Test unitaires avec JUnit

JUnit

Jérôme Cheynet

Les test unitaires : théorie

- Tester les méthodes d'une classe
- Un test par méthode, un test case par classe
- Un test case fonctionne séparemment de l'application
-
- Avantages
  - Composants validés : meilleure qualité du produit final
  - Evite un bug causé par une régression
  - Force à séparer interface graphique et implémentation

Les tests unitaires : vous en avez déjà fait

- Exemple : classe calculatrice
  - Add seulement
- Problèmes :
  - Nombreux Affichages “Scroll blind”
  - Demande beaucoup d'efforts

Qu'est ce que JUnit ?

- Framework pour faire des tests : standardise la façon de faire les tests unitaires
- Standardise :
  - Les classes de tests (à vous de les écrire)
  - L'affichage des résultats des tests (programme d'affichage fourni)

Ecrire un test

- Classes de tests “standardisées”
  - Classe qui étend junit.framework.TestCase
  - Ne contient pas de main
  - Contient des méthodes testXXXXXX()
    - Chargées automatiquement
  - Les méthodes doivent contenir des assertions
    - assertTrue(bool condition)
    - assertEquals(int a, int b)
    - fail()

Ecrire un test

- Exemple
  - Calculatrice simple (add)
  -
  - Configuration : Eclipse installé
  - Eclipse intègre un Wizard
  - Eclipse permet de lancer l'interface graphique JUnit sur une classe de test

JUnit appliqué à un projet

- Convention de nommage
- Convention pour structurer le projet
- Convention pour réaliser les tests
  - Travail en binome autour d'une spécification
  - Interface puis tests puis implémentation
- Test si possible non triviaux
- Test des méthodes publiques
- Exécuter les tests en même temps que la compilation

JUnit et Ant

- En pratique, la barre vert ou rouge est trop lente : utiliser l'interface texte
- Sous Eclipse, créer un fichier build.xml
- Permet de séparer les tests des classes normales
  - Src/Main
  - src/Tests
-

JUnit et Ant

- Démonstration
  - Classe “Calculatrice” pour calculer 4 opérations
  - Classe “Calculatrice étendue” pour faire le carré d'un nombre
  - 2 test case
  - 1 test suite
  -
  - Compilation avec Ant

JUnit et Ant
Ant lance le testrunner de JUnit

```
<target name="runtests" depends="compiletests" if="junit.present">
        <java fork="yes" classname="junit.textui.TestRunner"
            taskname="junit" failonerror="true">
            <arg value="fr.umlv.exposeJUnit.calculators.AllTests"/>
            <classpath>
                <!-- TestRunner se trouve dans junit.jar -->
                <pathelement location="C:\Program
                                       Files\junit3.8.1\junit.jar"/>
                <!-- Les classes à tester sont dans un jar -->
                <pathelement location="build/lib/${app.name}.jar" />
                <!-- Les classes de test sont dans un répertoire -->
                <pathelement location="build/testcases" />
            </classpath>
        </java>
    </target>
```

Tips pour Ant

- Ant ne trouve pas un compilateur javac : deux solutions
  - Spécifier à eclipse une autre machine virtuelle
  - Windows/Preferences/Ant/Runtime/Runtime Classpath : ajouter lib/tools.jar qui est dans le répertoire java home
- Ant ne connaît pas JUnit : deux solutions
  - Configurer Ant via Eclipse : Window / Preferences / Ant / ajouter le jar de junit
  - Spécifier le classpath dans le fichier build.xml
- Attention avec les accents
  - Si vous ne spécifiez pas le jeu de caractère utilisé, il ne doit pas y avoir d'accents même dans

Présentation JUnit

Questions / réponses