**Task - 15 (GUVI-JAT18WD)**

[Q.No:1](http://q.no:1/)

**Selenium Introduction:**

Selenium Introduction – It is an open source (free) automated testing suite to test web applications. It supports different platforms and browsers. It has gained a lot of popularity in terms of web-based automated testing and giving a great competition to the famous commercial tool HP QTP (Quick Test Professional) AKA HP UFT (Unified Functional Testing).

Selenium is a set of different software tools. Each tool has a different approach in supporting web based automation testing.

It has four components namely,  
i. Selenium IDE (Integrated Development Environment)  
ii. Selenium RC (Remote Control)  
iii. Selenium WebDriver  
iv. Selenium Grid

QTP is a famous automation tool and it was originally developed by Mercury Interactive before HP acquired it. Selenium is used as an antidote in the treatment of mercury intoxication. Jason Huggins (An engineer at ThoughtWorks – the one who introduced Selenium tool) suggested the name of this automation tool as Selenium.

**What is Selenium IDE?**

Selenium IDE (Integrated Development Environment) is a Firefox plugin. It is the simplest framework in the Selenium Suite. It allows us to record and playback the scripts. Even though we can create scripts using Selenium IDE, we need to use Selenium RC or Selenium WebDriver to write more advanced and robust test cases.

**Operation System Support** – Windows, Mac OS, Linux

**Browser Support** – Mozilla Firefox

**What is Selenium WebDriver?**

Selenium WebDriver AKA Selenium 2 is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It controls the browser by directly communicating with it. Selenium WebDriver supports Java, C#, PHP, Python, Perl, Ruby.

**Operation System Support** – Windows, Mac OS, Linux, Solaris  
**Browser Support** – Mozilla Firefox, Internet Explorer, Google Chrome 12.0.712.0 and above, Safari, Opera 11.5 and above, Android, iOS, HtmlUnit 2.9 and above

**What is Selenium Grid?**

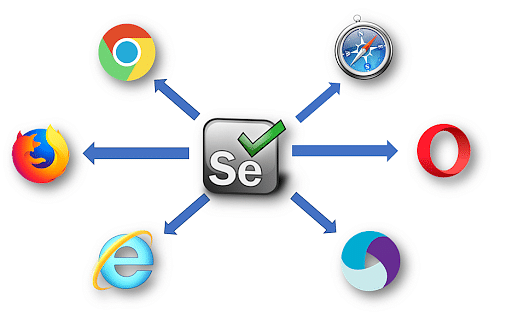
Selenium Grid is a tool used together with Selenium RC to run tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems.

**The differences between selenium IDE, RC & Web driver are listed below :**

| **Functionalities** | **Selenium IDE** | **Selenium RC** | **Selenium Webdriver** |
| --- | --- | --- | --- |
| Record and playback | It has the record and playback feature. | It does not have a record and playback. | It does not have a record and playback. |
| Server | It requires no server to start execution of test cases. | It requires the server to start execution of test cases. | It requires no server to start execution of test cases. |
| Browser | It can be used for testing only in Firefox. | It can be used for testing in the majority of browsers. | It can be used for testing in the majority of browsers including in headless mode. |
| Object Oriented | It is based on Selenese which is a procedural language. | It can be partially used for object oriented programming. | It is majorly used for object oriented programming. |
| Dynamic Locators | Elements cannot be identified. | Elements cannot be identified. | Elements can be identified. |
| Alerts | Cannot handle alerts. | Cannot handle alerts. | Can handle alerts |
| Mouse Actions | Cannot handle mouse actions. | Cannot handle mouse actions. | Can handle mouse actions. |
| Dropdown | Cannot handle dropdown. | Cannot handle dropdown. | Can handle dropdown. |
| iPhone/Android | Cannot perform iPhone/Android testing. | Cannot perform iPhone/Android testing. | Can perform iPhone/Android testing with the help of Android Driver, iPhone Driver. |
| Listener | Does not have a Listener. | Does not have a Listener. | Have Listeners. |
| Performance | Fast [comes as a Firefox plugin]. It interacts with the browser directly. | It does not interact with the browser directly. Hence on a slower side compared to webdriver. | Fast as it interacts directly with the browser. |
| Architecture | Derived from Javascript. | Derived from Javascript. | Not derived from Javascript. |
| Usage | UI interface available to create scripts. | Standalone Jars available to execute test cases in the browser. | Contains API and supported by languages like Java, Python, and Ruby and so on. |
| Xpath | Only has absolute xpath. | Only has absolute xpath. | Has both absolute and relative xpath. |
| Navigation | Cannot handle navigation. | Cannot handle navigation. | Can handle navigation. |

