Emerald labs test automation framework

Using Selenium-Java

EMERALD LABS

**Introduction to Test Automation**

***What Is Test Automation?***

The term automation refers to the automatic handling of various industrial processes. It indicates that there is little to no human intervention. When we define test automation in the IT sector, it means performing tests on applications via different automation tools to check how applications behave or respond to different actions. These tools can be both open-source and licensed. When the application is deployed, a variety of users perform a variety of actions on the application.

*“Test automation is the process of automating well-known or predictive actions of users to make sure the application behaves as expected.”*

***What Is a Test Automation Framework?***

Before we go on to discuss test automation frameworks, let’s first understand what a framework is. In general, a framework is a combination of standards and rules which, when followed, can be used by an enterprise to make the best bang for their buck.

Similarly, *a test automation framework is a platform that is a combination of programs, compilers, features, tools, etc.* It provides an environment where you can execute automated test scripts.

In short, a test automation framework is a set of components that facilitate executing tests and comprehensive reporting of test results. The major components that implement a test automation framework successfully are equipment, testing tools, scripts, procedures, and most importantly, test automation engineers.

**Why Do We Need a Test Automation Framework**

In the modern era, the entire world is moving toward automation. With this, the need for test automation is rising. Proper planning and execution of test automation frameworks have a lot of perks to offer.

***1. Optimization of Resources***

A test framework helps in the optimization of resources. It does this by facilitating the use of different resources according to organizational needs.

For instance, to achieve established goals, test automation frameworks provide a set of processes. These processes have to match resources to requirements. The higher the flexibility of adoption, the better your resource optimization will be.

***2. Increased Volume of Testing***

Test automation frameworks increase the volume of testing. For instance, new mobile devices emerge every other day. It’s impossible to perform manual testing on all. Even if a firm managed to do so, it would take forever. But automated testing enables testers to run tests on thousands of mobile devices at the same time.

***3. Simultaneous Testing***

Test automation frameworks enable simultaneous testing of different types of devices. When the test scripts are automated, all testers need to do is run them on different devices. Since the parameters are same, testers can quickly generate comparative test reports.

***4. Enhanced Speed and Reliability***

Writing and running tests can be time-consuming for any software company. Test automation frameworks reduce the time to carry out these activities. How? Suppose you’re testing the logout functionality of a website. If there are multiple testing scenarios, for each scenario you have to manually test whether the log out feature is working properly. But if you’re using a framework, you can simultaneously run all the scenarios and get the test results in very little time.

Moreover, automated testing is more reliable due to the use of automated tools. This reduces the chances of making mistakes.

***5. More Output in Less Time***

Test automation reduces challenges in synchronization, local configuration, error management, and report generation. An automation script minimizes the time taken to prepare and run tests. With increased efficiency and speed, a firm can gain more output in less time.

***6. Fixing Bugs at an Early Stage***

A test automation framework helps in fixing bugs at an early stage. You don’t need much manpower to carry it out for you, which means the working hours and expenses involved are also reduced. A test automation engineer can write scripts and automate tests.

By using the right test automation frameworks, an organization can implement the concept of shift-left testing. That refers to the idea that you should move testing to as early in the software development lifecycle as possible. The earlier you can get is actually creating automated testing before the writing the production code. That’s exactly the modus operandi of techniques such as TDD (test-driven development) and BDD (behavior-driven development.)

***7. Remote Testing***

With a test automation framework, it’s not necessary to stay at the office premises 24/7. For instance, you can start running a test before leaving. When you come back after a few hours, the test results will be ready. Moreover, you don’t need to buy a lot of devices since you can test remotely.

***8. Reusable Automation Code***

You can reuse test automation scripts in a different application. Suppose the testers of your organization wrote some scripts for testing the login functionality. You can use the same script for another application that has a login functionality.

***9. Increased ROI***

The initial investment involved in test automation frameworks is off-putting for many. But the long-term return on investment is high. As discussed earlier, a test automation framework saves time and facilitates speedy delivery. It also reduces the need for more staff. For instance, a company doesn’t have to hire multiple testers if the testing framework is automated. A test automation engineer can carry out most of the tasks like configuring the framework or running the scripts.

***10. Continuous Testing***

The importance of continuous integration and continuous delivery/deployment can’t be overstated. Having a fully automated software pipeline is the surest way to ensure your code reaches production as fast as possible. However, it’s no use to ship broken code super-fast. That’s why an essential piece of the CI/CD puzzle is continuous testing. What is continuous testing?

In a nutshell, it’s the practice of running your suite of automated tests continuously. Test automation frameworks are key in achieving continuous testing, since they enable not only the creation of the tests, but also their automatic execution.

**Different Types of Test Automation Frameworks**

Now that you understand what a test automation framework is and what its components, let’s look at the different types of frameworks out there. Automated testing covers a range of test frameworks. The most common types:

1. Linear Automation Framework
2. Modular Based Testing Framework
3. Library Architecture Testing Framework
4. Keyword-Driven Framework
5. Data-Driven Framework
6. Hybrid Testing Framework

In our case, we’re using Hybrid Testing Framework because a hybrid test framework mitigates the weaknesses of different test frameworks. It provides flexibility by combining parts of different frameworks to harness the advantages. Hence, the efficiency of testing also improves.

