INF2050 – Outils et pratiques de développement logiciel

Tests unitaires

Jacques Berger

© 2012-2019 Jacques Berger

Objectifs

Introduire la pratique des tests unitaires

Introduire JUnit

© 2012-2019 Jacques Berger

Prérequis

Java

Refactoring

© 2012-2019 Jacques Berger

Tests

Test unitaire

Un petit test qui ne vérifie qu'une petite partie du logiciel

Test fonctionnel

Vérifie une fonctionnalité du système, peut être fait à partir de l'interface utilisateur

© 2012-2019 Jacques Berger

Tests

Test de régression

Vérifie que la fonctionnalité n'a pas été brisée par un changement

Test de charge

Habituellement pour les applications web ou distribuées, vérifie le comportement de l'application lors de forts achalandages

© 2012-2019 Jacques Berger

Tests

Test d'intégration

- Vérifie l'intégration de différents composants du système
- Reproduire l'environnement de production

Test d'acceptation

- Le client approuve que la fonctionnalité développée correspond à ce qu'il voulait

© 2012-2019 Jacques Berger

Tests

Toutes les formes de tests sont recommandées dans un projet de développement logiciel

© 2012-2019 Jacques Berger

Tests unitaires

Du code pour tester le code

La forme de test la plus légère, avec la portée la plus limitée

Ne s'applique qu'à une seule fonctionnalité d'une classe

Plusieurs tests unitaires peuvent être nécessaires sur une méthode

© 2012-2019 Jacques Berger

Tests unitaires

Sert de test fonctionnel à très petite échelle

Sert de test de régression

© 2012-2019 Jacques Berger

Tests unitaires

Sur une classe, on peut tester :

- La fonctionnalité
- La non-fonctionnalité
- La gestion des erreurs
- Les exceptions
- Les cas limites
- Les cas hors bornes

© 2012-2019 Jacques Berger

Tests unitaires

Les propriétés d'un test unitaire :

- Exécution très rapide
- Code simple
- Indépendant des autres tests
- Doit pouvoir s'exécuter en tout temps

© 2012-2019 Jacques Berger

Tests unitaires

Un test unitaire ne doit pas :

- Manipuler un fichier
- Traiter avec une base de données
- Effectuer une communication sur un réseau
- Dépendre d'un environnement de test
- Communiquer avec le système d'exploitation

© 2012-2019 Jacques Berger

Vocabulaire

Classe du domaine : Une classe contenant de la logique qu'il faut tester

Classe de test : Une classe qui contient les tests pour une classe du domaine

Suite de tests : Un ensemble de plusieurs classes de tests

© 2012-2019 Jacques Berger

Avantages

Permet de détecter les erreurs plus tôt

Transfert de connaissances

Facilite la maintenance et le refactoring

Très payant à long terme

© 2012-2019 Jacques Berger

Inconvénients

Couplage fort entre la classe du domaine et la classe de tests

Plus de code à maintenir (code de test)

Les tests peuvent être bogués

Activité souvent difficile

© 2012-2019 Jacques Berger

Framework

Le framework de tests unitaires habituel avec Java est JUnit

© 2012-2019 Jacques Berger

Assertions

On vérifie les résultats dans un test unitaire à l'aide d'assertions

Une assertion est une condition qui doit toujours être vraie

© 2012-2019 Jacques Berger

Bonnes pratiques

Entretenir le code de test comme si c'était du code de production

- Éliminer la duplication
- Faire du refactoring
- Appliquer des patrons de test

Essayer de n'avoir qu'une assertion par test

© 2012-2019 Jacques Berger

Bonnes pratiques

Avoir une bonne couverture de tests

Exécuter nos tests après chaque modification du code

© 2012-2019 Jacques Berger

Bogues

Un bogue est un test oublié!

Lorsqu'on détecte un nouveau bogue, on tente de l'isoler dans un test unitaire

Ce test nous assure qu'on ne réinjectera pas le bogue une deuxième fois dans le code

© 2012-2019 Jacques Berger

JUnit

JUnit est une plateforme xUnit pour Java

On retrouve des implémentations de xUnit dans plusieurs langages

© 2012-2019 Jacques Berger

Emplacement

En général, on regroupe les tests pour une classe du domaine dans une classe de tests JUnit

Cette classe de tests peut être placée à peu près n'importe où

© 2012-2019 Jacques Berger

Emplacement

Pratique courante : on crée un répertoire test à la racine (au même niveau que src), ensuite on crée un package de tests pour chaque package du domaine qu'on veut tester

On place la classe de tests dans le même package que la classe du domaine, mais pas nécessairement dans le même répertoire

© 2012-2019 Jacques Berger

Test unitaire

Exemple

```
public class SqlTransformerTest {

    @Test
    public void testTransformIdentifierNormal() {
        Assertions.assertEquals(
            "FIRST_NAME",
            SqlTransformer.transformIdentifier("first_name"));
    }
}
```

© 2012-2019 Jacques Berger

Annotations

Les annotations Java ajoutent de la flexibilité à JUnit

Les méthodes de test doivent avoir l'annotation @Test

© 2012-2019 Jacques Berger

Annotations

On peut ignorer un test avec @Ignore

On peut ajouter un timeout qui fera échouer le test si la méthode est trop longue à s'exécuter, le timeout est en millisecondes @Test(timeout=10)

© 2012-2019 Jacques Berger

Annotations

@Before indique que la méthode sera exécutée avant chaque test, habituellement nommée setUp

@After indique que la méthode sera exécutée après chaque test, habituellement nommée tearDown

© 2012-2019 Jacques Berger

Annotations

@BeforeClass exécutera la méthode une seule fois avant l'ensemble des tests de la classe

@AfterClass exécutera la méthode une seule fois après l'ensemble des tests de la classe

© 2012-2019 Jacques Berger

Assertions

fail : fait échouer le test

assertTrue : fait échouer le test si le paramètre vaut false

assertFalse : fait échouer le test si le paramètre vaut true

© 2012-2019 Jacques Berger

Assertions

assertEquals : vérifie l'égalité de 2 valeurs

assertNull : vérifie la nullité

assertNotNull : vérifie la non nullité

assertSame : les deux paramètres sont la même instance

assertNotSame : les instances sont différentes

© 2012-2019 Jacques Berger

Pratique

On exécute les tests aussi souvent que possible

Toutes les classes ne sont pas faciles à tester, certaines sont presque impossibles à tester

Un refactoring pourrait être nécessaire avant de pouvoir créer un test sur une méthode

© 2012-2019 Jacques Berger

Plus loin...

jUnit

http://www.junit.org/

© 2012-2019 Jacques Berger