Pet Store

Un magasin de vente en ligne générique basé sur:
Le Kit de Développement Logiciel de Java 2, Edition Entreprise (J2EE)

Pasquier Barthelémy Ticula Omont

lundi 9 mars 2009

1) Objectif de l'étude

L'objectif de ce projet est de comprendre

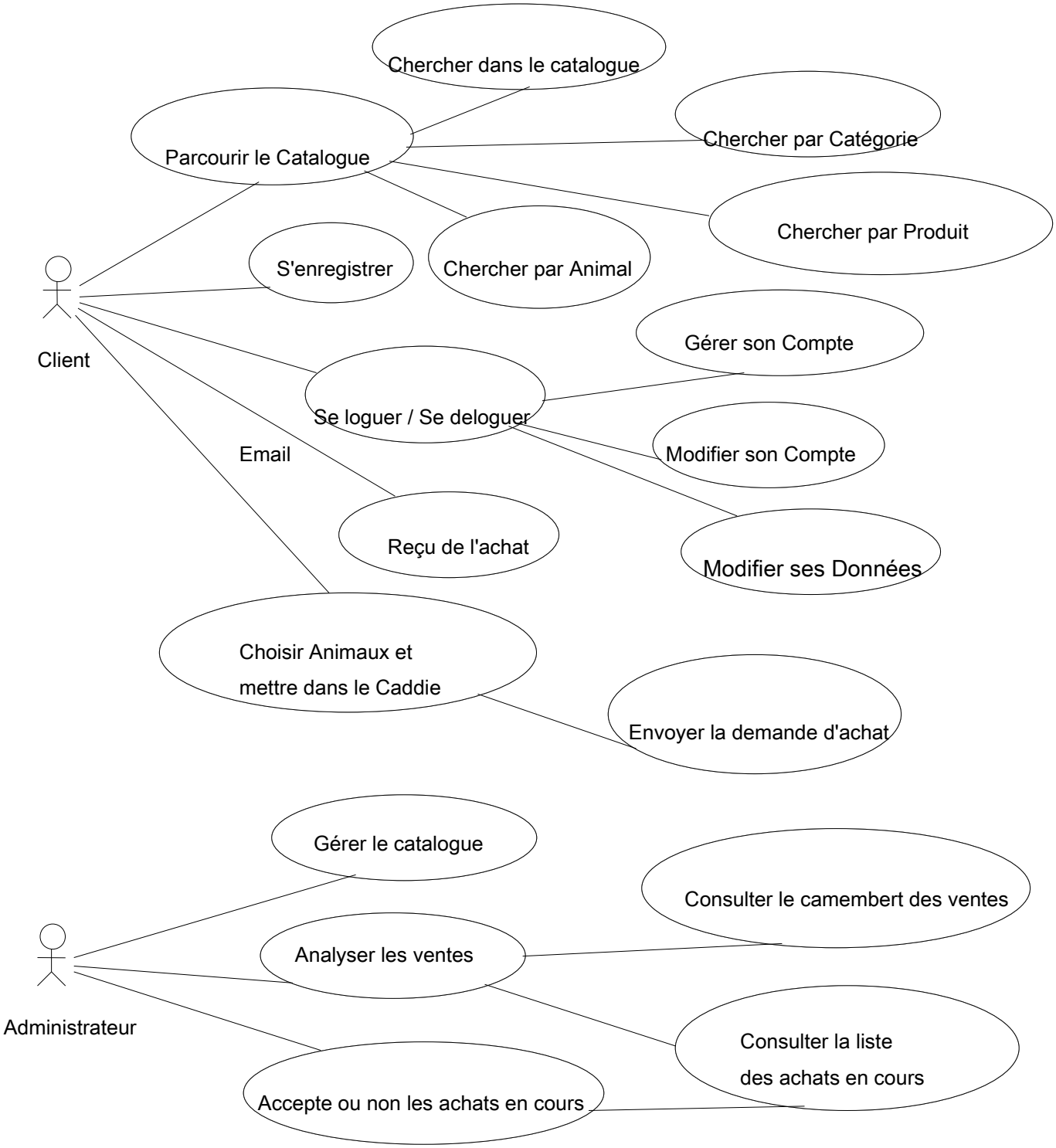
- le fonctionnement du J2EE
- les règles de « bonne » analyse et implémentation d'une application

à travers l'application type « PetStore » conçue comme un exemple par les architectes logiciel de SUN, puis de vérifier que nous avons bien compris en implémentant des extensions de cette application.