**Introduction to Selenium**

***What is Selenium?***

Selenium refers to a suite of tools that are widely used in the testing community when it comes to cross-browser testing. Selenium cannot automate desktop applications; it can only be used in browsers. It is considered to be one of the most preferred tool suites for automation testing of web applications as it provides support for popular web browsers which makes it very powerful. It supports a number of browsers (Google Chrome 12+, Internet Explorer 7, 8,9,10, Safari 5.1+, Opera 11.5, and Firefox 3+) and operating systems (Windows, Mac, Linux/Unix).

Selenium also provides compatibility with different programming languages – C#, Java, JavaScript, Ruby, Python, and PHP. Testers can choose which language to design test cases in, thus making Selenium highly favorable for its flexibility.

**Selenium Components**

The Selenium test suite comprises four main components:-

* **Selenium IDE**

Selenium IDE (Integrated Development Environment) is primarily a record/run tool. It is an Add-on or an extension available for both Firefox and Chrome that generates tests quickly through its functionality of record and playback. You don’t need to learn any test scripting language for authoring any functional tests.

* **Selenium RC**

In the case of working with Selenium RC (Remote Control), one must have good knowledge of at least one programming language. This tool allows you to develop responsive design tests in any scripting language of your choice. Server and client libraries are the two main components of Selenium RC. Its architecture is complex and it has its limitations.

* **Selenium Webdriver**

Selenium WebDriver is an enhanced version of Selenium RC. It was introduced in the market to overcome the limitation faced in Selenium RC. Though it is an advanced version of RC, its architecture is completely different from that of RC. Just like Selenium RC, Selenium WebDriver too supports multiple programming platforms to provide wider flexibility and requires knowing any one programming language.

* **Selenium Grid**

Selenium Grid is a tool that is used for concurrent execution of test cases on different browsers, machines, and operating systems simultaneously. This tool makes Cross-browser compatibility testing very easy. There are two versions of the Selenium Grid – the older version is known as Grid 1 and the recent version is known as Grid 2.

***What is Selenium WebDriver?***

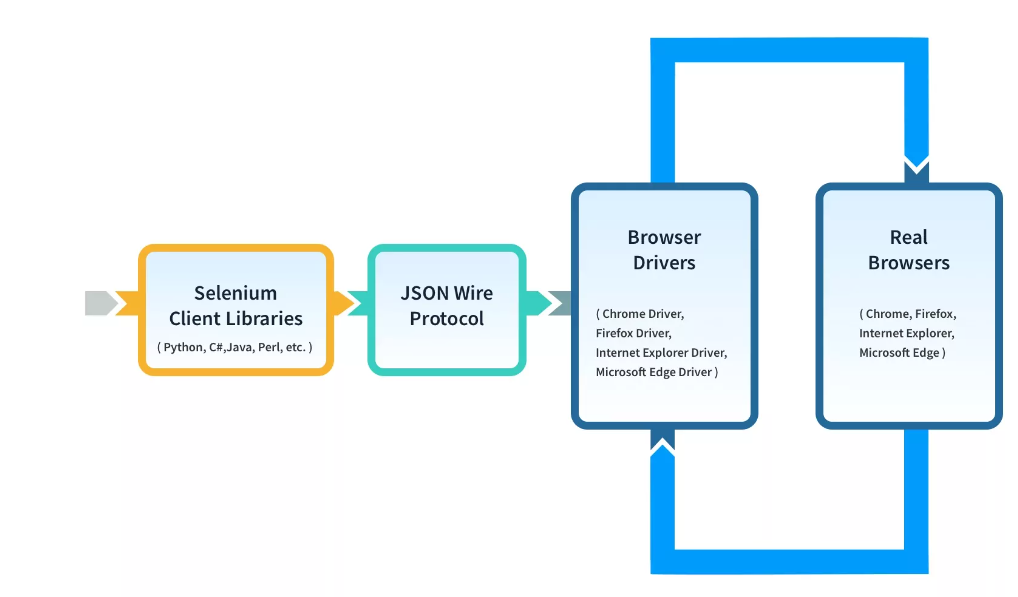
Selenium WebDriver is a web framework that permits you to execute cross-browser tests. This tool is used for automating web-based application testing to verify that it performs expectedly.

Selenium WebDriver allows you to choose a programming language to create test scripts. As discussed earlier, it is an advancement over Selenium RC to overcome a few limitations. Selenium WebDriver is not capable of handling window components, but this drawback can be overcome by using tools like Sikuli, Auto IT, etc.

**Selenium WebDriver Framework Architecture**

*WebDriver Architecture is made up of four major components:*

1. Selenium Client library
2. JSON wire protocol over HTTP
3. Browser Drivers
4. Browsers



**Selenium Client Libraries/Language Bindings**

Selenium provides support to multiple libraries such as Ruby, Python, Java, etc. as language bindings have been developed by Selenium developers to provide compatibility for multiple languages. For instance, if you want to use the browser driver in Python, use the Python Bindings. You can download all the supported language bindings of your choice from the official site of Selenium.

