JUnit pour tests unitaires et d'intégration

Cette formation permet de comprendre les enjeux et les techniques des tests unitaires et d'intégration, avec la mise en Suvre de jUnit et d outils complémentaires, comme les Mock Objects ou Maven pour l automatisation.

Elle aborde aussi toutes les bonnes pratiques nécessaires à la réalisation de tests efficaces et à l'élaboration d'une architecture pleinement compatible avec les tests unitaires. A l'issue de cette formation, vous serez aussi en mesure d'exécuter vos tests dans un environnement d'intégration continue.

3 jours

Tarif

- intra : 3630 euros HT (maxi 6 participants)
- inter : 1590 euros HT

Principes et démarche

- Les enjeux de la qualité logicielle
- Les types de tests dans un projet
- L'intégration des tests dans la démarche
- Les tests en démarche agile : eXtrem Programming et SCRUM
- La pratique du TDD (Test Driven Development)

Bases du framework JUnit

- Présentation des tests unitaires
- Le framework jUnit
- Développer un cas de test
- L'initialisation et finalisation d'un cas de test
- La réutilisation des portions de test
- Les suites de tests et les runners
- Le traitement des exceptions

Assertions

- Les assertions de JUnit
- De meilleurs assertions avec HamCrest
- Améliorer la fluidité des assertions avec AssertJ

Mock Objects

- Nos tests sont-ils réellement unitaires ?
- Différencier les tests unitaires des tests d'intégration
- Le principe des objets de leurre (Mock)
- Les frameworks de Mock
- La mise en Suvre avec Mockito
- Les fonctionnalités supplémentaires de PowerMock

Bonnes pratiques de tests

- L'organisation en packages
- Les conventions de nommage
- L'indépendance et l'isolation des tests

- Trouver la bonne granularité
- Gérer la durée et la fréquence des tests
- Réaliser des tests aux limites

Automatisation des tests
- Le principe de l'intégration continue
- La place des tests en intégration continue
- L'automatisation avec Maven
- La configuration des plug-ins Surefire et Failsafe
- La mise en Suvre avec Jenkins CI

Bonnes pratiques pour l'écriture de code testable
- Le développement par composants
- La délégation plutôt que l'héritage
- Une gestion souple des dépendances avec l'inversion de contrôle et l'injection
- Le problème des méthodes *static*
- La gestion des dates

Couverture des tests
- Les métriques de couverture de tests
- Les objectifs de couverture
- L'évaluation de la couverture des tests avec JaCoCo et Sonar

Tests d'intégration
- La différence avec les tests unitaires
- L'intégration avec la base de données
- Les outils DbUnit et DbSetup
- Tester les applications Web avec HttpUnit et Selenium

Tests en architecture JavaEE
- Rappel sur les architectures JavaEE
- Les tests de composants EJB 3
- Les tests des classes d'affichage JSF
- Les tests des classes persistantes JPA
- Arquillian pour faciliter les tests d'intégration

Tests avec Spring Framework
- Tests unitaires avec Spring
- Bonnes pratiques Spring
- Spring / JUnit
- Gestion des scopes
- Ressources autonomes et mocks

Fonctionnalités avancés de JUnit
- Les Rules
- Les tests paramétrés et les théories
- L'organisation des tests en catégories
- Les outils complémentaires Unitils
- Les plugins pour Eclipse : MoreUnit et Infinitest

Synthèse et Conclusion

- Intégrer les tests unitaires dans la démarche
- Que faut-il tester en priorité ?
- Quels types de tests sont les plus rentables ?