

Test Driven Development ou la programmation pilotée par les tests en Java

Référence : TDDJ

Les objectifs de qualité ambitieux fixés en début de projet résistent rarement aux contraintes du quotidien. Comment maintenir la qualité d'un code qui tend naturellement à diminuer, si les tentatives d'amélioration se traduisent par des régressions incontrôlées ? Le développement piloté par les tests (TDD) propose une solution radicale pour reprendre la maîtrise du code et de la conception : les tests systématiques sont écrits avant le code à tester ! Le retour sur investissement peut être important (qualité du code, non régression, évolutivité, maintenabilité). Encore faut-il éviter les écueils d'une mise en oeuvre inadaptée de la démarche.

Après ce cours, vous saurez utiliser des tests automatisés comme moyen de spécification, de conception et bien sûr de test. Vous saurez développer des tests pour du code existant hérité. Vous connaîtrez les techniques et outils, tels les doublures et Mocks, pour développer efficacement en TDD.

Vous verrez le TDD à l'oeuvre au travers d'exemples concrets et d'outils disponibles sur la plateforme Java. Une étude de cas réaliste vous permettra d'acquérir les réflexes du TDD, d'aborder les divers problèmes qui se posent aux développeurs en TDD et de mettre en oeuvre les bonnes pratiques, des plus simples aux plus élaborées.

Vous allez apprendre à :

- Découvrir les principes fondamentaux et les bonnes pratiques du TDD
- Utiliser JUnit dans une approche TDD
- Mettre en oeuvre les divers types de tests automatisés
- Utiliser des techniques avancées d'écriture de tests
- Mettre en oeuvre le TDD en présence de code hérité (legacy)
- Appliquer le TDD dans des contextes spécifiques (bases de données, IHM)
- Pratiquer le Refactoring d'un code développé en TDD

Durée : 3.0 jours - 21.0 heures

Audience :

Développeurs Java, responsables tests, chefs de projet, responsables qualité

Pré-requis :

Pratique de la conception objet

Pratique du développement avec Java ou avoir suivi le cours JOD ou IJOP

Méthode pédagogique : 60% de travaux pratiques

Programme détaillé :

- Object Builder

Le test dans le processus de développement

- Processus, qualité, tests
- Tests et agilité
- Tests et spécifications

Test de code hérité

- Qu'est-ce que du code hérité ?
- Cycle d'évolution du code hérité

Tests automatisés avec le Framework JUnit

- Le besoin d'un Framework de test
- Le Framework JUnit
- Les Matchers Hamcrest
- Bonnes pratiques associées à JUnit

TDD dans des situations particulières

- Tests en présence d'interface utilisateur
- Tests en présence de bases de données

Principes fondamentaux du TDD

- Le cycle de développement du TDD
- Test First
- Refactoring

Tests fonctionnels avec FitNesse

- Tests fonctionnels et TDD
- Écriture de tests fonctionnels exécutables avec FitNesse

Stratégies de Test First

- Tests comme moyen de spécification
- Tests comme moyen de conception
- Tests indépendants

Styles de TDD

- Tests basés sur l'état ou le comportement
- Utilisation de doublures
- Outside-In vs. Inside-Out

Écrire du code testable

- Composition plutôt qu'héritage
- Éviter le code statique
- Inverser les dépendances

Couverture des tests

- Les axiomes sur la couverture des tests
- Combien de tests faut-il écrire ?
- Outils de couverture

Mocks et doublures

- Quand les utiliser
- Types de doublures
- Bibliothèques de Mocks

Le Refactoring en TDD

- Quelques "mauvaises odeurs"
- Techniques de Refactoring en TDD

Techniques d'écriture des tests

- Langage universel