Les moyens utilisés sont développés et organisés à travers les thèmes classiques de l'analyse, de la conception et de l'implémentation comme l'indique la table de matières :

2) Documentation d'analyse

a) Diagramme d'utilisation de « Petstore » et de sa partie d'administration



On voit sur ce diagramme d'utilisation les deux types d'acteurs : L'administrateur et le client, auxquels sont associés des fonctions différentes

Ce diagramme correspond au périmètre d'utilisation de l'application tel qu'il a été défini au départ et tel qu'il est réellement dans la version finale de PetStore. Dans la version que nous avons étudiée (1.1.2), la seule fonction administrateur implémentée est « Accepte ou non les achats en cours » et bien sur, la fonction « Consulter la liste des achats en cours » qui lui est liée.

Par la suite, nous implémenterons la fonction « Consulter le camembert des ventes » et nous développerons la fonction « Consulter la liste des achats en cours » grâce à F.O.P. (Formatted Objects Processor) qui nous permettra d'implémenter la fonction « Analyser les ventes » par la génération de documents PDF à la volée.

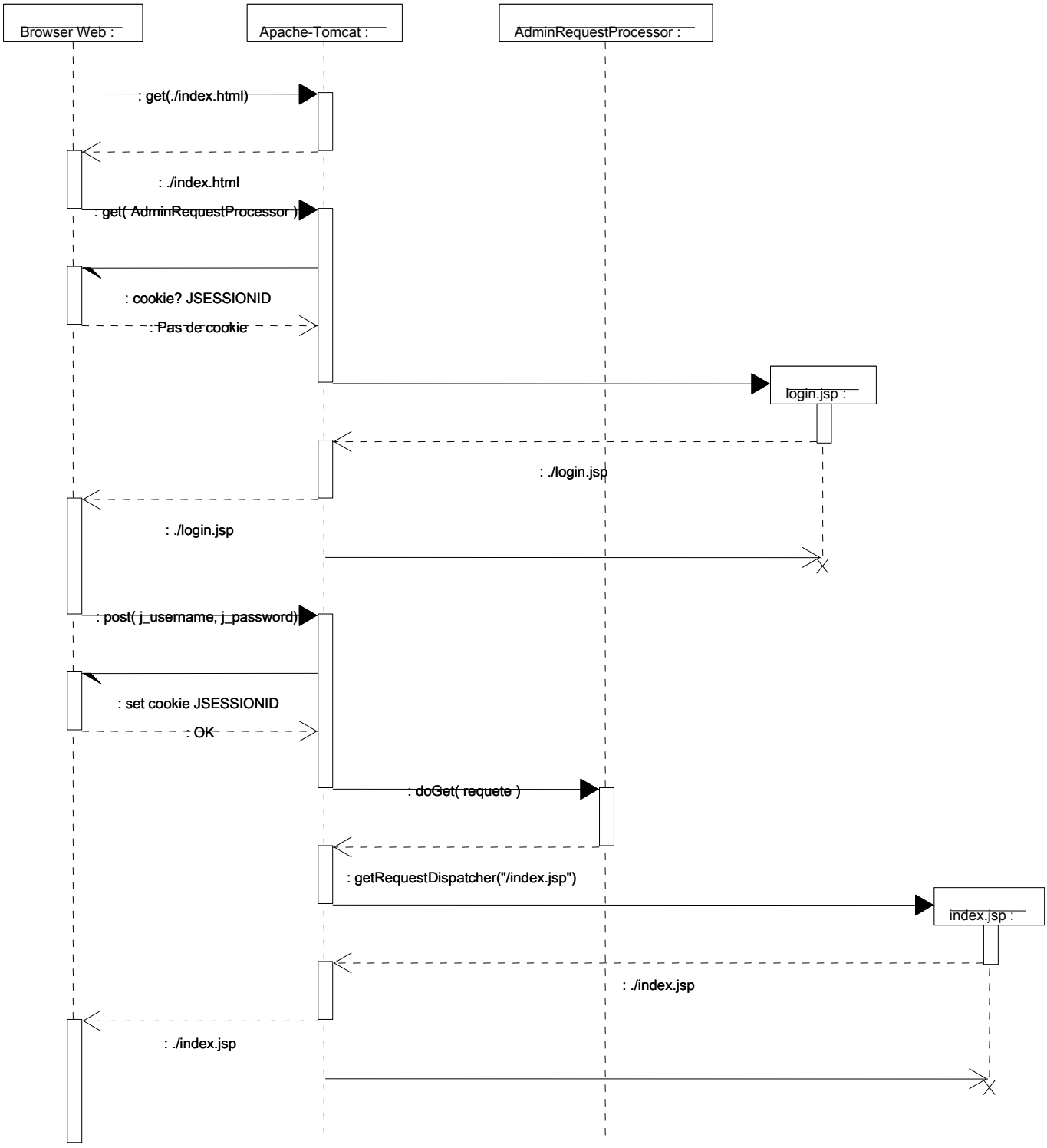
b) Modélisation en Z de fonctions « client »

(Comme fuzz, l'outil de test de la formalisation en Z fonctionne sur des fichiers latex, cette partie du rapport est réalisée avec le paquetage latex correspondant et s'insère mal dans le reste du rapport…)

