

Tests unitaires en Java avec JUnit

Référence : TUJU

L'objectif de qualité qui s'impose dans toutes les activités de l'entreprise devient une préoccupation centrale dans le développement logiciel où la recherche du zéro défaut se décline en grande partie sous forme de tests. Le principe Lean suggérant de supprimer les défauts dès leur apparition incite à tester le code dès qu'il est écrit.

La mise en place de tests unitaires automatisés répond à ce besoin mais recèle de nombreux pièges. On entend trop souvent parler de projets dans lesquels les équipes ont tenté l'aventure mais ont abandonné en cours de route par manque de temps, à cause des coûts de maintenance des tests, etc. Ce cours apporte des réponses en méthodologie et outils pour réussir une mise en place effective des tests unitaires automatisés dans vos projets.

Ce cours répond aux questions suivantes : "Quels bénéfices peut-on tirer de la mise en place des tests unitaires automatisés ?", "Quel est l'outillage à mettre en oeuvre ?", "Comment exploiter au mieux les capacités de JUnit ?", "Comment intégrer les tests unitaires dans le processus de développement ?", "Jusqu'où faut-il tester ?".

Vous allez apprendre à :

- Positionner les tests unitaires dans le processus de développement
- Exploiter les possibilités du Framework JUnit
- Améliorer votre code grâce aux tests unitaires automatisés
- Ecrire du code testable
- Choisir un niveau de test acceptable

Durée : 2.0 jours - 14.0 heures

Audience :

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

Pré-requis :

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

Méthode pédagogique : 60% de travaux pratiques

Programme détaillé :

Le test dans le processus de développement

- Processus, qualité, tests
- Typologie des tests

- Combien de tests faut-il écrire ?
- Stratégies de test
- Tests de régression
- Outils de couverture

Tests unitaires automatisés avec JUnit

- Le besoin d'un Framework de test
- Le Framework JUnit
- Cas et suites de tests avec JUnit
- Alternatives (TestNG) et outillage complémentaire

Quelques problèmes liés au test dans des situations particulières

- Tests Web
- Tests EJB
- Tests en présence de bases de données
- Tests d'interface utilisateur
- Tests et multi-threading

Bonnes pratiques associées aux tests unitaires

- Bonnes pratiques associées à JUnit
- Liste de tests
- Tests isolés
- Granularité raisonnable
- Refactoring

Écrire du code testable

- Composition plutôt qu'héritage
- Éviter le code statique
- Isoler les dépendances
- Inversion of Control
- Code hérité (legacy)

Les Mock Objects

- Quand les créer
- Types de Mock
- Types d'implémentations
- Bibliothèques de Mocks
- Styles de test unitaire

Couverture des tests

- Les axiomes sur la couverture des tests
- Types de couverture