Unit Testing Essentials using JUnit and EasyMock

Course Summary

Description

Unit Testing Essentials using JUnit and EasyMock is a three-day, comprehensive hands-on unit testing training course geared for developers who need to get up and running with essential unit testing skills using JUnit, EasyMock, and other tools. Throughout the course students learn the best practices for writing great programs in Java, using unit testing techniques. This comprehensive course also covers essential TDD topics and skills.

Objectives
At the end of this course, students will be able to:

- Understand what unit testing is and what it is not intended to cover
- Understand JUnit.
- Understand and use the JUnit Test Runner interface.
- Use JUnit to drive the implementation of Java code.
- Test applications using native IDE support.
- Best practices and patterns for unit testing.
- Understand JUnit's strengths and weaknesses
- Understand the role of debugging when done in conjunction with tests.
- Understand not only the fundamentals of the TDD using Java, but also its importance, uses, strengths and weaknesses.
- Understand how JUnit affects your perspective on development and increases your focus on a task.
- Learn good JUnit coding style.
- Create wellstructured JUnit programs.
- Understand how JUnit testing can be used for either state-based or interaction-based testing.
- How to extend testing with mock objects using EasyMock.
- Look at refactoring techniques available to make code as reusable/robust as possible.
- Discuss various testing techniques.

Topics

- JUnit
- Testing Tools and Techniques
- Appendix: Adding Testing to the Build Process

Audience

This is an intermediate-to-advanced level Java course, designed for developers who wish to get up and running on test-driven development immediately.

Prerequisites

Attendees should be familiar with Java and object-oriented technologies. Real world programming experience is a must.

Duration

Three days

ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Unit Testing Essentials using JUnit and EasyMock

Course Outline

I. JUnit

A. Lesson: JUnit Overview
   1. Purpose of Unit Testing
   2. Good Unit Tests
   3. Test Stages
   4. Unit Testing Vs Integration Testing

B. Lesson: Jumpstart: JUnit 4.x
   1. JUnit Overview
   2. How JUnit Works
   3. Launching Tests
   4. Test Suites
   5. JUnit Test Fixture

C. Lesson: @Test Annotation
   1. Test Execution Cycle
   2. Checking for Exceptions
   3. Using Timeouts

D. Lesson: Hamcrest
   1. About Hamcrest
   2. The Hamcrest Matcher Framework
   3. Hamcrest Matchers

E. Lesson: Parameterized Tests
   1. Injecting the Parameters
   2. Setting the Parameters
   3. Test Execution Cycle
   4. Observations

F. Lesson: Theories
   1. Writing Theory Enabled Tests
   2. Defining DataPoints
   3. Defining Theories
   4. Observations

G. Lesson: JUnit Best Practices
   1. "Good" Tests
   2. Bad Smells
   3. White-Box Unit Testing
   4. Black-Box Unit Testing
   5. Automation and Coverage

II. Testing Tools and Techniques

A. Lesson: Improving Code Quality Through Refactoring
   1. Refactoring Overview
   2. Refactoring and Testing
   3. Refactoring to Design Patterns
   4. Naming conventions

B. Lesson: Mocking of Components
   1. Why We use Test Dummies
   2. Mock Objects
   3. Working with Mock Objects
   4. Using Mocks with the User Interface
   5. Mock Object Strategies

C. Lesson: Mock Objects and EasyMock
   1. EasyMock Description and Features
   2. EasyMock Object Lifecycle
   3. Create/Expect Phase
   4. Replay/Verify Phase
   5. Mocking Complex Objects
   6. EasyMock HOWTO

D. Lesson: PowerMock
   1. PowerMock Description and Features
   2. PowerMock Object Lifecycle
   3. Mocking a Static Method

III. Appendix: Adding Testing to the Build Process

A. JUnit and Ant
B. The Ant JUnit Tag
C. Running JUnit Tests From Ant
D. Generating a JUnitReport