3) Documentation de conception

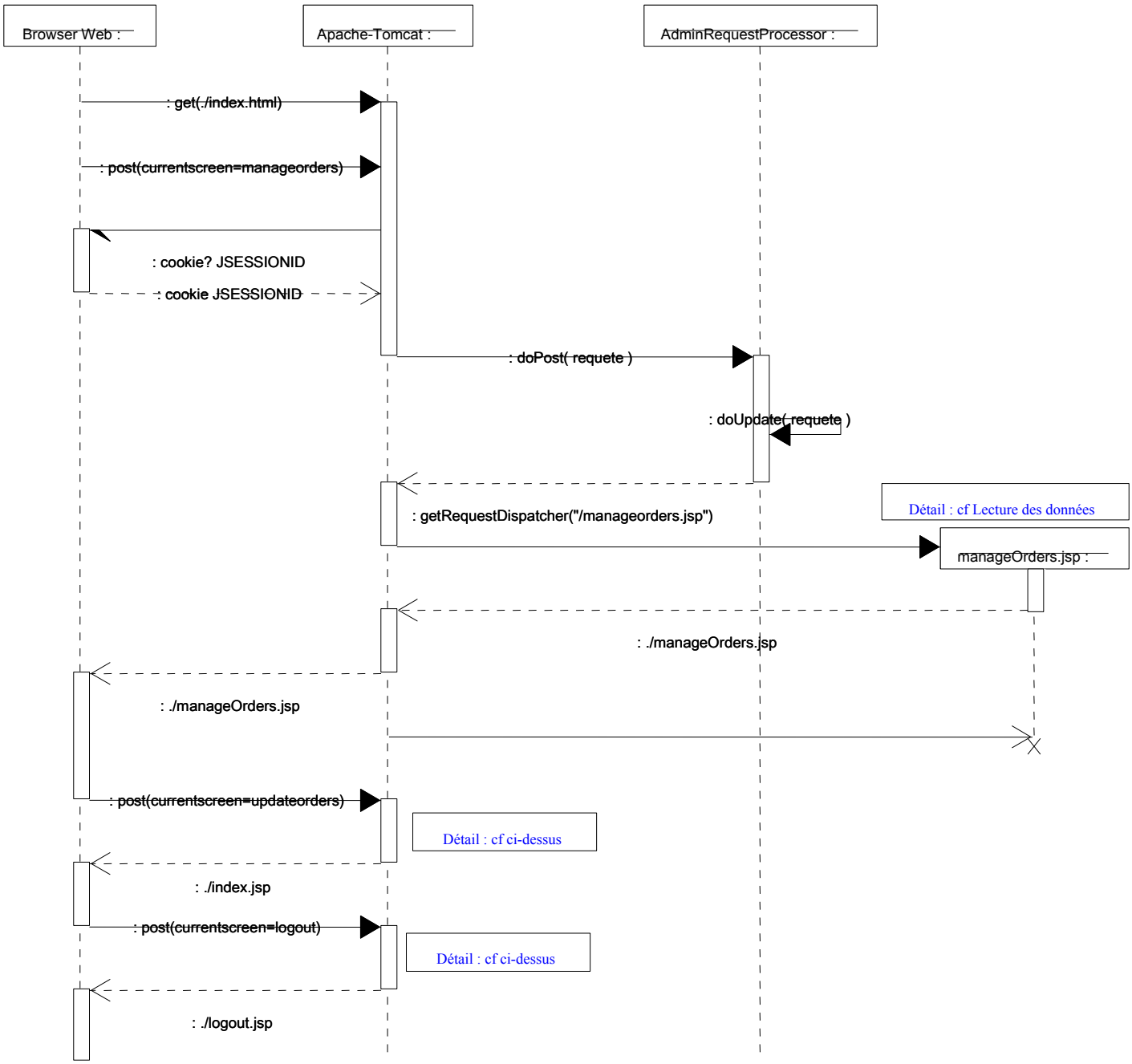
a) Diagrammes de séquences

i) Utilisation normale



(Suite page suivante)

