Implementing ATDD with Cucumber and Java

Course Summary

Description

Acceptance Test Driven Development (ATDD) has become a standard development technique used by Agile teams to work through a collaborative process to develop a set of acceptance tests that describe how the system under development should behave. These tests are written in a special language called Gherkin and executed using the Cucumber program.

This course focuses on the process of writing the executable step definitions in Java for the Cucumber program so that the acceptance tests can be run automatically, in much the same way JUnit runs unit tests automatically. All of the theory presented in this course is reinforced with examples, real world case studies, hands on exercises and projects.

The course assumes that the acceptance tests have been written and validated and walks through the process of writing and testing the automation code and then implementing the step definitions to run the tests through several different kinds of interfaces.

Topics

- How Gherkin works: Gherkin syntax and Step Definitions
- Agile automated testing – testing through interfaces
- Cucumber architecture and functionality
- How Cucumber executes Gherkin via Step Definitions
- The three stage test automation model: tests, interface and application
- Testing the tests with a test interface and application mock
- Verification of the step definition logic
- Building the test class around the step definition
- Using JUnit assertions for test evaluation
- Handling complex data with Data Tables and Java Collections
- Using Selenium to test through a web interface
- The Page Object Pattern
- Testing through Web Services
- Selective Scenario execution
- Cucumber reporting and formatting
- Responding to changes in an interface or the acceptance tests

Audience

The course is appropriate anyone who is involved in the implementation of automated ATDD tests.

Prerequisites

Students should be at least intermediate level Java programmers and should have some working familiarity with a Java IDE and test automation framework like JUnit or TestNG. The course uses Eclipse as the Java IDE and JUnit as the test framework.

Duration

Two days

ProTech "Charting the Course ...

protechtraining.com

... to Your Success!"

Implementing ATDD with Cucumber and Java

Course Outline

I. Gherkin Basics
- A. Gherkin as a structured acceptance test language
- B. Features and feature files
- C. Gherkin scenarios as test cases
- D. The Given, When, Then syntax for scenario steps
- E. The Background section
- F. Using And and But for readability
- G. How Cucumber executes a feature file
- H. Comments and feature file documentation

II. More Gherkin
- A. Using Scenario Outlines and Example Tables
- B. Moving complex test data with Data Tables
- C. Well structured Gherkin Scenarios
- D. Best practices in organizing Scenarios and Steps
- E. Best practices for using data in a scenario step
- F. When to use Scenarios, Backgrounds and Outlines
- G. Organizing the acceptance test artifacts for automation

III. Tests as Drivers of Development
- A. Unit testing and integration testing
- B. The Agile testing quadrants
- C. Testing through interfaces
- D. The three layer problem: test code, interface and application
- E. Implementation strategy for automation
- F. Unit testing the three layers

IV. Testing Behind the Interface and Mocks
- A. Unit testing the test code
- B. Using a mock application
- C. Working with a test interface
- D. Using mocks for interface testing

V. Cucumber Architecture
- A. Organization of a Cucumber project
- B. Organizing feature files and step definitions
- C. How Cucumber finds and executes step definitions
- D. Where Cucumber can fail and how to prevent it
- E. Matching step definitions with annotations using regular expressions.
- F. Managing complex data in Data Tables
- G. Running Cucumber in the IDE and at the command line

VI. Cucumber Test Logic
- A. Defining the test pass and fail criteria
- B. Using assertions to implement test decisions
- C. Using assertions to ensure test set ups are complete
- D. Mapping the Step Definition declarations to imperative code
- E. Running tests that depend on application state

VII. Selenium and Web Interface Testing
- A. Selenium basics and setting up a selenium project
- B. Before and After methods
- C. Navigating around the web app and running tests
- D. Using assertions to evaluate web results

VIII. Page Object Pattern
- A. Using a layer of indirection – pages as classes
- B. Setting up a page object project structure
- C. Exposing functionality of pages
- D. Page object factories

IX. Web Services
- A. Review of Service Oriented Architecture
- B. RESTful and SOAP based web services
- C. Best practice: Testing behind the interface
- D. Testing the web service transport implementation
- E. Working with JSon data
- F. Best practices for working with web services

X. Summary
- A. Review of the course
- B. How to move forward and apply what has been learned
- C. Other topics requested by students
- D. Assessment of knowledge and skills gained