Sparks.

Programme de formation
TDD : Test-Driven Development

• Objectifs

Cette formation sur le TDD, ou Test-Driven Development pour développement piloté par les tests, permettra aux développeurs, chefs de projet ou encore ingénieurs qualité de comprendre comment mettre en oeuvre ce processus de développement. Basé sur la répétition de cycles très courts, le TDD est une technique de programmation très utilisée dans un contexte Agile. A l'issue de cette formation de 3 jours, les apprenants maîtriseront les meilleures pratiques pour créer des logiciels en suivant les principes du Test-Driven Development, et seront en mesure de mettre en application les acquis pédagogiques dès leur retour en entreprise. A noter que cette formation est proposée en Java, mais les principes sont applicables sur .NET (avec le langage C#), dans des environnements PHP, Ruby ou encore Python.

• Pré requis

Une expérience en programmation Java ou C#. Des connaissances de base sur les process Agiles (Scrum ou XP) sont également conseillées.

• Durée

3 jours

• Public

Architectes, Architectes-techniques, Chefs-de-projet, Developpeurs, Testeurs

•Plan de formation

Introduction à la formation TDD : découvrir le développement piloté par les tests

Objectifs de la formation TDD, attentes des participants
Les différents types de tests logiciels : unitaires, d'intégration, fonctionnels...
Frameworks et outils pour l'automatisation des tests
Retour sur la démarche Agile et ses composants (user stories, kanban board, tâches, burndown chart...)
Qu'est-ce que le TDD ? Approche et principes fondamentaux
Cycles de développement
TDD et eXtreme Programming (XP)
Exemples de cas pratiques : présentation des stagiaires, du formateur ou de la formatrice et échanges sur les différents contextes professionnels. Discussions autour des bonnes et mauvaises pratiques de développement, démystification du TDD et échanges sur les perspectives d'application des acquis de la formation en entreprise.

Tests unitaires

Vue d'ensemble, principe et structure du test unitaire
Différences entre tests unitaires et tests d'intégration
La famille xUnit : JUnit, PHPUnit, PyUnit...
Les rôles des développeurs et testeurs dans le test unitaire
Exécuter des tests automatisés
Niveaux de tests
Exemples de cas pratiques : configuration de l'environnement de développement (IDE) et d'un projet, exécution de tests unitaires, découverte du framework JUnit et manipulation d'attributs de test.

Techniques de doublure de tests

Les objets factices (ou fantaisie) et leur intérêt

d'utilisation
Dummy, stub, fake, spy et mock : comprendre et utiliser les différentes méthodes
Les bibliothèques disponibles
Exemples de cas pratiques : utilisation d'un Mock pour simuler un appel de méthode, d'un Stub pour résoudre un problème de données de tests.

Code legacy et refactoring : quand, comment et pourquoi réécrire le code source d'un programme ?
La dette technique et ses conséquences
Les principales méthodes et le cycle du refactoring
Exemples de cas pratiques : détection de « bad smells », mise en application de différentes techniques de refactoring liées au TDD.

Mise en oeuvre du TDD
L'intérêt de tester avant vs tester après
Par où commencer ? Revue des étapes de base
Le rythme du TDD : rouge, vert, refactor
Principes de conception
Patterns TDD ("Fake it 'til you make it", "Triangulate", "You ain't gonna need it"...)
Antipatterns ("Succès inattendu d'un test", "Ecrire des tests trop grands"...)
Principes SOLID dans un projet TDD
Les différents styles de TDD
Exemples de cas pratiques : simulation d'un projet conduit en mode TDD, manipulation de patterns, pair-programming dans un cadre TDD avec la technique du ping-pong pair programming, mise en oeuvre de divers outils.

Synthèse et retour sur les points clés de la formation