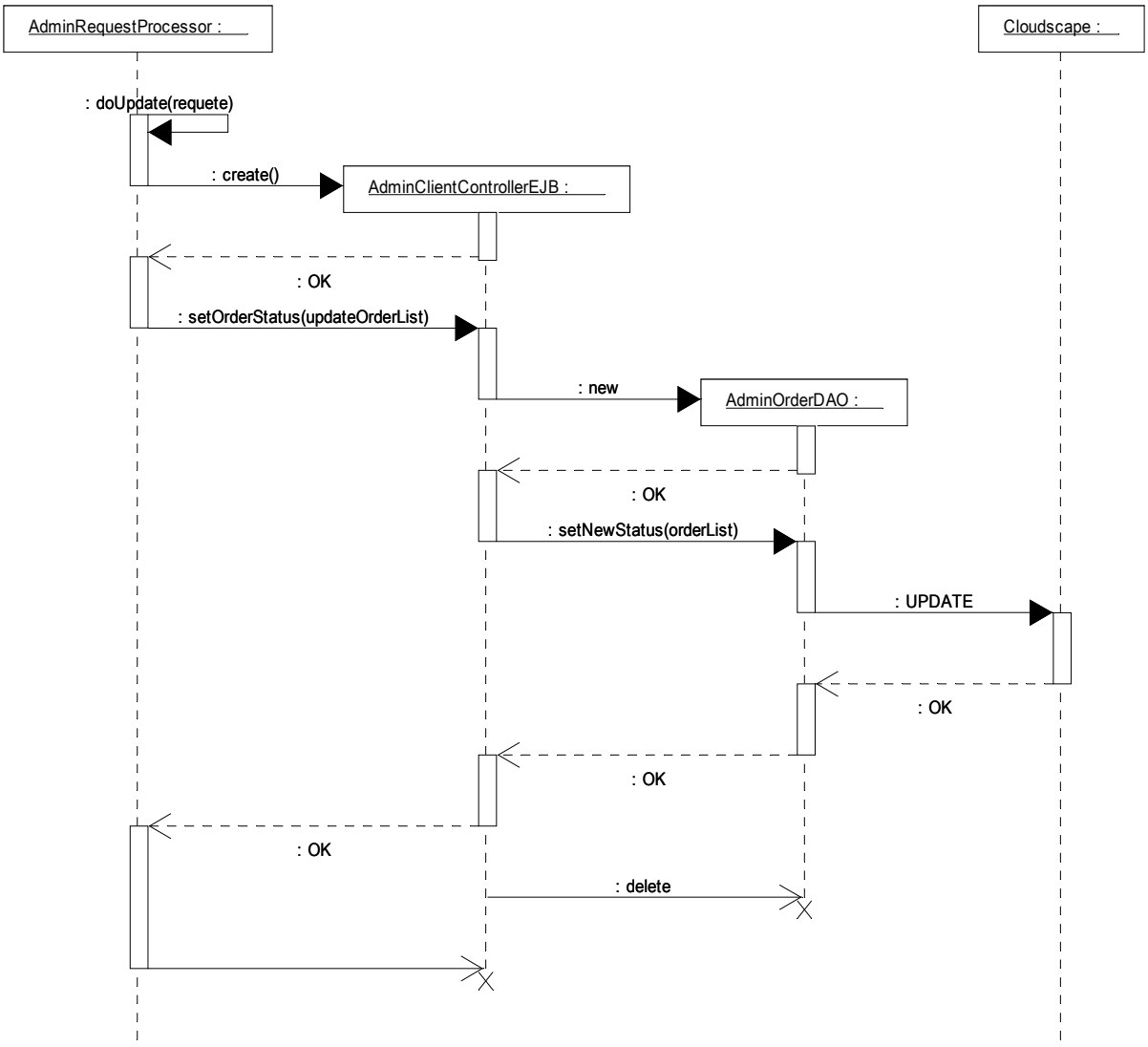
(Suite)



On voit que le système repose sur l'utilisation d'une servlet qui redirige les requêtes vers des pages JSP en fonction du paramètre caché « currentscreen » qui les accompagne. La servlet s'occupe aussi de lancer la mise à jour des données lorsque c'est nécessaire.

On remarque aussi que le processus d'authentification est intégré au J2EE qui établit lui-même un cookie sur le brouteur de l'administrateur.

