Dossier de Conception Système

Document	**Dossier de Conception Système**
Version	**1.2**
Commencé le	**30 novembre 2006**
Dernière modification	**4 décembre 2006**
Statut	**Finale**
Client	**Enseignants du M2P GI**
Équipe	**DEBROUX Lionel** **FORNARA Romain** **KHLOUFI Samira** **TOKMEN Ali Savas**

Table des matières

1. But et portée du document

Le dossier de conception système est un document ayant pour but de permettre à toute personne de connaitre les principaux composants J2EE du site à réaliser, c'est-à-dire d'un site de vente de produits mangas en ligne.

Ce document est destiné:

- à l'équipe pédagogique

2. Glossaire

Voici l'explication d'une partie des termes techniques utilisés dans ce document:

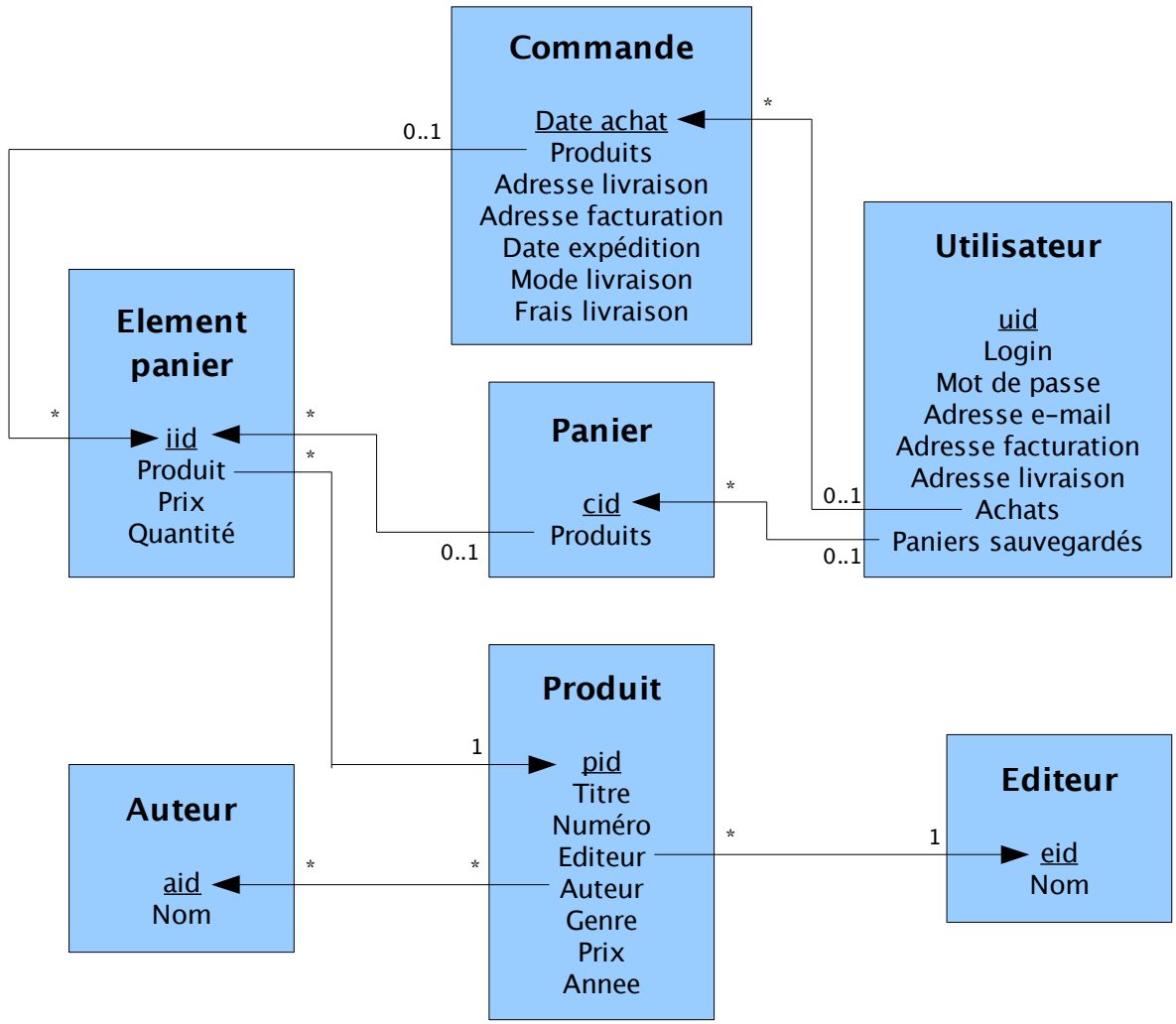
J2EE (ou Java EE)	Acronyme pour Java 2 Enterprise Edition, c'est la plateforme utilisée par le serveur du site
EJB	Acronyme pour Enterprise Java Bean, c'est un objet Java qui a des propriétés particulières du point de vue du serveur J2EE
Bean	Un composant logiciel écrit en Java
JNDI	Acronyme pour Java Naming and Directory Interface, c'est le service Java pour enregistrer et retrouver des objets partagés
servlet	Objet Java sur un serveur J2EE qui peut être accédé via une URL HTTP.
URL	Acronyme pour Universal Resource Locator, décrit une adresse de ressource sur le web.
HTTP	Acronyme pour HyperText Transfer Protocol, c'est le protocole standard utilisé pour accéder à des sites web
JSP	Acronyme pour Java Server Pages, c'est un mélange de HTTP et de Java pour pouvoir générer des pages dynamiques
DTO	Acronyme pour Data Transfer Object, c'est un objet intermédiare entre les beans et les JSP, pour que le JSP soit indépendant de la représentation interne du bean.
JSTL	Acronyme pour JSP Standard Tag Library, c'est une librairie pour pouvoir écrire des JSP contenant des tags similaires au tags HTML au lieu du code Java, ce qui rend le code plus lisible et indépendant de l'architecture sous-jascente.

