**TASK 15**

**Q1. DIFFERENCE BETWEEN SELENIUM IDE, SELENIUM WEBDRIVER AND SELENIUM GRID**

Selenium IDE, Selenium WebDriver, and Selenium Grid are three components of the Selenium suite, each serving different purposes in web automation testing. Here's a breakdown of the differences between them:

**Selenium IDE:**

1. **Definition**: Selenium IDE (Integrated Development Environment) is a record and playback tool for creating automated tests in the browser.
2. **Features**:
   * Record and Playback: Allows testers to record their interactions with the web application and play them back as automated test cases.
   * UI-based Test Creation: Provides a simple interface for creating test cases without writing code.
   * Export Test Cases: Test cases recorded in Selenium IDE can be exported to various programming languages such as Java, Python, etc., for further customization and execution.
3. **Use Cases**:
   * Quick Test Creation: Selenium IDE is useful for creating simple, quick tests or prototypes.
   * Learning and Experimentation: It can be used by beginners to learn Selenium and experiment with automated testing.

**Selenium WebDriver:**

1. **Definition**: Selenium WebDriver is a programming interface that allows testers to write code to automate web browser interactions.
2. **Features**:
   * Programmatically Control Browser: Provides APIs in various programming languages (Java, Python, C#, etc.) for interacting with web browsers programmatically.
   * Cross-Browser Testing: Supports automation across different browsers and platforms.
   * Dynamic Test Scenarios: Allows testers to create dynamic and complex test scenarios by writing code.
3. **Use Cases**:
   * Comprehensive Test Automation: Selenium WebDriver is suitable for automating complex web applications with dynamic content and interactions.
   * Integration with Test Frameworks: It is often integrated with testing frameworks such as JUnit, TestNG, NUnit, etc., for structured test automation.

**Selenium Grid:**

1. **Definition**: Selenium Grid is a tool for distributing test execution across multiple machines or browsers in parallel.
2. **Features**:
   * Parallel Test Execution: Allows tests to be run concurrently across multiple browsers, operating systems, and machines.
   * Scalability: Provides scalability by distributing test execution load across a grid of nodes.
   * Cross-Browser Testing: Supports running tests on different browsers and versions simultaneously.
3. **Use Cases**:
   * Cross-Browser Testing: Selenium Grid is used for testing web applications across multiple browsers and platforms simultaneously.
   * Scalable Test Execution: It is suitable for running large test suites in parallel to reduce test execution time.

**Key Differences:**

* **Purpose**: Selenium IDE is primarily for record and playback of test cases, Selenium WebDriver is for writing code-based test automation scripts, and Selenium Grid is for distributing tests across multiple environments.
* **Level of Automation**: Selenium IDE offers low-level automation with record and playback, while Selenium WebDriver provides high-level automation with programmatically controlled browser interactions.
* **Scalability**: Selenium Grid is designed for distributed test execution and scalability across multiple machines and browsers, while Selenium WebDriver and IDE focus on single-machine automation.

**Q3. 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. It provides a suite of tools and libraries that allow testers and developers to automate the testing of web applications, ensuring that they function correctly across different browsers, platforms, and scenarios. Selenium is widely used in the field of automation testing to streamline and improve the testing process.

**Selenium is useful in automation testing for the advantages listed below:-**

**1.Language and Framework Support**

Selenium supports various languages, including Java, Perl, Python, C#, Ruby, Groovy, JavaScript, and more. It has its own script, but it is not limited by that language. It can work with various languages – whatever the developers/testers are comfortable with.

**2. Open Source Availability**

Selenium is a publicly accessible automation framework and is free, with no upfront costs. It is continuously helping developers and software engineers in automating the web browser features and functionalities. Selenium being open source also helps us to customize the code for better code management and enhance the functionality of predefined functions and classes. Selenium has become the most reliable web automation tool because of the ease of generating test scripts to validate functionality.

**3. Multi-Browser Support**

“One Selenium script for all browsers” is what the Selenium community has been working on and improvising every day. As per [StatCounter](http://gs.statcounter.com/browser-market-share" \t "_blank), Chrome, Firefox, Safari, Internet Explorer, Opera, and Edge browsers are the most used browsers worldwide and Selenium script is compatible with all the mentioned browsers. There is no need to rewrite scripts for every browser, just one script for all browsers.

**4. Support Across Various Operating Systems**

Different people use different operating systems and it is necessary that the automation tool supports all of them. Selenium is yet a highly portable tool that supports and can work across different operating systems like Windows, Linux, Mac OS, UNIX, etc. Selenium test suites can be created over any platform like Windows and can execute the same test suite on another platform.