[Q.No:3](http://q.no:3/)

**What is Selenium?**

Selenium is an open-source, automated testing tool used to test web applications across various browsers. Selenium can only test web applications, unfortunately, so desktop and mobile apps can’t be tested. However, other tools like Appium and HP’s QTP can be used to test software and mobile applications.

After looking into what Selenium is, let us learn the popularity of Selenium.

**What makes Selenium Such a Widely Used Testing Tool?**

1. Selenium is easy to use since it’s primarily developed in JavaScript
2. Selenium can test web applications against various browsers like Firefox, Chrome, Opera, and Safari
3. Tests can be coded in several programming languages like Java, Python, Perl, PHP, and Ruby
4. Selenium is platform-independent, meaning it can deploy on Windows, Linux, and Macintosh
5. Selenium can be integrated with tools like JUnit and TestNG for test management

Now that we have learned what is Selenium, let us look into the Selenium suite of tools.

**Importance of Testing in Selenium**

Manual testing can be time-consuming and prone to human errors. [Selenium Automation](https://www.browserstack.com/guide/selenium-framework) allows tests to be executed quickly and accurately, reducing the likelihood of human mistakes and ensuring consistent test results.

Selenium allows developers and testers to automate the testing of web applications across different browsers and platforms.

1. **Language Support:** Selenium allows you to [create test scripts](https://www.browserstack.com/guide/run-selenium-test-script) in different languages like [Ruby](https://www.browserstack.com/guide/selenium-ruby-tutorial), [Java](https://www.browserstack.com/guide/selenium-with-java-for-automated-test), [PHP](https://www.browserstack.com/guide/selenium-and-php-tutorial), Perl, [Python](https://www.browserstack.com/guide/automate-with-selenium-python), [JavaScript](https://www.browserstack.com/guide/automation-using-selenium-javascript), and [C#](https://www.browserstack.com/guide/selenium-with-c-sharp-for-automated-test), among others.
2. **Browser Support:** Selenium enables you to test your website on different browsers such as [Google Chrome](https://www.browserstack.com/guide/run-selenium-tests-using-selenium-chromedriver), [Mozilla Firefox](https://www.browserstack.com/guide/run-selenium-tests-using-firefox-driver), [Microsoft Edge](https://www.browserstack.com/guide/launch-edge-browser-in-selenium), [Safari](https://www.browserstack.com/guide/run-selenium-tests-on-safari-using-safaridriver), [Internet Explorer (IE)](https://www.browserstack.com/guide/run-selenium-tests-using-ie-driver), etc.
3. **Scalability:** Automated testing with Selenium can easily scale to cover a wide range of [test cases](https://www.browserstack.com/guide/how-to-create-selenium-test-cases), scenarios, and user interactions. This scalability ensures [maximum test coverage](https://www.browserstack.com/guide/how-to-ensure-test-coverage) of the application’s functionality.
4. **Reusable Test Scripts:** Selenium allows testers to create reusable test scripts that can be used across different test cases and projects. This reusability saves time and effort in test script creation and maintenance.
5. [**Parallel Testing**](https://www.browserstack.com/guide/what-is-parallel-testing)**:** Selenium supports parallel test execution, allowing multiple tests to run concurrently. This helps reduce the overall testing time, making the development process more efficient.
6. **Documentation and Reporting:** Selenium provides detailed test execution logs and reports, making it easier to track test results and identify areas that require attention.
7. **User Experience Testing:** Selenium can simulate user interactions and behavior, allowing testers to assess the user experience and ensure that the application is intuitive and user-friendly.
8. **Continuous Integration and Continuous Deployment (CI/CD):** Selenium can be integrated into CI/CD pipelines to automate the testing of each code change. This integration helps identify and address issues earlier in the development cycle, allowing for faster and more reliable releases.

