

Selenium WebDriver Test Automation with Java

Java, Maven, TestNG, browser interactions and framework design

FA-0749

Software Development · DevOps, Cloud & Infrastructure

Introduction

Build and organise automated web-application tests using Java and Selenium WebDriver. The course combines a Java refresher with Maven project setup, TestNG, browser interactions, synchronisation and selected distributed-testing examples.

Develop a small reusable test framework with page objects and data-driven tests. Cucumber, Grid and CI/logging integrations are introduced through selected examples; browser automation is distinguished from specialist database and performance testing.

Learning outcomes

- Use core Java constructs needed for browser tests.
- Create a Maven/Selenium/TestNG project and troubleshoot its configuration.
- Locate elements, interact with browser UI and assert expected results.
- Apply suitable waits and browser/session cleanup.
- Organise parameterised tests, suites, listeners and execution reports.
- Implement a small page-object/data-driven framework.
- Explain Cucumber, Grid/parallel execution and CI/logging roles.
- Recognise where database/API checks and purpose-built performance tools are needed instead of WebDriver.

Prerequisites

- Programming experience in any language and basic file-system/internet knowledge.
- A compatible JDK, browser, Maven and IDE such as Eclipse, with approved setup permissions.
- Reliable internet and access to the designated application under test.
- Authorised remote access where the Grid/CI lab requires it; unrestricted root/admin privileges are not a general prerequisite.

Training outline

1. Module 1 – Java Foundations for Tests

- Java versions/platform context: application, mobile, web and integration roles.
- Algorithms, flow, logic, subroutines, dependencies and memory concepts.
- JShell: hello world, arithmetic, basic I/O, expressions, variables and data types.
- Operators, conditions and loops; selected coding exercises.

2. Module 2 – Maven Project and First Selenium Test

- JDK/Maven and Eclipse setup; create/configure a Maven project and resolve dependencies.
- Review the application under test and write a first test class/method.
- Run a TestNG test and debug project or test failures.
- WebElement, locators, WebDriver commands, invalid-locator behaviour and assertions.
- Explain Ant as an alternative build-tool context rather than a second full build track.

3. Module 3 – TestNG Organisation and Reports

- TestNG Annotations Introduction
- Xml File
- Execution Order of TestNG Annotations
- Skipping Tests in TestNG
- Prioritise Tests in TestNG
- Assertion in TestNG : Verification & Validation
- TestCase Grouping in TestNG
- Set TestCase Dependency in TestNG
- Test Parameterisation Using testng.xml File
- DataProvider : Test Parameterization

- Listeners in TestNG
- Run Multiple TestNG Suites
- Reporting in TestNG
- Review dependencies/priorities without making independent tests rely unnecessarily on execution order.

4. Module 4 – WebDriver and Rich Browser Interfaces

- WebDriver architecture, selected supported browsers and browser-specific options.
- Dynamic browser selection, JavaDoc use and close versus quit.
- Timeouts, implicit versus explicit waits and Ajax readiness; avoid mixing both wait strategies.
- Find one/multiple elements; CSS and XPath locators, visibility and scoped searches.
- Dropdowns, frames, browser tabs/windows, alerts and keyboard/mouse actions.
- Ajax suggestions and selected JavaScript-executor use.
- Inspect links and distinguish browser navigation assertions from direct HTTP status checks.

5. Module 5 – Framework Design and Distributed Execution

- Create page objects with reusable UI services and keep business assertions in tests.
- Introduce data-driven and hybrid organisation with selected test-data examples.
- Cucumber behaviour scenarios and a small integration example.
- Selenium Grid, parallel execution, isolated browser sessions and a selected cross-browser run.
- Introduce Jenkins CI execution, maintained Log4j logging and test reports through a supplied integration example.
- Database verification requires a separate database/API client, not WebDriver itself.
- Discuss why WebDriver is not recommended for representative load/performance testing; compare a purpose-built tool conceptually.
- Framework exercise and assessment.