Table des matières

Stage DAFOP 2014 : intégration continue avec Jenkins

Introduction

Méthodes agiles

- Modèle en spirale : Livrer une application fonctionnalité par fonctionnalité.
- Travail en équipe : modèle de travail distribué. Le code est re-centralisé.
- Automatisation : utilisation d'un vieil outil **ANT** (possibilité d'utiliser **Maven** ou des scripts écrits en *Gradle* ou *langage shell*)
- Générer la documentation du projet de façon automatique

Configuration

Serveur (service)	Logiciel	Adresse IP	Nom
Un serveur de révisions	Scm-Manager	http://192.168.108.111:8080/scm/hg	badafop2014scm
Un gestionnaire d'intégration continue	Jenkins	http://192.168.108.112:8080	badafop2014jenkins
Un gestionnaire de projet	Redmine	http://192.168.108.113	badafop2014redmine

Démonstration

Scm-Manager

- Dépôt : demo1
- Sécurité : accès par compte anonyme
    - Option General : cocher la case "Allow Anonymous Access"

Redmine

- compte : daniele, bruno
- projet : demo1
- Membres :
- daniele : administrateur
- bruno : développeur

Problèmes : pour la synchronisation du dépôt

Jenkins

Créer les mêmes comptes et projets que Redmine

Définir le type de gestionnaire de révisions et l'adresse du dépôt du projet

- dépôt mercurial
- ajouter action au build : ant
    - cible : toutes-cibles

Ajouter une action après le build :

- Générer javadoc
- dossier build/javadoc
- construire = Lancer un build
    - console : le dossier n'existe pas
    - ajouter un mkdir dans build.xml et une property javadoc.api : build/javadoc

Exercice 1

Le but de l'exercice est de :

- organiser le travail grâce au logiciel de suivi de projet (créer le projet et faire des demandes).
- partager le code grâce à un dépôt sur un serveur de révisions (otut cela est expliqué dans d'autres documents : papier et électronique);
- organiser et créer les branches : default, stable, maintenance, prep-release
- gérer les ajouts de caractéristiques les unes après les autres :
    - d'abord, type de bien dans la branche f-domaine-typedebien,
    - puis Commune dans f-commune,
    - puis ClasseEnergetique....,
    - puis Bien.....,
- intégrer tout cela dans un "job" Jenkins pour générer une documentation commune et lancer des tests d'intégration

Voici le diagramme des classes de l'exercice ex1 :

- Écrivez le code des classes *TypeDeBien*, *ClasseEnergetique*, *ClasseEnregtiqueAbstraite*, ... et *Commune*.
- Écrivez le code de test de ces classes et notamment les méthodes *toStirng()*, *equals()* et *hashCode()*.

La méthode *equals()* est complexe, car il y a plusieurs conditions à respecter : on vous demande d'écrire d'abord le code de la classe de Test en séparant les conditions testées en 2 (ou 3) et d'écrire chacun(e) la moitié des méthodes de tests de cette méthode *equals()*. Il faudra, ensuite, fusionner votre code.

Le contrat général des méthodes *equals()* et *hashCode()* en Java est ici.

La documentation de ces classes est ici.