[Q.No:4](http://q.no:4/)

Selenium WebDriver supports various browser drivers to automate web browsers. Here's a list of commonly used browser drivers:

1. **ChromeDriver**: Used to automate Google Chrome browser.
2. **GeckoDriver**: Used to automate Mozilla Firefox browser.
3. **WebDriver for Microsoft Edge**: Used to automate Microsoft Edge browser.
4. **SafariDriver**: Used to automate Safari browser (though it's only supported on macOS).
5. **OperaDriver**: Used to automate Opera browser.
6. **InternetExplorerDriver**: Used to automate Internet Explorer browser (though it's mostly deprecated due to Internet Explorer's decreasing usage).

Each of these browser drivers needs to be downloaded and set up according to the browser version you are using and the programming language you're working with for Selenium automation. Make sure to use the compatible version of the browser driver with the browser you intend to automate.

[Q.No:5](http://q.no:5/)

**The steps to create a simple WebDriver script using Java with Selenium WebDriver:**

1. **Set Up Your Development Environment:** Ensure you have Java Development Kit (JDK) installed on your system. You'll also need an Integrated Development Environment (IDE) like Eclipse or IntelliJ IDEA.
2. **Download Selenium WebDriver:** Download the Selenium WebDriver Java bindings from the Selenium website or include them as dependencies in your project using a build automation tool like Maven or Gradle.
3. **Create a New Java Project:** Open your IDE and create a new Java project.
4. **Configure Selenium WebDriver:** Include the WebDriver binaries for the browser you want to automate (e.g., ChromeDriver for Chrome) in your project. Make sure they are executable and accessible from your code.
5. **Write Your WebDriver Script:** Write your WebDriver script in Java, including importing necessary packages and classes from Selenium WebDriver.
6. **Instantiate WebDriver:** Create an instance of the WebDriver interface corresponding to the browser you want to automate (e.g., ChromeDriver).
7. **Navigate to a Web Page:** Use the WebDriver instance to navigate to a specific URL.
8. **Interact with Web Elements:** Find web elements using locators like ID, class name, XPath, etc., and perform actions like clicking buttons, entering text, etc.
9. **Close the Browser:** Once you're done with the automation, close the browser window.

Here's a simple example using Java and Selenium WebDriver to automate opening Google's homepage, searching for a term, and clicking the search button:

import [org.openqa.selenium.By](http://org.openqa.selenium.by/" \t "_blank);

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import [org.openqa.selenium.chrome](http://org.openqa.selenium.chrome/" \t "_blank).ChromeDriver;

public class SimpleWebDriverScript

{

public static void main(String[] args)

{

// Step 6: Instantiate WebDriver (Chrome in this example)

WebDriver driver = new ChromeDriver();

// Step 7: Navigate to Google homepage driver.get("[https://www.google.com](https://www.google.com/)");

// Step 8: Find the search box element and enter a search term driver.findElement([By.name](http://by.name/)("q")).sendKeys("Selenium WebDriver");

// Press Enter to perform the search driver.findElement([By.name](http://by.name/)("q")).sendKeys(Keys.RETURN);

// Alternatively, find the search button and click it // driver.findElement([By.name](http://by.name/)("btnK")).click();

// Step 9: Close the browser window // driver.close(); // Close current tab driver.quit(); // Quit the entire browser

}

}

Make sure you have the appropriate WebDriver executable for your browser installed and its location added to your system's PATH environment variable. For Chrome, you'll need ChromeDriver, for Firefox, you'll need GeckoDriver, etc. You can download them from the respective browser driver websites.

Adjust the code according to your specific requirements and the structure of the website you're automating.

**Note:**

**The Example Program is available in below link (**[Q.No](http://q.no/" \t "_blank)**: 2 & 5).**

<https://github.com/DhivyaPrabhu/SeleniumMavenProject.git>