JavaAirlines

1.0

Karim Lamouri & Chandara Alexy Im (Groupe 214)

Projet réalisé dans le cadre de l'enseignement : Conception de Programmes à Objet Avancé
IUT Paris Descartes — Département Informatique — Semestre 4 décalé
Correcteur : M. Jean-François Brette

Table des matières

Chapitre 1

Synthèse du projet ”JavaAirlines”

1.1 Présentation de l’application

1.1.1 Introduction

"JavaAirlines" est une application développée dans le cadre de l’enseignement de Conception de Programmes à Objet Avancé. Cette application est développée en Java EE. Elle simule le site de gestion d’une compagnie aérienne. Par exemple les passager peuvent réserver un vol, les pilotes demander à ne pas effectuer un vol. L’administrateur peut créer un départ, le supprimer et modérer les demandes de désaffectation des pilotes. Cette application est une application 3-Tiers le premier Tiers est l’ensemble des pages affiché sur un navigateur avec lesquels interagi un utilisateur, le deuxième Tier est le coeur de l’application il gère les demandes des utilisateurs et la communication avec la base de données qui est le Troisième Tier de cette application.

1.1.2 Structuration des Composants

Dans l’ensemble notre application fonctionne sur un patron de conception Modèle Vue Contrôleur (grâce aux JSP et à la bibliothèque JSTL). Elle implémente également d’autres patrons et des techniques évoluées de programmation. Ainsi en plus du patron Modèle Vue Contrôleur, notre application est implémenté avec :

– Un gestionnaire d’objet persistant (Patron de conception Data Acces Object) permettant de dissocier les couches sémantique du code et celle d’accès aux données. Cela permet, entre autres, de changer plus rapidement le mode de stockage des données persistante (XML ou SGBD).
– Le patron ci-dessus est couplé à un patron de conception de type singleton
– Ces derniers patrons sont également complété par un patron de conception de type Fabrique pour abstraire correctement la couche données de l’application
– Les connexions à la base de données Oracle s’effectue par un bassin de connexion, géré par le serveur Tomcat, ainsi, un fois ce bassin initialisé, obtenir une connexion est toujours très rapide. Et la programmation en deviens plus transparente et claire.
– Une abstraction total du code Java dans les JSP grâce à la bibliothèque JSTL
– Les types date tournis par Java étant un peu désuètes notre application s’appuie sur la bibliothèque JodaTime pour la gestion des dates. Celle-ci ayant de bonnes chance de devenir un

standard de l'API Java.
– Une portabilité parfaite est assuré par notre application, en effet nos composant sont relié entre eux statistiquement, mais dynamiquement à leur racine, en effet s'il l'ont modifie le nom racine de notre application (ici JavaAirlinesIMLAMOURI214) celle-ci continuera de fonctionner. Car les chemins absolus sont assuré dynamiquement
– Ainsi notre patron de conception et nos beans nous permettent de n'avoir qu'une seule variable de session elle correspond à l'utilisateur connecté via son bean utilisateur

1.2 Tâches accomplies

Dans ce projet nous avons accompli plusieurs tâches comme :
– Utilisation des JSP et de la Bibliothèque JSTL (Abstraction du code Java dans les Vues)
– Départ d'une base de données non vierge, impliquant l'ajout de nouvelles contraintes sans modifier l'existant
– Exploitation fine des paramètres de Tomcat comme la mise en place d'un filtre de redirection ou encore un filtre réglant les problème d'encodage avec UTF-8

1.3 Difficultés rencontrées

Dans ce projet quelques difficultés on été rencontré et surmonté comme :
– La compréhension du patron de conception DAO
– La création du bassin de connexion
– L'apprentissage de la syntaxe tournis par la bibliothèque JSTL
– Les propriété à affecter aux Beans de l'applications

1.4 Bilan

Le projet est totalement fonctionnel et, espérons-le, optimisé. Son aboutissement nous a conduit à manipuler le langage Java EE en utilisant ses spécificités sémantiques et syntaxiques. A notre connaissance, notre projet semble dépourvu de bugs, nous pouvons donc dresser un bilan positif de notre projet, avec un sentiment de satisfaction. Il faut noter que nous avons privilégié la qualité à la quantité, ainsi par manque de temps nous n'avons pas pu faire l'envoie des e-mails, mais toute les autres fonctions sont présentes et notre application a une base solide et pourrait évoluer très facilement grâce à tout les patrons que nous avons appliqué.

Chapitre 2

Fonctionnement de l'application

2.1 Configuration

Notre application doit simplement être déposé dans le dossier webapps de Tomcat pour fonctionner Note : Les fichiers sources sont encodés en UTF-8, il faut donc changer l'encodage du projet dans Eclipse pour profiter pleinement du projet

2.2 Fonctionnement général

Pour une meilleure appréhension du projet voici, les fonctions réalisé par notre application :

- Pour un Administrateur
  - Modération de la demande de désaffectation de départ d'un pilote (Servlet Planning)
  - Ajouter un départ (Servlet AjoutDepart)
  - Supprimer un départ et son vol associé (Servlet SupprimerDepart)
  - Déconnexion (Servlet Deconnexion)
- Pour un Pilote
  - Demande de désaffectation d'un départ et vue du planning des départ (Servlet Planning)
  - Changer de mot de passe (Servlet MotDePasse)
  - Déconnexion (Servlet Deconnexion)
- Pour un Passager
  - Planning des réservations (Servlet Planning)
  - Reserver un vol (Servlet Reservation)
  - Changer de mot de passe (Servlet MotDePasse)
  - Déconnexion (Servlet Deconnexion)

Chapitre 3

Code source