Automation Testing and DEVOPS Conceptual Training - Session 6

Learn Automation Testing and DEVOPS Concepts in Depth

Session#6: How to Develop Selenium Automation Framework from Scratch

Trainer – Haradhan Pal

Agenda

- 15 Important Steps to Develop Selenium with Java Test Automation Framework from Scratch

Develop Selenium Automation Test Framework

In case user wants to start automation testing framework design, Selenium is often the first option that comes to tester mind. Being open-source with large community support, Selenium has established itself as the favorite choice of automation testers over the last decade with significant traction. Test Scripts in Selenium can be developed using popular programming languages like Java, Python, C#, JavaScript, PHP, and Ruby etc. Selenium WebDriver is a widely used automation testing framework used for cross browser (Chrome, Firefox, Safari etc.) automation testing of any web applications. Selenium Supports automation across multiple platforms like Windows, Macintosh, Linux, etc. User can use Selenium Grid, which helps run tests in parallel across different browser and platform combinations. Selenium Grid follows a hub-node concept where user can run the WebDriver scripts on a single machine (hub), but the execution will be done by multiple remote machines (nodes) via routing commands. Java comes with multiple complete automation frameworks like TestNG, JUnit, etc., that can be leveraged with Selenium for better coding, reporting, and maintenance.

Steps to Create Automation Test Framework

- **Install Java:** User need to install Java on their system to use Selenium with Java. User can download the latest version of Java from the official website: Once the installation is complete, set the environment variables.
- **Install an Integrated Development Environment (IDE):** Choose an IDE for any automation testing project. Some popular IDEs for Selenium with Java are Eclipse, IntelliJ IDEA, and NetBeans. Need to create a workspace which will be a place where user can store all recent projects.
- **Install Selenium WebDriver Jar File:** Download the latest version of Selenium WebDriver from the official website and configure it in the project. User can work with latest Selenium 4.
- **Create a Project in the IDE:** User need to have a project set up for writing and storing their test cases. Create a new project in the chosen IDE and add the Selenium API WebDriver JAR files to the project's class path. User can also create a maven-based project as well. So all the dependencies will be in the Maven POM (Project Object Model) file and the test suite can be triggered using maven commands.
- **Planning and Designing of Test Cases:** Before getting started with writing any automation tests, it is recommended that the QA teams invest time in creating a proper test plan. QAs should come up with all possible logical scenarios and create maximum test cases based on the end-user perspective. Define a test case using the test framework of their choice (JUnit, TestNG, etc.). The test case should include the Selenium WebDriver API calls to automate the testing process.

Steps to Create Automation Test Framework – Contd.

- **Choosing the Best-suited Locators:** While writing automation test scripts, a large chunk of the time is spent locating the web elements. Selenium offers us multiple strategies to locate a web element such as ID, Name, Link Text, Partial Link Text, XPath, CSS Selector, Tag Name etc.
- **Create Reusable Components:** Create reusable functions/methods such as login, logout, and data entry that can be used across multiple test cases.
- **Implement Page Object Model (POM):** Implement the POM design pattern to create a clean and maintainable test framework. The POM separates the test code from the UI elements and makes the test code more readable and reusable. In the Page Object Model, all the web elements, actions, and validations happening on each web page are wrapped into a class file called Page Object. The POM allows a clean separation between test and page-specific codes such as locators and layouts. Due to the centralized nature of the page-specific code, they can be reused throughout the test cases, reducing code duplication. Also, in case of any UI changes, the fix needs to be done only on the related Page Objects, not on dependent test cases making code maintenance easier. In POM, Base Class should consists set-up and tear-down operations like browser configurations, implicit and explicit waits handling, cookies deletion, etc. Each test classes containing the test scripts must extend this class.
- **Implement Data-Driven Testing:** Implement Data-Driven Testing using Excel sheets or CSV files to run the same test with different variety of data sets.

Steps to Create Automation Test Framework – Contd.

- **Add Logging and Reporting:** As the test suite becomes extensive, locating the failed test case and debugging becomes challenging. In these cases, reporting and logging practices can act as huge saviors. Add logging and reporting to the test framework to track test execution and generate reports. Logging helps user to understand their code better and hence makes debugging easier. Examining logs is the quickest way to analyze why user code faces a problem. Some of the popular log levels are debug, info, warning, error, and critical. Good logging saves time. However, if overdone, it can slow down test execution.
- **Incorporating Wait Commands:** The web browsers take some time to load the web page contents. The time taken for page load depends on many external factors like network speed, server issues, machine capabilities, etc. To tackle this, testers need to add some delay into the code so that the web element will be loaded and available before performing any action on it. In Java, testers often achieve this delay using Thread.sleep(). However, this is not the recommended practice. As the name implies, when the Thread.sleep() method is called, the thread goes to sleep for the specified amount of time and does absolutely nothing to pause the automation script. After the time is over, it moves on to executing the next line of code.
- **Capture Screenshots:** In automation, user can rely on the assertion messages that they print in case of failure. In addition to that, user can also have screenshot of the browser in case of failure due to assertion or unavailability of any web element.
- **Use TestNG Framework for Parallel Test Execution:** In actual project, user need to write, run and maintain hundreds to thousands of test cases. Each testing cycle will demand user to execute their test cases on multiple platforms and devices to ensure that the application behaves as expected under various conditions to assure the product quality before going live. It becomes highly desirable to reduce the overall test execution time, and one way to achieve it is through parallel testing. TestNG test automation framework allows user to run tests in parallel or multithreaded mode by utilizing the Multi-Threading concept of Java.
- **Debug, Run and Maintain the Test Cases:** Run the test cases and verify the results. Debug any errors and update the test cases as and when required. User should maintain the test scripts frequently to be sync with latest requirement/functionality change.

Steps to Create Automation Test Framework – Contd.

- **Integration with any CICD Tools:** The maximum ROI of test automation comes from its frequent execution either as a part of the CI/CD pipeline or running them often. Jenkins is one of the most widely used DevOps tools used in automating the complete life cycle of Continuous Integration, Delivery, and Deployment. Jenkins allows teams to execute a series of automated tests and builds. This ensures the code is always up-to-date and the automation suite is getting executed after every commit.

Benefits of using CICD Integration With Selenium Testing:

- **Quicker Release Cycle:** Pacing up the build and testing cycle will allow the team to get new features into production faster, and it also helps to reduce the testing execution time exponentially by triggering from the pipeline. Pipeline helps the organization to accelerate the release rate with utmost quality.
- **Ensures High Quality:** Any software development process that includes continuous testing is on its way toward establishing a critical feedback loop to go fast and build effective software by making the Selenium automated suite part of the pipeline. As user run their suite after every change done by the developer team, it always provides a quality check of each and every code pushed by the developers. The pipeline with continuous testing builds quality and reduces risk and waste in the software development lifecycle.
- **Simplify Rollbacks:** Integration of automation with the pipeline helps to run regression on code changes and if the tests fail then user can easily rollback the changes to working state. In most of the cases they basically rollback to the last successful build for stability.