**JSON Wire Protocol**

JSON is an acronym for JavaScript Object Notation. It is an open standard that provides a transport mechanism for transferring data between client and server on the web. It provides support for various data structures like arrays and objects which makes it easier to read and write data from JSON.

JSON serves as a REST (Representational State Transfer) API that exchanges information between HTTP servers.

**Browser Drivers**

Selenium provides drivers specific to each browser and without revealing the internal logic of browser functionality, the browser driver interacts with the respective browser by establishing a secure connection. These browser drivers are also specific to the language which is used for test case automation like C#, Python, Java, etc.

You can download the browser driver of your choice as per your language requirements. When a test script is executed with the help of WebDriver, the following tasks are performed in the background:

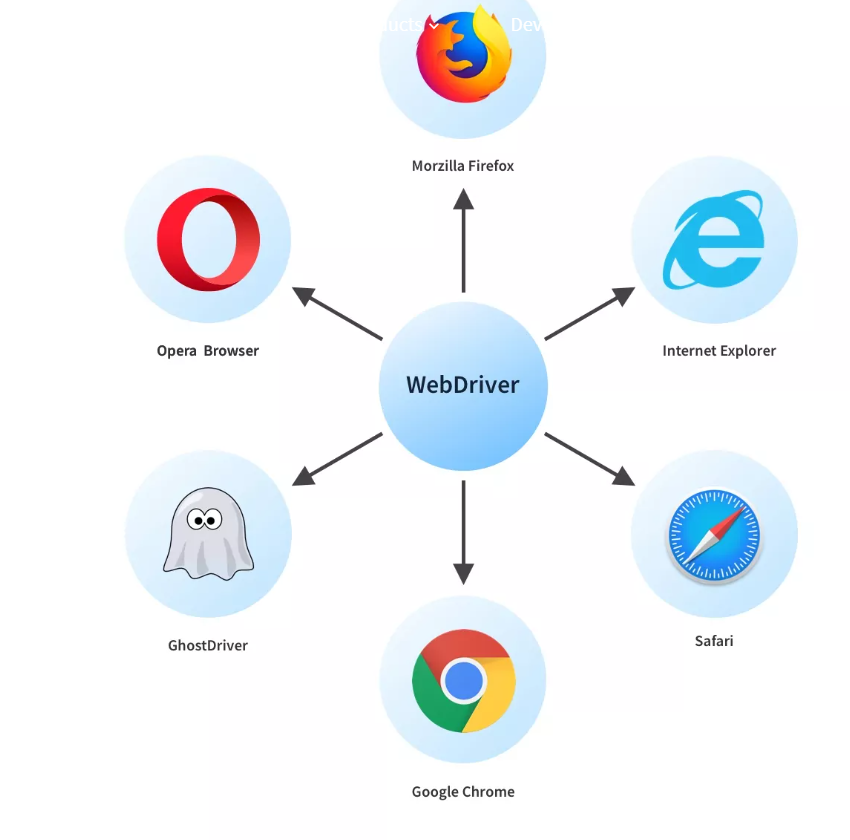
* An HTTP request is generated and it is delivered to the browser driver for every Selenium Command
* The HTTP request is received by the driver through an HTTP server
* All the steps/instructions to be executed on the browser is decided by an HTTP server
* The HTTP server then receives the execution status and in turn sends it back to the automation scripts

**Browsers**

As discussed earlier, Selenium provides support for multiple browsers like Chrome, Firefox, Safari, Internet Explorer etc.

**Benefits of Selenium WebDriver**

* It is one of the most popular Open-Source tools and is easy to get started with for testing web-based applications. It also allows you to perform cross browser compatibility testing.
* Supports multiple operating systems like Windows, Mac, Linux, UNIX, etc.
* It provides compatibility with a range of languages including Python, Java, Perl, Ruby, etc.
* Provides support for modern browsers like Chrome, Firefox, Opera, Safari, and Internet Explorer.
* Selenium WebDriver completes the execution of test scripts faster when compared to other tools
* More Concise API (Application Programming interface) than Selenium RC’s
* It also provides compatibility with iPhoneDriver, HtmlUnitDriver, and AndroidDriver



**Limitations of WebDriver**

* **No support for desktop applications** – Selenium does not support testing for desktop applications.
* **Expertise** – Selenium requires expertise of your team — and resources to manage.
* **Maintenance and Scalability** – Selenium is a maintenance-heavy framework — and is difficult to scale as one grows.
* **Open Source Forums** – Since Selenium is open source software, one has to rely on community forums to get your technical issues resolved.
* **No support for REST and SOAP Platforms** – We can’t perform automation tests on web services like SOAP or REST using Selenium.
* **No Reporting capability** – Selenium does not have any inbuilt reporting capability, one has to rely on plug-ins like JUnit and TestNG for test reports.
* **Image Testing** – It is not possible to perform testing on images. One needs to integrate Selenium with Sikuli for image testing.