**5. Ease Of Implementation**

* Selenium automation framework is very easy-to-use tool. Selenium provides a user-friendly interface that helps create and execute test scripts easily and effectively. Its open-source features help users to script their own extensions that make them easy to develop, customize actions, and even manipulate at an advanced level.
* Tests run directly across browsers; users can watch while the tests are being executed. Additionally, Selenium’s reporting capabilities are one of the reasons for choosing it, as it allows testers to extract the results and take follow-up actions

**6. Reusability and Integrations**

Selenium Test Automation framework uses scripts that can be tested directly across multiple browsers. Concurrently, it is possible to execute multiple tests with Selenium, as it covers almost all aspects of functional testing by implementing add-on tools that broaden the scope of testing.

**7. Flexibility**

Test management is what which is very important in testing lifecycle. It becomes easier and more efficient with Selenium features like regrouping and refactoring of test cases. This helps developers and testers in quick changes to the code, reducing duplication, minimizing complications and improving maintainability. These features make Selenium more flexible and usable as compared to other automation testing tools and hence helps Selenium to keep an edge.

**8. Parallel Test Execution and Faster Go-to-Market**

The main aim of automated testing is to save time and efforts. With the help of Selenium Grid, we can execute multiple tests in parallel, hence reducing the test execution time. With the help of cloud-grids for cross-browser testing across as many as hundreds of browsers in parallel using Selenium saves more time.

**9. Less Hardware Usage**

On comparing Selenium with other vendor focused automation tools like QTP, UFT, SilkTest, The Selenium requires less hardware as compared to other testing tools.

**10. Easy to Learn and Use**

Selenium scripts are not something like writing hundred-page complex algorithm. Writing Selenium scripts is not more than writing a few pieces of codes to automate functionalities of the website. Also, documentation on the [Selenium website](http://www.seleniumhq.org/) is very helpful for developer and testers to start with Selenium automation testing. With the radically growing community, Selenium tutorials, testing, and development support is just a Google search away.

**11. Constant Updates**

As Selenium is supported by a community and we all know that an active community doesn’t like to stay stagnant, the Selenium community is also constantly releasing constant updates and upgrades. The best part about having a community is that these upgrades are readily available and easy to understand. This makes Selenium resourceful as compared to other tools and cost-effective as well.

**Q.4 What are all Browser driver used in Selenium?**

Some of the commonly used browser drivers in Selenium:

**Chrome Driver:**Used for interacting with the Google Chrome browser. You need to download the Chrome Driver executable and set its path in your Selenium script.

**Gecko Driver (Firefox):** Used for interacting with the Mozilla Firefox browser. It is required for automating Firefox with Selenium. Similar to Chrome Driver, you need to download Gecko Driver and set its path in your script.

**Microsoft WebDriver (Edge):** Used for interacting with the Microsoft Edge browser. It is necessary for automating Edge with Selenium. The Edge Driver is available as a separate download.

**Internet Explorer Driver (IE):** Used for interacting with the Internet Explorer browser. It is required for automating Internet Explorer with Selenium. Note that Internet Explorer is being phased out, and Microsoft recommends using Microsoft Edge instead.

**Safari Driver (Safari):** Used for interacting with the Safari browser. Safari Driver comes bundled with Safari, and you need to enable the “Remote Automation” option in Safari’s Develop menu to use it.

**Opera Driver (Opera):** Used for interacting with the Opera browser. Similar to Chrome and Firefox, you need to download the Opera Driver executable and set its path in your script.

**Q.5 Steps to create a web driver script?**

**package** task15;

**import** java.time.Duration;

**import** org.apache.commons.lang3.time.DurationFormatUtils;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** ChromeSelenium {

**public** **static** **void** main(String[] args) {

Step 1. Launching the browser

WebDriver driver = **new** ChromeDriver();

Step 2. Get the URL link to launch in the browser

driver.get("https://www.google.com/");

Step 3. Maximize the screen resolution of the browser

driver.manage().window().maximize();

Step 4.Timeout to check the execution process

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(20));

Step 5. Locating an value from the ID or class or from Xpath of the value selected/ here ID value is selected from the search field and its clicked to enter the input to it

driver.findElement(By.*id*("APjFqb")).click();

Step 6. Input to the search field is provided using sendkeys

driver.findElement(By.*id*("APjFqb")).sendKeys("selenium web driver");

Step 7. Input is entered to the field(search) by submit

driver.findElement(By.*id*("APjFqb")).submit();

}

}