



Vos contacts :
Marie-Jeanne ou Marjorie
au : 05 61 34 39 80

JAVATES - Test et Intégration de code JAVA

Objectif	A l'issue de ce stage, vous aurez découvert les techniques et les moyens à mettre en œuvre pour assurer des tests logiciels et améliorer ainsi le processus de développement.
Pré requis	Pour suivre ce stage, il est nécessaire de connaître les bases du langage Java. Le suivi préalable de la formation Optimisation de code Java est vivement recommandé. La connaissance du framework Spring sera aussi un plus pour cette formation
Durée	3 jours

Contenu

Module 1 : Introduction

- Rappels sur le méthodologie de développement objets
- UML (Unified Modeling Language)
- Le processus unifié
- Le processus de développement est itératif, incrémental et dirigé par les cas d'utilisation
- Besoin de tests et d'intégration en continu
- On teste tout, tout le temps, quel que soit la granulosité
- Jeux de tests unitaires et de non régressions
- Les cas d'utilisation fixent les jeux de tests fonctionnels
- Nécessité des jeux de tests techniques
- Tests d'endurance
- Stress de la l'application
- Quelques rappels sur l'IDE Eclipse
- Le modèle de plugin Eclipse
- Aide à la génération de code source
- Possibilité de refactoring

Module 2 : Les différentes API de logging

- Introduction
- Nécessité de conserver les traces d'un jeu de tests
- Nécessité de filtrer les traces
- L'API de SUN (java.util.logging)
- Fonctionnement général
- Utilisation de l'API
- L'API Log4J (the Apache Software Foundation)
- Fonctionnement général
- Utilisation de l'API
- Editer le fichier de configuration de Log4J
- L'API JCL (Java Commons Logging - the Apache Software Foundation)
- Fonctionnement général
- Utilisation de l'API

Module 3 : Le framework JUnit

- Nécessité des jeux de tests unitaires
- Garantir le fonctionnement nominal
- Garantir la non régression de votre code
- Mise en oeuvre d'un jeu de test JUnit
- Intégration du framework au sein d'Eclipse
- Codage d'un test unitaire
- Exécution de vos tests
- Quelle granulosité de code tester ?
- Garantir un taux de couverture du code minimal

Module 4 : Utilisation de techniques de bouchonnage (MOCK Objects)

- Il faut réduire les dépendances entre composants
- Présentation de la problématique
- Couplage par interfaces
- Concepts généraux sur l'utilisation de bouchon
- Présentation générale (théorie, vocabulaire, etc)
- Comparaison entre pratique classique du test et pratique bouchonnée
- Présentation des frameworks JMock et EasyMock
- Bouchonnage de composants Web J2EE
- Tester un état
- Tester un comportement
- Utilisation du Framework SPRING
- Intérêts du framework Spring
- Exemple concret de bouchonnage via Spring

Module 5 : Utilisation des Design Patterns et autres patterns

- Les patterns de conception et d'implémentation
- Patterns et réduction des dépendances entre composants
- Unifier les développements de vos composants logiciels
- Les principaux Design Patterns
- Les patterns d'architecture
- Concepts de couches logicielles (3-tiers, n-tiers, ...)
- Pattern MVC (Model-View-Controller)
- Approche par composants

Module 6 : Outils graphiques de monitoring et de profiling

- La JConsole
- Présentation du modèle JMX (Java Monitoring eXtensions)
- Démarrage d'une JVM supportant le monitoring
- Lancement et attachement de la JConsole
- Les différentes possibilités de la JConsole
- TPTP (Test and Performance Tool Platform)
- Rappel sur l'utilisation de plugin Eclipse
- Présentation du plugin TPTP
- Test de détection de fuite de mémoire
- Mesure de performances d'une application Java
- Taux de couverture de code
- Analyse de logs
- Utilisation de ces outils dans le cadre des tests