Test Driven Development en Java

TDD sans concession

Cette formation se veut à la fois constructive et critique concernant le développement à l'aide du "Test Driven Development".

Elle vous permettra d'identifier les forces et les faiblesses du processus TDD en fonction de votre projet. Nous aborderons également les techniques et outils poussés par le TDD, dont certains peuvent être utiles hors TDD.

Les objectifs de la formation

- Définition du TDD
- Comprendre les risques induits par le TDD dans un projet de développement
- Pratiquer le TDD tout au long d'un projet
- Comprendre les outils et techniques (Patrons de conception, Outils de tests unitaires et intégrations, Outils de Mocking)
- Comprendre la gestion des tests et de leurs statuts. Comprendre l'Intégration continue

Participants

- Développeur
- Chef de projet technique

Prérequis

- Savoir programmer en Java

Spoc pratique
Travaux pratiques continus

Durée
3 jours

Votre animateur
Olivier Charles

Code Kertify : ASD

S'inscrire

www.kertify.com/inscription

Le programme de notre formation

1. Définition et principes du TDD

- Le test dans le processus de développement. Processus, qualité, tests. Typologie des tests.
- Origine du TDD. L'agilité et les tests.
- Cycle de développement. Les 3A.
- Gestion des exceptions.
- Refactoring et conception émergente.
- Gestion des scénarios. Gains du TDD ?
- **Travaux pratiques.** Conception et intégration de tests dans le cycle de développement d'un projet.

2. Tests automatisés avec le framework JUnit

- Le besoin d'un framework de test. JUnit.
- Alternatives (TestNG) et outillage complémentaire.
- Bonnes pratiques associées à JUnit.
- **Travaux pratiques.** Mise en œuvre de JUnit.

3. Les bonnes pratiques de développement Agiles

- TDD et gestion des données SGBDR, des interfaces graphiques, des interfaces Web.
- **Travaux pratiques.** Mise en œuvre de pratiques.

4. Les objets Mock et Stub

- La théorie.
- Application de la théorie sans utiliser de bibliothèque.
- Découverte des bibliothèques du marché.
- Etude en détail de Mockito.
- **Travaux pratiques.** Utilisation des objets Mock.

5. Techniques d'écriture de tests

- Fixtures. Qualités d'un code de test.
- Tests basés sur la responsabilité, l'implémentation.
- Styles de TDD.
- **Travaux pratiques.** Améliorer la qualité des tests écrits.

6. Test de code hérité

- Qu'est-ce que du code hérité ?
- Cycle d'évolution du code hérité.
- Tests fonctionnels avec Fit et FitNesse.
- Tests fonctionnels et TDD.
- Exécution de tests fonctionnels avec FitNesse.

7. Les outils

- Les outils Open Source et commerciaux.
- Architecture matérielle de tests.
- Etude d'un outil d'intégration continue.
- Etude et choix d'un intégrateur continu.
- Etude d'un outil de couverture de test.
- Etude d'un outil de gestion des tests et de communication entre MOA et MOE : FitNesse ou Gherkin/Cucumber
- **Travaux pratiques.** Mise en œuvre de plusieurs outils