Rapport de laboratoire

**Département de génie logiciel et des technologies de l’information**

|  |  |
| --- | --- |
| **No de laboratoire** | 2 |
| **Étudiant(s)** | François Gagné  Patrick Lavallée  Julien Pelletier-Morin  Simon Turcotte |
| **Code(s) permanent(s)** | GAGF20067801  LAVP12048408  PELJ03098909  TURS17118905 |
| **Cours** | GTI525 |
| **Session** | Hiver 2013 |
| **Groupe** | 01 |
| **Professeur(e)** | Éric Boivin |
| **Chargé(e) de laboratoire** | Mohamed Outellou |
| **Date** | CHANGE LA DATE! |

Table des matières

[1 Introduction 2](#_Toc349430644)

[2 Diagramme de classe 4](#_Toc349430645)

[3 Discussion 5](#_Toc349430646)

[3.1 Quel est la stratégie que vous avez utilisée pour associer un URL au contrôleur associé? 5](#_Toc349430647)

[Quels patrons de conception Web ont été utilisés dans votre application? Si vous avez utilisé des frameworks, quels patrons sont implémentés par celui-ci? 5](#_Toc349430648)

[Expliquez à quel endroit vous avez utilisé AJAX dans votre application. 5](#_Toc349430649)

[Avez-vous utilisé une technologie où un framework additionnel? Si oui, expliquez quel est le rôle de cette technologie dans votre application. 5](#_Toc349430650)

[Suite à l'expérience du laboratoire 2, quelles sont les éléments du diagramme de classe du laboratoire 1 que vous auriez planifié différemment? 5](#_Toc349430651)

[Quel a été la principale difficulté technique rencontrée dans le laboratoire? 5](#_Toc349430652)

[Comment a été faite la séparation des tâches dans votre équipe? 5](#_Toc349430653)

[De quelle manière avez-vous géré la gestion de la session? 5](#_Toc349430654)

[4 Conclusion 6](#_Toc349430655)

Table des figures

[Figure 1 - Patron Transfer Object 4](#_Toc349781047)

[Figure 2- Architecture des tests unitaires 5](#_Toc349781048)

# 

# Introduction

Dans le cadre de ce cours, l’équipe doit concevoir et déployer un système de vente de billet en ligne. Il s’agit de la deuxième itération du projet qui vise à dynamiser l’expérience utilisateur notamment en ajoutant une couche applicative qui répondra à des requêtes HTTP.

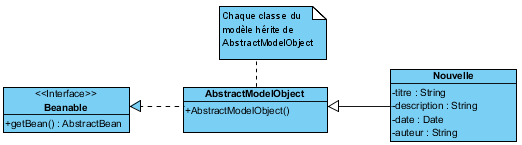
Lors de la première itération, un prototype statique a été élaboré. Ce prototype est dynamisé en transformant les pages statique au format JSP et en incluant des mécanismes d’obtention de données ce trouvant sur un serveur distant. Ces données sont pour l’instant « *stubbé »*. De plus, la plateforme *Spring* est mise de l’avant et favorise une approche MVC qui est un standard de développement Web.

De plus, le diagramme de classe est mis-à-jour et est une évolution des concepts élaborés. Finalement, une discussion est fournie et démontre les difficultés et les bons coups de l’équipe.

# Diagramme de classe

Le diagramme de classe initial fournis pour la première itération du projet a évolué. En effet, plusieurs concepts ont été revisité et bonifié afin d’offrir plus de souplesse au système en développement.

Le patron « *Transfer Object »* est utilisé comme monnaie d’échange entre un *JavaBean* et un objet du modèle. Pour ce faire un *JavaBean* implémente l’interface *Modelable* et en retour, l’entité implémente l’interface *Beanable*. De cette façon, l’information peut être transigée de la vue au modèle pour y subir les transformations propre aux logiques métiers.



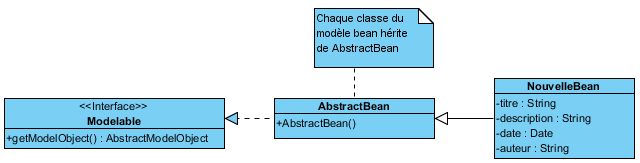


Figure 1 - Patron Transfer Object

En utilisant les outils offerts par la plateforme de test JUnit4, les fonctionnalités critiques de cette itération ont été identifiées, testées et inclus dans une suite de test automatisée. Une suite de test maîtresse, nommé *AllTests* constitue le point d’entrée. Les suites de tests enfants sont invoquer par cette dernière qui à leurs tours appelent leurs méthode de tests.

La suite de test fournis avec cette deuxième itération couvre l’ajout, la modification et la suppression de données au travers les interfaces gestionnaires. Elle couvre aussi l’implémentation du patron *« Transfer Object »*.

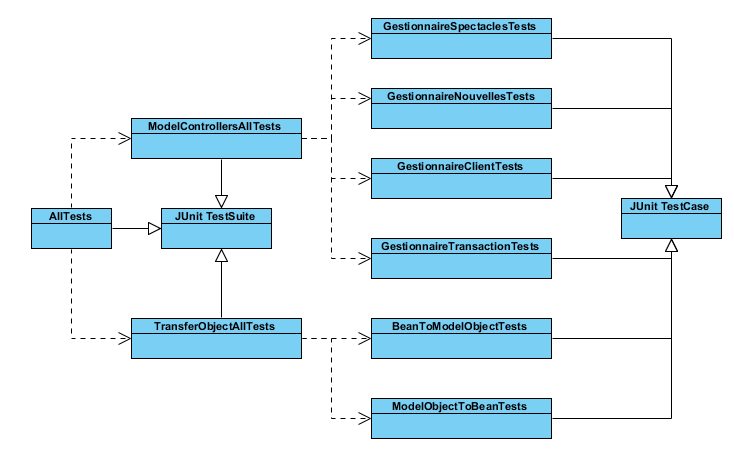
**

Figure 2- Architecture des tests unitaires

# Discussion

## Quel est la stratégie que vous avez utilisée pour associer un URL au contrôleur associé?

Chacun des contrôleurs est définis dans le fichier Web.xml associant le servlet approprié à l’URL demandée. Les méthodes de traitement sont affublé d’une annotation @RequestMapping qui offre plus de contrôle sur la gestion des URL.

# Quels patrons de conception Web ont été utilisés dans votre application? Si vous avez utilisé des frameworks, quels patrons sont implémentés par celui-ci?

Le patron *Transfer Object* à été mis en place dans une optique d’avoir une monnaie d’échange pour transiger les données de la page Web jusqu’aux classes Java. Cette approche ce veux évolutive afin d’appliquer une persistence aux données, soit l’objectif de la dernière itération du projet.

De plus, la plateforme Spring favorise une approche MVC et implémente elle-même sa propre version du patron *Front Controller.*

# Expliquez à quel endroit vous avez utilisé AJAX dans votre application.

# Avez-vous utilisé une technologie où un framework additionnel? Si oui, expliquez quel est le rôle de cette technologie dans votre application.

# Suite à l'expérience du laboratoire 2, quelles sont les éléments du diagramme de classe du laboratoire 1 que vous auriez planifié différemment?

# Quel a été la principale difficulté technique rencontrée dans le laboratoire?

# Comment a été faite la séparation des tâches dans votre équipe?

# De quelle manière avez-vous géré la gestion de la session?

# Conclusion