de faire le lien entre la vue et le modèle lorsqu'une action utilisateur est intervenue sur la vue. C'est cet objet qui aura pour rôle de contrôler les données et les traitements sur ces données.

4. L'ORM HIBERNATE : La persistance des objets

a. Pourquoi Hibernate ?

Un ORM (Objet Relational Mapping) est un outil qui permet de passer d'un modèle logique objet vers un modèle physique relationnel. Il en existe plusieurs comme MyBatis mais le plus utilisé est Hibernate.

L'ORM Hibernate fait la correspondance entre les types Java et les types de colonnes des tables de la base de données. Il a aussi pour but de récupérer les contraintes sur les tables des objets du domaine et réciproquement. Il propose en plus des fonctionnalités supplémentaires comme la journalisation des données, la fourniture de statistiques sur l'usage qui est fait des données ou sir les performances. La mise en place d'un cache simple ou partagé permet également d'avoir des gains importants en performance si l'application est bien pensée.

Hibernate est un Framework open source gérant la persistance des objets en base de données relationnelle. La manipulation de SQL dans le langage de programmation JAVA est rendue possible par l'utilisation du JDBC. Puisque, chaque requête est effectuée sur le modèle logique de la base de données, cette approche présente l'inconvénient de lier très fortement le code de l'application au schéma de la base de données. En conséquence, toute évolution apportée au modèle logique doit être répercutée sur le code de l'application.
L'outil Hibernate propose une solution à ce problème. Celle-ci consiste à définir, dans des fichiers de configurations, le lien entre le diagramme de classes de l'application qui exploite une base de données et le modèle logique de cette base de données. Il permet ensuite de manipuler les données de la base de données sans faire la moindre

référence au schéma de la base de données en utilisant l'API fournie par cet outil grâce au lien établi dans les fichiers de configuration.

b. Configuration d'Hibernate :

Hibernate est conçu pour être utiliser dans différents types d'application. Pour cela, chaque application doit indiquer à Hibernate comment il peut accéder et manipuler la source de données dans un fichier de configuration nommé hibernate.cfg.xml. Les principaux éléments à paramétrer sont les suivantes :

- ✓ Le SGDB utilisé. Hibernate doit connaitre le type de SQL qu'il doit générer.

- ✓ La Connexion à la base de données. Si la connexion à la base de données se fait en utilisant JDBC, il faut indiquer à Hibernate, le driver JDBC, l'url de connexion ainsi qu'un nom d'utilisateur et un mot de passe permettant de se connecter à la base de données. Les connexions peuvent également être gérées par un serveur d'application. Dans ce cas, il faut indiquer à Hibernate comment il peut accéder aux connexions créées par ce serveur (Annuaire JNDI).

Pour résumer, le paramétrage de Hibernate nécessite :

- ✓ La définition du modèle de classes exploitant la base de données ;
- ✓ Une correspondance (mapping) entre le modèle de classes et la base de données ;
- ✓ Une configuration au niveau système de l'accès.

c. Utilisation d'Hibernate

L'utilisation de Hibernate se fait principalement au travers de la classe Session qu'il fournit. Un objet session offre les fonctionnalités suivantes :

- ✓ Rendre persistant un objet d'une classe. C'est la méthode *save* qui offre cette fonctionnalité. Elle prend en paramètre l'objet à rendre persistant.
- ✓ Charger un objet d'une classe à partir de la base de données. La méthode *load* est utilisée à cette fin. Elle prend en paramètre la classe de l'objet à charger ainsi que la valeur de l'identifiant (clé primaire) de cet objet.
- ✓ Modification d'un objet persistant. Il suffit pour cela de modifier la valeur des propriétés d'un objet puis d'appeler la méthode *flush* de l'objet session.
- ✓ Suppression d'un objet persistant. L'appel de la méthode *delete* avec en paramètre un objet persistant se charge d'effectuer la suppression dans la base de données.

- ✓ Rechercher des objets. Hibernate propose un langage de requête orienté objets nommé HQL dont la syntaxe est similaire au SQL et qui permet d'effectuer des requêtes sur le modèle objet.