ii) Mise à jour des données

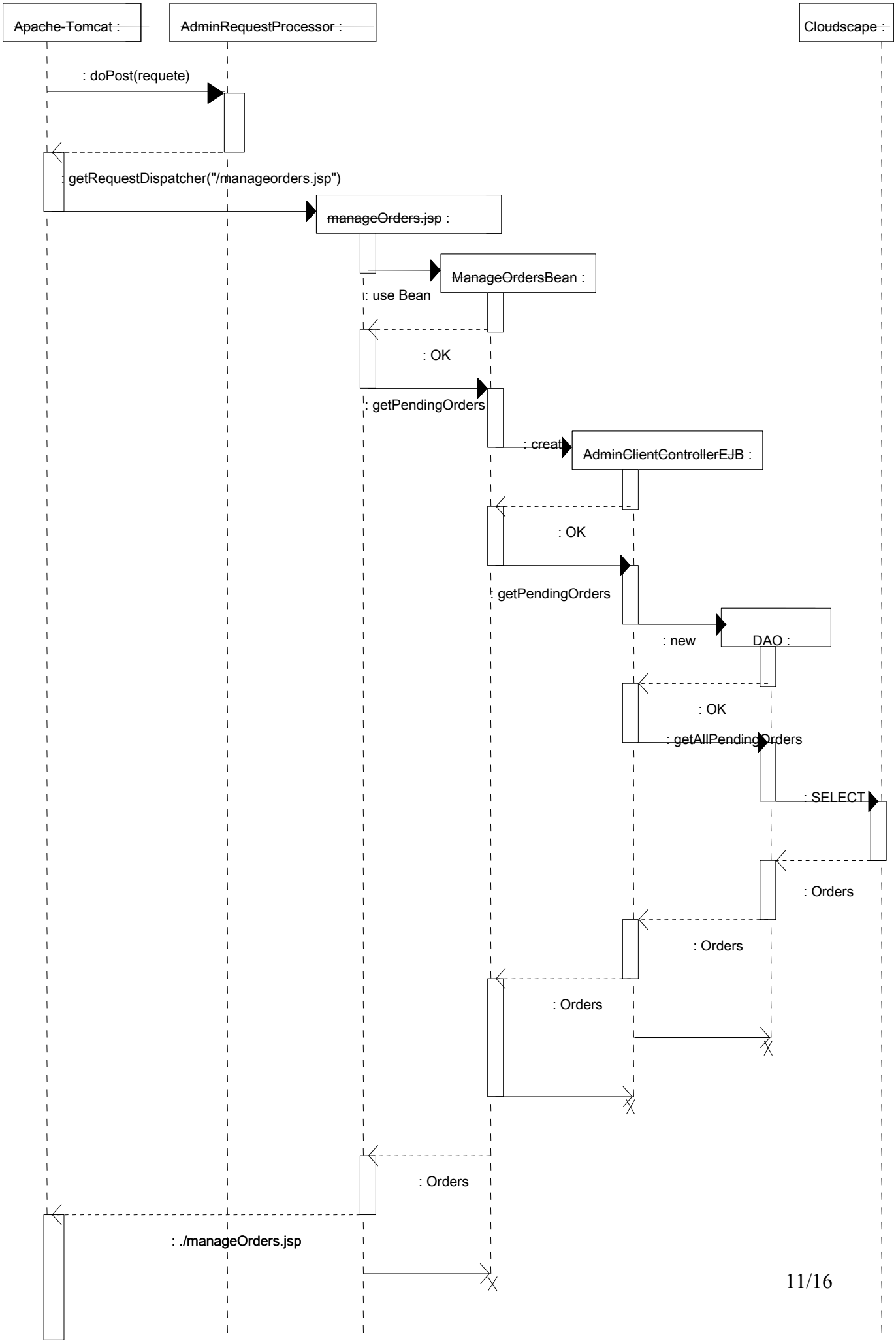


On constate que c'est la servlet qui lance la mise à jour des données en appelant un EJB qui, lui-même appelle un DAO (Data Object Accessor), qui n'est en fait qu'une interface qui possède une implémentation par base de donnée sous-jacente, ici Cloudscape. Ainsi, par cette sorte de façade, on rend l'application modulaire en facilitant le changement de base de données.

iii) Lecture des données

(Le diagramme de séquence est à la page suivante pour un problème de taille).
On constate que, dans ce cas, c'est une page java de serveur (JSP) qui appelle un bean, qui lui-même appelle un EJB, qui, comme précédemment appelle un DAO afin d'afficher les ordres qui demandent confirmation sur l'écran du brouteur de l'administrateur.

Cette organisation avec un bean permet d'imaginer simplement d'autres formats de présentation des données. La génération de documents PDF avec FOP utilise cette spécificité du système.


Apache-Tomcat :
AdminRequestProcessor :
Cloudscape :
: doPost(requete)
: getRequestDispatcher("/manageorders.jsp")
manageOrders.jsp :
ManageOrdersBean :
: use Bean
: OK
: getPendingOrders
: creat
AdminClientControllerEJB :
: OK
: getPendingOrders
: new
DAO :
: OK
: getAllPendingOrders
: SELECT
: Orders
: Orders
: Orders
: Orders
: ./manageOrders.jsp

b) Analyse de design patterns utilisés dans Petstore

Nous avons identifié plusieurs design patterns utilisé par SUN pour construire PetStore.

i) Front Controller

Il y a un seul composant qui est responsable pour la gestion de toutes les requêtes du client. C'est le rôle du contrôleur de sélectionner la prochaine page à afficher en fonction des résultats de la demande du client. Dans notre cas le « front controller » est la servlet MainServlet.java.

Diagramme de séquence

Un diagramme de séquence simplifié est le suivant. On a omis de représenter les traitements métiers réalisés par la couche EJB. Ils sont effectués par l'objet Model qui est une vue macroscopique de la logique métier.

