

TESTS UNITAIRES CODE2UTF('038',0) INTÉGRATION CONTINUE EN JAVA AVEC JUNIT CODE STAGE : J-TES

OBJECTIFS

Développer un test unitaire et comprendre comment les tests unitaires s'intègrent dans les différentes phases d'un projet

Connaître les méthodes de développement liées aux tests unitaires et maîtriser les outils sous-jacents à la mise en oeuvre des tests unitaires (frameworks dédiés type JUnit, DBUnit, Mock Objects)

Développer des tests unitaires dans des environnements J2EE (EJBUnit ...)

Comprendre et mettre en place un environnement d'intégration continue

DURÉE

2 jours

PUBLIC

Architecte

Chef de projet technique

Concepteur, développeur

PRÉ-REQUIS

Connaissance du langage Java. Connaissances de base en UML (diagramme de classes et de séquences).

PROGRAMME

1. PRÉSENTATION DES TESTS UNITAIRES

Définition et objectifs

. Objectifs, intérêts et enjeux des tests unitaires.

. Positionnement des tests unitaires dans l'univers des tests applicatifs.

Méthodologie XP, Test Driven Development et Test First

. Présentation d'une méthodologie agile : eXtreme Programming.

. Présentation de la méthode Test First, principe de base des tests unitaires.

2. IMPACT SUR LE DESIGN

La méthodologie « tests unitaires » impose le respect de certaines règles dans l'écriture des classes applicatives.

Présentation des règles et avantages

Les enjeux du Design

Principe de Liskow sur l'héritage

Principe de l'inversion de contrôle

. « Utilisons des interfaces ! », injection de dépendances.

3. ÉCRIRE UN TEST UNITAIRE

Présentation du Framework Junit

. Historique et présentation générale.

. Structure d'une classe de test : présentation de la classe TestCase, constructeur, création du contexte de test, méthodes de test, autonomie des méthodes de test.

Apport de JUNIT 4

. Travaux pratiques : premier test avec Junit.

Autres frameworks

. DBUnit, TestNG, outil de mesure de couverture du code testé (code coverage).

. Travaux pratiques avec DBUnit.

Préconisation sur l'organisation des sources

Méthodologie de test

. Comment écrire des tests dits « gagnants » ?

. Faut-il chercher à tout tester unitairement ?

. Assertion simple, test de méthodes sans valeur de retour, test d'un domaine de validité, test des exceptions.

. Travaux pratiques : exercices portant sur chacune des méthodes de tests.

Self-Shunt

. Présentation de la méthode, intérêts, inconvénients.

Mock-object

. Présentation de la méthode, intérêts, inconvénients, services rendus, contrôle des mock-objects et simulation de comportement.

. Comment utiliser les implémentations postiches ?

. Injection des implémentations postiches ou technique d'extraction.

. Intérêt des conteneurs légers tels que Spring pour la mise en place des tests unitaires.

. Travaux pratiques avec Jmock.

Tests unitaires en environnement JEE (EJB)

. Tests des EJBs (EJBUnit).

4. INTÉGRATION CONTINUE

Présentation du principe de l'intégration continue.

Panorama des outils d'intégration continue.

Présentation, configuration et déploiement de tests unitaires automatisés.

5. SYNTHÈSE

Que peut-on attendre des tests unitaires pour le développement en architecture J2EE ?

Impacts des tests unitaires sur la qualité du code Impact des tests unitaires sur les équipes de développement.

Impact des tests unitaires sur la gestion de projet.

Bilan économique.

Principaux ouvrages et sites de référence.

