TP1: Tests unitaires et Automatisation

L'objectif de ce TP est d'illustrer les outils d'environnement de développement souvent indispensables notamment pour les tests unitaires ou pour automatiser la construction d'application.

Les concepts abordés dans ce TP sont universels, cependant nous allons les implémenter dans le contexte de la programmation java.

Pour les tests unitaires nous avons choisis JUnit et pour la construction d'applications Ant. Nous avons choisi Ant qui présente clairement les concepts à illustrer bien qu'aujourd'hui largement concurrencé par Maven.

Installation Ant

1. Aller sur la page d'accueil du site ANT : http://ant.apache.org. Téléchargez la dernière version au format zip.
2. Dézippez. Notez l'arborescence globale Le répertoire qui nous intéresse est le répertoire bin dans lequel se trouve ant.bat.
3. Analysez le contenu du répertoire lib. Le jar le plus important est ant.jar. Il contient l'ensemble des core tasks (taches principales) d'ANT.
4. Créez la variable d'environnement ANT_HOME qui contient le lien vers le répertoire ANT.
5. Ajoutez le répertoire bin d'ANT à votre variable d'environnement PATH utilisateur.
6. Lancez une Nouvelle invite de commande MS-DOS entrez la commande : ant. S'il y a une réponse du type *no build found* c'est qu'ANT est bien installé.

INFO : Ant recherche systématiquement dans le répertoire à partir duquel il est lancé fichier xml (par défaut build.xml) dans lequel des taches (target) sont définies.

7. Vous pouvez connaitre la version utilisée d'ant via la commande ant –version.

Automatisation de tests avec Junit et Ant

8. Téléchargez la dernière version de junit et copiez/collez le jar de Junit (ex junit-4.4.jar) dans le répertoire ANT_HOME\lib
9. Vérifiez avec la commande **ant -diagnostics** que le jar est désormais pris en compte par ANT.
10. Telechargez le projet se trouvant ici : http://bit.ly/2dtozo5. L'archive contient les sources d'un projet ainsi que les tests unitaires.
11. Localisez et analysez les classes de test. Quelles sont les caractéristiques de ces classes ?
12. Localisez le fichier build.xml. Identifiez les différentes cibles (*target*) définies dans le fichier et spécifiez leur rôle.
13. Quelle cible permet de nettoyer l'arborescence des répertoires générés automatiquement ? Exécuter la cible. Qu'observez-vous ?
14. Quelle cible permet d'automatiser le processus depuis la mise en place de l'arborescence jusqu'à la génération des rapports de test ? Exécuter la cible.
15. Combien de tests ont été exécutés ? Combien d'erreurs ou d'échecs sont rapportés par la commande.
16. Localisez les rapports de test au format HTML.
17. Générer la documentation du projet au format HTML. Localisez et affichez la documentation de la classe TaxRate dans un navigateur.

UFRMI
diallo.med@gmail.com

18. Modifiez la méthode calculateTax de TaxRate.java par le code suivant :

```
public double calculateTax(double totalRevenue) {
return getApplicableAmount(totalRevenue) * rate /2;
}
```

19. Régénérez le rapport de test. Que constatez-vous ?
20. Quel bilan tirez-vous de ce TP ?

Reference

Tutorial Objis

UFRMI
diallo.med@gmail.com