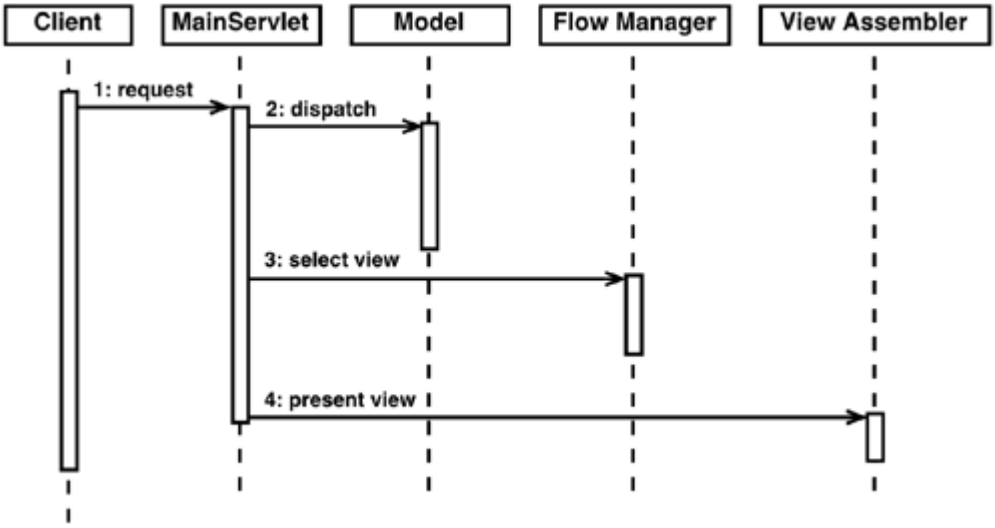


Figure 1- Diagramme de séquence simplifié Front Controller

L'objet Flow Manager gère l'enchaînement des pages JSP. En fonction des résultats des opérations qui sont effectuées le Flow Manager décide vers quelle page renvoyer le client. Par exemple si il y eu une erreur lors de l'identification le client sera redirigé vers une page d'erreur. Tout ces choix étant centralisés les pages JSP ne sont pas aussi fortement liées qu'elles le seraient dans une architecture qui n'utiliserait ce design pattern.

ii) Data Access Object

L'utilisation de ce design pattern permet l'isolation des opérations spécifiques à un système particulier de stockage des données qu'il soit un SGBD relationnel, des fichiers plats, des fichiers XML.
Ce design pattern utilise le design pattern Factory. Les opérations qu'un objet doit supporter sont spécifiées à l'aide d'une interface. Les objets qui implémentent cette interface sont particularisés pour supporter les spécificités d'un système particulier de stockage. L'objet Factory est chargé d'instancier la bonne implémentation de

l'interface. Ce choix est fait dans PetStore en modifiant un paramètre dans les fichiers de configuration de l'application.

