Java Testing with JUnit 5

Course Summary

Description

This course introduces experienced Java developers to the fundamentals and best practices in unit testing. It uses the JUnit 5 and Mockito libraries, both of which are ubiquitous in the Java community. It is intended for both developers who are new to testing, as well as those who are already familiar with it, but want more experience with testing using JUnit 5.

Note that JUnit 5 is substantively different than JUnit 4. Although the core testing principles are the same, there are substantial implementation differences, important new features, and a brand new extension model. Developers with JUnit 3 and 4 experience will be well-served by the course.

Testing with mocks is covered in detail, using Mockito. Mocking is a standard technique, and we'll go beyond the basics to introduce more flexible options, as well as Mockito's dependency injection facilities. We use the new MockitoExtension to integrate Mockito into JUnit tests.

We explore testing enterprise components in the persistence, service, and web layers, outlining the issues involved in each. In the persistence layer, we show how to set up and use an embedded database, and contrast this to using mocks or fakes as a standalone alternative. We examine the unique issues in database testing with regard to test repeatability and independence, using various techniques for auto-rollback of transactions.

In-container testing is introduced as a more meaningful approach to testing enterprise components, vs. standalone testing with mocks, and we examine this at each of the main layers in an enterprise application. Best practices are emphasized and demonstrated throughout, and our goal is to make you "test infected," so that your development effort is as much about good testing and writing testable code as it is about writing the business code itself.

All labs are done with the Eclipse IDE, and the lab instructions include detailed directions for setting up and using it. The standard applicaiton server used is TomEE, but it is available for other major app servers, including WildFly/JBoss AS and IBM WebSphere AS, with advanced notice and possible additional charge.

Objectives

After taking this course, students will be able to:

- Understand new JUnit 5 library structure, role of each component, and how they interact
- Configure IDE projects to run tests natively, and via Maven Surefire
- Write cohesive and effective tests, and design classes for testability
- Understand the full test lifecycle, and employ it to configure test fixtures
- Run tests using all available mechanisms: IDE, Maven, JUnit Console Launcher, Launcher API
- Use test discovery and filtering to define and run test plans, including conditional test execution
- Employ naming conventions at every level - test name, classname, display name
- Organize tests with assertion groups and nested tests
- Use test interfaces to apply good OO principles to testing
- Understand the new JUnit extension model, and how to write and use them
- Understand JUnit 4 compatibility and migration
- Use mock objects with Mockito to support isolated testing
- Explore Mockito's facilities for dependency injection of mocks
- Use argument matchers for more generalized testing with mocks
- Implement partial mocking with spies
- Understand the issues in testing enterprise components
- Understand the two basic approaches: standalone testing with mocks, and in-container testing
- Test database access components, using both fakes and an embedded database
- Understand the additional issues involved in testing the service and web layers
- Exercise best practices throughout the testing effort

Java Testing with JUnit 5

Course Summary (cont'd)

Topics

- Unit Testing with JUnit 5
- Writing and Running Tests (includes brief primer on Java 8 new features)
- Testing with Mocks
- Testing Enterprise Components

Audience

This course is designed for experienced Java developers to introduce them to the fundamentals and best practices in unit testing.

Prerequisites

Prior to taking this course, the student is expected to have a good working knowledge of Java and OO, including the use of interfaces, abstract classes, collections, factories, and generics. Experience with Java 8 lambda expressions is helpful, but not strictly required. A brief primer on the Java 8 features employed by JUnit is included in the course and covered if necessary.

Duration

Two days

ProTech "Charting the Course ...

protechtraining.com

... to Your Success!"

Java Testing with JUnit 5

Course Outline

I. Unit Testing with JUnit 5
- A. Overview
  1. Unit Testing and JUnit Overview
  2. New Features in JUnit 5
  3. JUnit 5 Library Components
  4. Naming Conventions and Organizing Tests
- B. Tests and Assertions
  1. Writing Test Methods
  2. Assertions
  3. Assertion Messages
- C. Test Fixtures and Test Lifecycle
  1. Creating and Using Text Fixtures
  2. Test Run Lifecycle: @BeforeEach and @AfterEach, @BeforeAll and @AfterAll
  3. Controlling Test Instances

II. Writing and Running Tests (includes brief primer on Java 8 new features)
- A. Additional Testing Needs
  1. Testing for Exceptions
  2. Setting Timeouts
  3. Assertion Groups
- B. Running Test
  1. IDE Support: Eclipse, IntelliJ IDEA
  2. Maven Configuration
  3. JUnit Platform Console Launcher
  4. Launcher API
  5. Test Discovery and Selection
  6. Display Names
  7. Grouping and Filtering with Tags
  8. Configuration Parameters
  9. Nested Tests
- C. Advanced Capabilities
  1. Custom Composed Annotations
  2. Inheritance with Test Classes
  3. Extensions
  4. Conditional Test Execution
  5. Parameterized Tests
- D. JUnit 4 Migration
  1. The Do-Nothing Case
  2. Using a JUnit 4 Runner
  3. API Changes
  4. JUnit 4 Runners and Rules
  5. JUnit 4 Test Suites
- E. Best Practices
  1. Testing Void and Private Methods
  2. Test Cohesion and Assertion Scope
  3. Characteristics of Good Tests
  4. Writing Testable Code
  5. Testing Anti-Patterns

III. Testing with Mocks
- A. Overview
  1. Mock Objects as Collaborators
  2. Mockito Overview
- B. Creating and Using Mocks
  1. Basic Steps in Mocking
  2. The Mockito Class
  3. Mock Creation with @Mock
  4. JUnit 5 MockitoExtension
  5. Stubbing
- C. Additional Capabilities
  1. Argument Matchers
  2. Partial Mocking with Spies
  3. Mocking the Unmockable
  4. Dependency Injection of Mocks

IV. Testing Enterprise Components
- A. Overview
  1. Unit Testing vs. Integration Testing
  2. Testing with Mocks vs. In-Container Testing
  3. Mocks vs. Fakes
- B. Testing the Persistence Layer
  1. Database Options: Installed, Embedded, Embedded-in-Memory
  2. Standalone vs. In-Container Testing
  3. Test Independence and Transaction Rollback
  4. In-Container Testing with Arquillian [Overview]
- C. Testing Services
  1. Similar Issues, Different Layer
  2. Working with External Resources
  3. The Argument for In-Container Testing
- D. Testing Web Components
  1. Interfacing with External Clients
  2. Difficulties in Standalone Testing with Mocks
  3. Manual vs. Automated Testing
  4. Automated Testing with Selenium [Overview]