



## TT3500: Mastering Test-Driven Development using JUnit (4 days)

**Duration:** 4 days

**Skill Level:** Intermediate+

**Focus:** Java5 or Java6 Applications

**Audience:** Experienced Java programmers

**Hands-On:** Extensive Hands-On Programming Labs; Expert lecture combined with open discussions and high-Level demonstrations

**Language / Tools:** Java 5 or Java 6/Delivered with most IDEs

**Mastering Test Driven Development using JUnit** is a four-day, comprehensive hands-on workshop geared for developers who need to get up and running with essential Test-driven development programming skills using JUnit and various open-source testing frameworks. Throughout the course students learn the best practices for writing great programs in Java, using test-driven development techniques. This comprehensive course also covers essential TDD topics and skills.

### ► Course Objectives: What You'll Learn

Students who attend **Mastering Test-driven Development using JUnit** will leave the course armed with the skills they require to develop solid Java programs, using sound coding testing techniques and best coding practices. This course quickly introduces developers to the features of JUnit and educates them regarding JUnit's strengths and weaknesses.

JUnit, and other testing frameworks based on JUnit such as Cactus, make it possible to write higher-quality Java code. It is a powerful tool designed to support robust, predictable and automated testing development in the Java enterprise application arena.

This course includes coverage of many of the essential JUnit capabilities, and can be tailored to focus exactly on the areas that you are interested in.

At the conclusion of this course, attendees will be able to:

- 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 test development.
- 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 the basics of JUnit, Cactus and other testing frameworks and how they relate to TDD.
- Learn to better control the development and quality of Java code.
- Understand how JUnit affects your perspective on

development and increases your focus on a task.

- Learn good JUnit coding style.
- Create well structured JUnit programs.
- Compile and execute programs using JUnit, Cactus, StrutsTestCase and DBUnit using the IDE of your choice.
- 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.

The following JUnit-based testing frameworks are examined:

- JUnit 3.8.1
- DBUnit
- jWebUnit
- StrutsTestCase
- EasyMock
- Cactus

### ► Experiential Learning – Course Structure

Throughout the three-day course, students will be led through a series of progressively advanced topics, where each topic consists of lecture, group discussion, comprehensive hands-on lab exercises, and lab review.

This workshop is about 50% hands-on lab and 50% lecture. **Multiple complete “mini-projects”** are laced throughout the course, designed to reinforce fundamental skills and concepts learned in the lessons, all working in the JUnit environment. Because these lessons, labs and projects are presented in a building block fashion, students will gain a solid understanding of not only

the core concepts, but also how all the pieces fit together in a complete application.

At the end of each lesson, developers will be tested with a set of review questions to ensure that he/she has fully understands that topic.

#### ► Audience & Pre-requisites: Who Should Attend

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

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

#### ► Related Courses – Suggested Options

**Take Before:** Students should have basic development skills and experience in the following topics, or attend these courses as a pre-requisite:

- TT2100 Core Java Programming for OO Developers (C++, etc)
- TT5100 Building J2EE Web Applications (Servlets, JSPs, JDBC, Security, etc.)

**Take After:** We offer a variety of introductory through advanced development, project management, engineering, architecture and

design courses. Students may want to consider the following topics as a follow-on to this course. Please contact us for recommended next steps tailored to your longer term education, project or development objectives.

- Students needing an essential J2EE follow up may take TT5100 Mastering J2EE Web Applications (Servlets/JSPs, Tags, JDBC, Security, etc.)
- Advanced Security topics
- Service-Oriented Analysis and Design
- Web Services – Intro through Advanced
- AJAX, XML or other Web Development topics
- Java EE topics: EJB3.0; Spring; Hibernate; Design Patterns & more.
- Advanced .Net developer topics
- Architecture & Analysis courses
- Software Engineering, Design or Project Management tracks

Please contact us for recommended next steps tailored to your longer term education, project or development objectives.

#### ► Delivery Environment

This course is nominally delivered using Eclipse, MyEclipse, and HSQL. However, any IDE that includes JUnit can be used and any front-end web framework can be used as an implementation technology for the labs.

---

## Workshop Topics Covered

---

#### Session: Test-Driven Development

- Overview of Test-driven Development
- The Problem
- The JUnit Solution
- Test, code, refactor, repeat
- The ROI of TDD
- Rationale
- Advantages
- Tools

#### Session: Testing Frameworks

- Integration Testing: jWebUnit/HttpUnit
- Presentation testing
- Integration testing
- jWebUnit
- jWebUnit/HttpUnit
- Testing Struts: StrutsTestCase
- Design of a Struts application
- StrutsTestCase
- Testing strategies

#### Session: Advanced TDD Topics

- Mock Objects and EasyMock
- Decoupling with Mock Objects
- Mock object frameworks
- EasyMock and JUnit
- Dependency Injection, Spring and Testing
- Dependency Injection and IoC
- The Spring Framework
- Mock Objects and Spring
- State-based vs. Interaction-based Testing
- State-based testing
- Interaction-based testing
- Dependencies vs. mock objects
- Interaction-based Testing

#### Session: Improving Code Quality Through Refactoring

- Refactoring
- Basics
- Samples of Refactorings
- Refactoring and Testing



**Session: Additional Testing Frameworks**

- Database Testing: DBUnit
- Issues related to database testing
- Persistence mechanisms
- DBUnit

**Session: Introduction to Spring (optional)**

- The Spring Framework
- Spring Basics
- Configuring a Spring bean
- Property Editors
- Constructor injection vs. Setter injection
- Wiring the collaborators

**Session: Advanced Refactoring (optional)**

- Advanced Refactoring
- Refactoring to Design Patterns
- Sample Refactorings
- Best Practices
- Refactoring
- Naming conventions
- Organizing test suites

**Session: Advanced Topics (optional)**

- Testing Business Rules
  - Fit
  - Fitnesse
  - Selenium
- Adding Testing to Your Build Process
- The Ant JUnit Tag
- Example Ant Build File
- Continuous Integration

**Session: Additional Testing Frameworks (optional)**

- Server-side Testing: Cactus
- Server-side testing
- Cactus: mock-container testing
- Cactus: in-container testing