Diagramme de classes

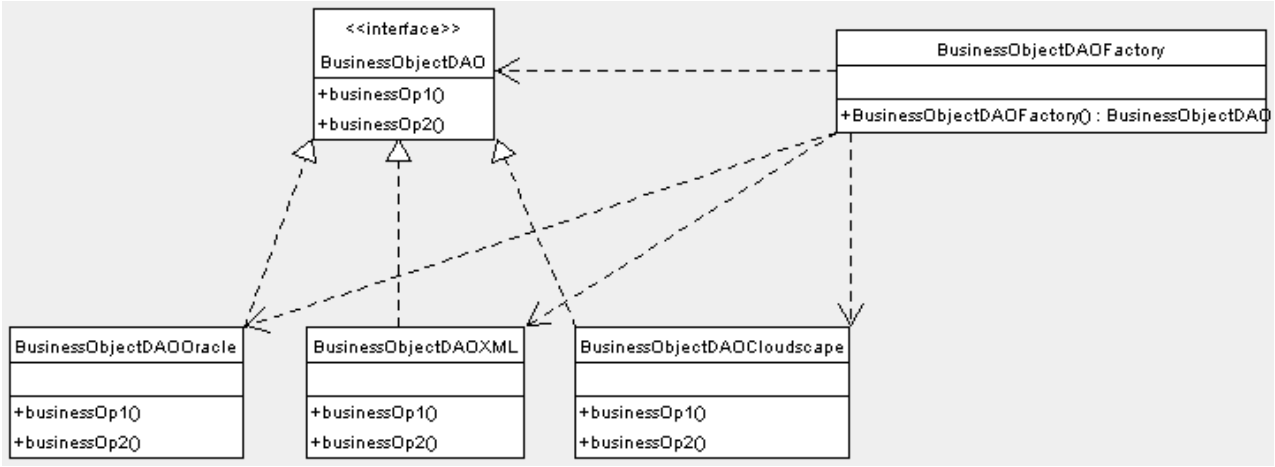


Figure 2- Le diagramme de classes pour le desgin pattern DAO

Diagramme de séquence

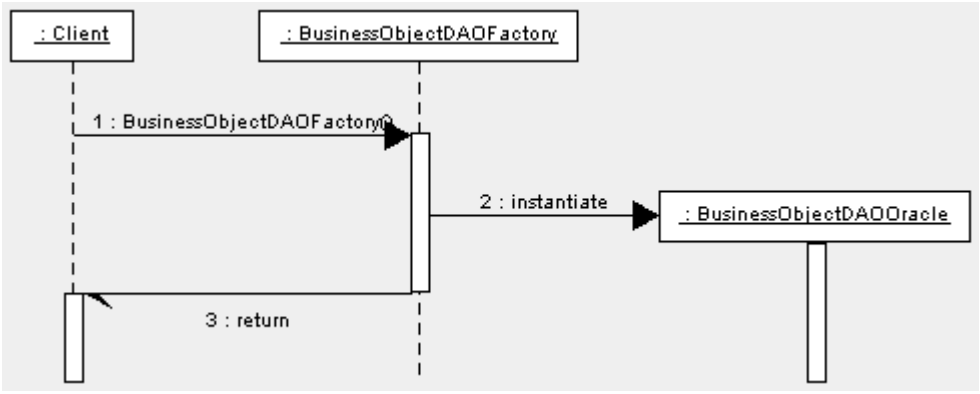


Figure 3- Le diagramme de séquence pour le design pattern Data Access Object

iii) Fast Lane Reader

L'extension, que nous avons développé, pour l'application d'administration de PetStore utilise ce design pattern. Quand on veut accéder à une base de données (ou à un autre système de stockage) en lecture seule et que l'on accepte que les données puissent être « périmées » on peut court-circuiter la couche EJB et accéder directement à la base de données. On fait cela pour des soucis de performance. Les services transactionnels offerts par les conteneurs EJB ne servent pas pour ce type d'accès.

Diagramme de séquence

Le diagramme de séquence suivant montre l'utilisation de ce design pattern pour l'application d'administration de PetStore