3. Architecture système

Le système sera organisé de la façon suivante:

- Nous distinguons trois types de personnes: les clients, les utilisateurs et les administrateurs
  - Les clients peuvent lister des produits, les ajouter à leur panier et les acheter.
  - Les utilisateurs peuvent faire tout ce que peuvent faire les clients, mais aussi sauvegarder des paniers, voir leurs commandes ou encore changer leurs données personelles.
  - Les administrateurs ont accès à tout le système: ils peuvent ajouter, modifier et enlever les produits, les utilisateurs, etc.
- Un élément primaire du site est l'élément **Produit**. Cet élement a de divers attributs (nom, date de sortie, ...), un **Editeur** et au moins un **Auteur**. Le client doit être capable d'accéder à une liste filtrée de produits.
- Avant d'acheter un **Produit**, le client doit le mettre dans son **Panier**. Un **Panier** est composé d'un ensemble d'**Eléments de Panier**, et un **Elément de Panier** est un **Produit** avec une quantité, d'éventuelles réductions, etc.
- Il nous faut garder un historique entier des **Commandes**. Une **Commande** est composée d'un ensemble d'éléments achetés (des **Eléments de Panier**), d'une date d'achat, et d'autres éléments nous permettant de le suivre.
- Finalement, nous avons des **Utilisateurs**: les **Utilisateurs** ont un login et un mot de passe, et ont le luxe de pouvoir sauvegarder des paniers et de voir leur historique de **Commandes**.

  De façon schématique:



4. Architecture des EJB

Les EJB reprennent l'architecture présentée en haut, à savoir:

- Auteur, Editeur, Produit, Utilisateur, Commande, Panier et ElementPanier seront des beans persistants, donc qui vont être sauvegardés de façon permanente. Leurs élements seront des méthodes publiques
- J2EE propose aussi des beans de session, qui ne sont pas permanents. En effet, ces beans seront détruits une fois que le client s'est déconnecté. Un Panier (avec son ElementPanier correspondant) seront aussi implémentés en tant que bean session (donc nous aurons deux Paniers: un PanierPersistant et un PanierSession.

Il faut noter que ces beans sont internes au serveur J2EE, donc ils ne sont pas interrogeables directement par une application extérieure. Nous créerons deux beans accessibles de façon externe:

- Un bean client (**EcomCustomerBean**), qui peut lister des produits, ajouter un produit à son panier, visualiser le contenu de son panier ou encore acheter son panier. Il peut aussi s'identifier au système (nom d'utilisateur + mot de passe): dans ce cas, ce bean contiendra une référence vers une entité utilisateur. Cet entité peut sauvegarder des paniers et consulter les paniers sauvegardés ainsi que les commandes précédentes (comme détaillé précédemment dans le schéma).
- Le bean administrateur (**EcomAdminBean**), qui peut gérer l'ensemble des beans persistants (ajouter, modifier et effacer des Auteurs, Editeurs, Produits et Utilisateurs). Il a aussi accès à l'historique des Commandes.

5. Architecture des clients

Les trois beans accessibles de l'extérieur possèdent donc des interfaces Remote. Pour accéder à cette interface, il suffit d'utiliser la méthode **lookup** du contexte JNDI.

5.1 Clients lourds

Nous avons deux clients lourds: un client normal et un administrateur.
Quand un client lourd est executé:

- Il obtient un bean client ou administrateur (via JNDI)
- Il affiche la liste des options et attend pour une entrée (*)
- Si l'entrée est une action (lister les produits, acheter, ...) il appelle la méthode correspondante sur le bean, affiche le résultat et va à l'étape (*)
- Si l'entrée est la commande quitter, termine le client lourd. Le serveur J2EE va automatiquement détruire le bean session associé.

5.2 Clients légers

Nous avons un client léger (un servlet). Ce client est appelé par le composant web (Tomcat) du serveur J2EE quand une requête web est reçue.
Quand il est appelé:

- Il obtient l'objet session correspondant à l'appelant ou en crée un s'il n'y en avait aucun.
- Il vérifie qu'un bean client valide est bien présent dans cet objet de session. Si ceci n'est pas le cas, le bean est créé par appel au JNDI.
- Il analyse et traite la requête HTTP (un switch similaire à celui du client lourd).
- Une fois la requête traitée (ce qui met à jour automatiquement les beans correspondants), il appelle le composant de rendu (**main.jsp**).
- **main.jsp** est le seul JSP qui contient du code Java: il met les variables de page (**pageContext.setAttribute**), crée les grandes lignes du site et dans chaque partie, inclut le JSP qui doit le rendre (**head.jsp**, **products.jsp**, **cart.jsp**, etc.)
- Les JSP inclus par **main.jsp** n'ont aucun code Java, ils sont tous en JSTL. Ceci nous permet d'avoir:
    - Une transparence parfaite par rapport aux beans sous-jascents sans avoir à réécrire des objets spéciaux (DTO)
    - Le support standard d'internationalisation (i18n): JSTL permet de gérer simplement plusieurs langues en mettant un fichier de propriétés par langue.