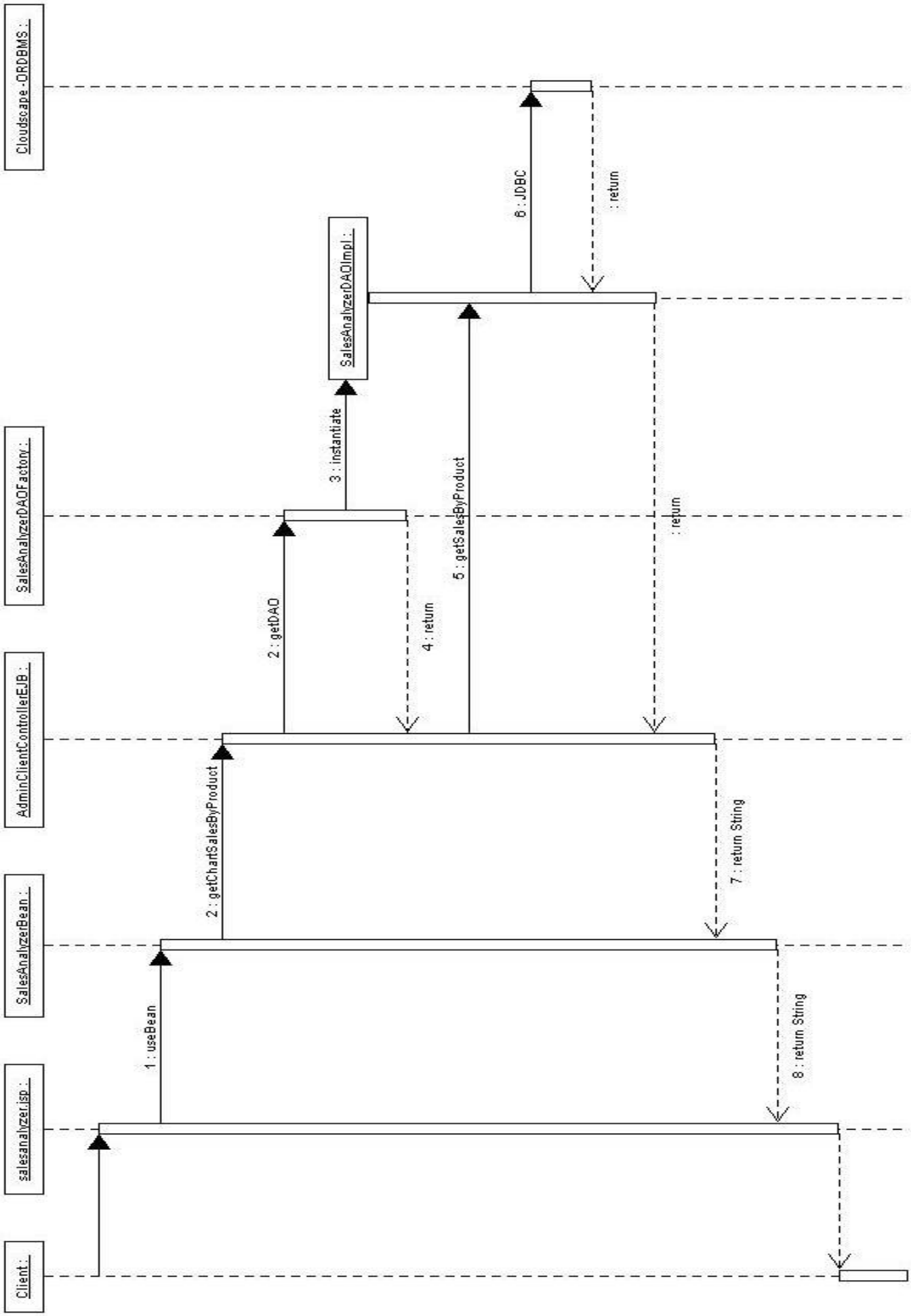


Figure 4- Diagramme de séquence Fast Lane Reader

iv) Session Facade

Pour accomplir des différents processus métiers le client est obligé d'appeler plusieurs EJB. En procédant de cette manière on obtient une dépendance forte entre le client et la couche EJB. En plus le client doit gérer toutes les erreurs EJB ce qui alourdit le code. Pour réaliser un couplage faible entre le client et la couche EJB on centralise toutes les opérations business dans un seul bean SessionBean. C'est à la charge du SessionBean de connaître la complexité de la couche EJB. Le client appelle ce bean pour toute opération qu'il veut exécuter. Dans la figure 4 –ci-dessus c'est l'AdminControllerEJB qui joue le rôle que l'on vient de décrire.

4) Documentation technique de réalisation

a) Implémentation du camembert des ventes

L'implémentation du camembert des ventes est réalisée selon le même schéma que la lecture des données décortiquée ci-dessus. Une page java de serveur utilise un bean écrit pour l'occasion pour accéder à une instance du seul EJB de l'application, qui accède à une version modifiée du DAO précédent afin de récupérer toutes les données de vente et non pas seulement les commandes en attente. L'EJB génère une image JPEG que la JSP récupère afin de l'afficher.

b) Implémentation de la génération des .PDF avec fop

La génération de PDF avec FOP nécessite la modification de la servlet principale. En effet, les JSP ne sont indiquées pour générer autre chose que des pages HTML. La servlet se charge donc elle-même de générer un flux au format FO (Formatted Object, basé sur XML) de la même manière que la JSP manageorders.jsp génère une page web : en utilisant ce bean. Pour montrer les capacités du langage, on insère aussi le camembert précédemment construit dans le flux.
Ensuite, on donne ce flux à traiter à FOP qui génère un flux PDF que la servlet envoie au brouteur de l'administrateur.

c) Déploiement de l'application

Le déploiement de l'application est relativement lourd :

- Il faut installer J2EE v1.2.1 et pas un autre.
- Il faut ensuite déployer les applications Petstore et Admin à l'aide de l'outil de déploiement livré avec le J2EE (deploytool.bat). Cela met longtemps car l'application Admin originelle faisait 500 ko et l'ajout de FOP et des librairies qui lui sont nécessaires la porte à 5 Mo.

- Puis il faut créer le sous-repertoire public_html/ChartImages
- Il faut donner au serveur des droits d'écriture dessus, ce qui se fait soit en écrasant le fichier server.policy avec celui fourni, soit en le modifiant à l'aide de l'utilitaire policytool.exe fourni dans le JDK de la manière suivante :
    - Ouvrir le fichier $J2EE/lib/security/server.policy
    - Créer une « policy entry » :

      Nom: file:${com.sun.enterprise.home}/public_html/admin/ChartImages/-
      FilePermission: java.io.FilePermission
      TargetName: <<ALL FILES>>
      Actions: read,write

- On peut alors lancer les serveurs par les commandes :
    - cloudscape –start
    - j2ee.bat –verbose
- Il faut ensuite entrer dans le site Petstore (http://localhost:8000) pour créer les tables de la base de données.
- Il faut ensuit créer un utilisateur administrateur à l'aide l'outil createadminuser.bat fourni avec petstore admin.

Si vous êtes arrivés au bout, vous bénéficiez d'un environnement à la fois flexible et capable de monter en puissance, ce qui vous consolera de vos difficultés. Prévoyez aussi une machine assez puissante, l'ensemble J2EE + FOP 0.20 semble assez lourd.

5) Références bibliographiques

- Java TM Pet Store 1.1.2 Release
    - http://developer.java.sun.com/developer/releases/petstore/petstore1_1_2.html

- Core J2EE Design Patterns
    - http://java.sun.com/blueprints/patterns/index.html

- Designing Enterprise Application with J2EE par Nicholas Kassen

- Designing Enterprise Applications with the J2EE Platform par Inderjeet Singh, Beth Stearns, Mark Johnson, Enterprise Team
    - http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/

- Site du projet FOP (Formatting Objects Processor)
    - http://xml.apache.org/fop/