UNIT IV – TESTING AND QUALITY IN SOFTWARE [9 hours]

Types of Testing: Unit, Integration, System, Automated Testing using JUnit or PyTest, Introduction to Selenium (UI testing basics), Code Quality Tools: SonarLint/SonarQube basics , Importance of Testing in Real Projects

---

INTRODUCTION TO SELENIUM

Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite.It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages. Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.

Selenium supports a variety of programming languages through the use of drivers specific to each Language. Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby. Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

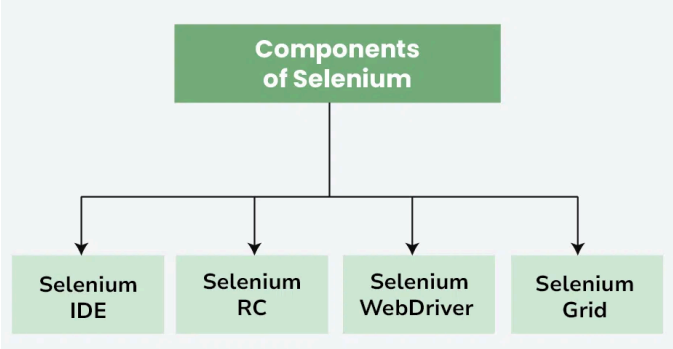
Selenium can be used to automate functional tests and can be integrated with automation test tools such as Maven, Jenkins, & Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports.

Selenium Features

- Selenium is an open source and portable Web testing Framework.
- Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
- It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
- Selenium supports various operating systems, browsers and programming languages. Following is the list:
  - Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
  - Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
  - Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It also supports parallel test execution which reduces time and increases the efficiency of tests.

- Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
- Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
- Selenium requires fewer resources as compared to other automation test tools.
- WebDriver API has been indulged in selenium which is one of the most important modifications done to selenium.
- Selenium web driver does not require server installation, test scripts interact directly with the browser.
- Selenium commands are categorized in terms of different classes which make it easier to understand and implement.

Components of Selenium:



1. Selenium IDE

Selenium IDE serves as an innovative toolkit for web testing, allowing users to record interactions with web applications. Selenium-IDE was initially created by " Shinya Kasatani " in 2006. Selenium IDE also helps to simplify the testing process. It is a friendly space for testers and developers to team up. This helps everyone quickly share important testing information and results, making things work better and feel accomplished.

- **Record**: With Selenium IDE, users can record how they use a web application.
- **Playback**: Selenium IDE automatically repeats what was recorded earlier.
- **Browser Check**: Selenium IDE works on various browsers for testing.
- **Check Elements:** Users can easily look at different parts of a webpage and set up how to work with them.
- **Spotting Errors**: Selenium IDE helps users find and fix issues in their automated tests, one step at a time.
- **Exporting Tests**: We can save tests created in Selenium IDE in different programming languages (like Java, Python, or C#). This lets you use them with other Selenium tools.

2. Selenium RC (Remote control)

Selenium Remote Control (RC) was one of the earliest Selenium tools, preceding WebDriver. It allowed testers to write automated web application tests in various programming languages like Java, C#, Python, etc. The key feature of Selenium RC was its ability to interact with web browsers using a server, which acted as an intermediary between the testing code and the browser.

WebDriver is the better choice over Selenium RC for several reasons are follows:

- **Improved API**: WebDriver offers a more straightforward and intuitive API compared to Selenium RC, making it easier for developers and testers to write and maintain automated tests.
- **Better Performance**: WebDriver interacts directly with the browser, bypassing the need for an intermediary server like Selenium RC, which leads to faster test execution and improved performance.
- **Support for Modern Web Technologies**: WebDriver has better support for modern web technologies such as HTML5, CSS3, and JavaScript frameworks, ensuring compatibility with the latest web applications.

3. Selenium Web Driver

Selenium WebDriver is a robust open-source framework for automating web browsers , primarily aimed at easing the testing and verification of web applications. As an important part of the Selenium suite, WebDriver offers a programming interface to interact with web browsers, allowing developers and testers to automate browser actions seamlessly.

- **Direct Communication with Browsers:** Unlike Selenium RC, WebDriver interacts directly with the browser's native support for automation, leading to more stable and reliable testing.
- **Support for Parallel Execution:** WebDriver allows for parallel test execution, enabling faster test cycles and efficient utilization of resources.
- **Rich Set of APIs:** WebDriver provides a comprehensive set of APIs for navigating through web pages, interacting with web elements , managing windows, handling alerts , and etc.

4. Selenium GRID

Selenium Grid is a server that allows tests to use web browser instances running on remote machines. With Selenium Grid, one server acts as the hub. Tests contact the hub to obtain access to browser instances.

- Selenium Grid allows running tests in parallel on multiple machines and managing different browser versions.

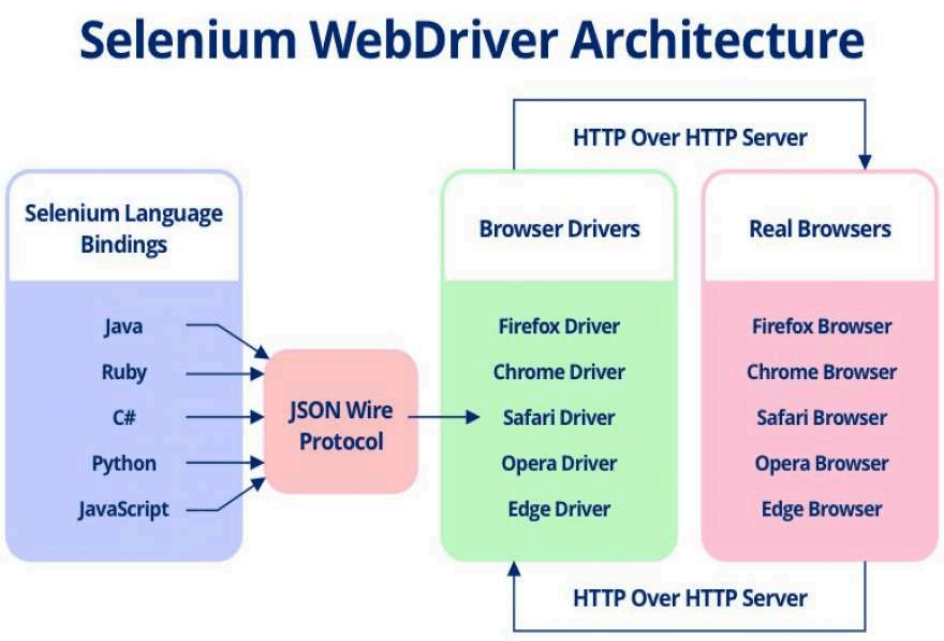
- The ability to run tests on remote browser instances is useful to spread the load of testing across several machines.
- Run tests in browsers running on different platforms or operating systems.

UI TESTING BASED ON SELENIUM

User Interface (UI) testing verifies an application's visual elements—buttons, menus, forms, and layout—to ensure they function correctly and meet design specifications. It simulates user interactions (clicks, input) to evaluate usability, responsiveness across devices, and overall user experience.

Selenium UI Testing Architecture

Selenium UI testing is performed using Selenium WebDriver, which executes tests on browsers installed on the system. Its architecture is as follows:



Selenium architecture contains four elements, as seen in the above image:

- **Selenium language bindings** – Selenium comes in different language bindings, also called client libraries. The testers need to download the same binding to write the test cases in Selenium.
- **JSON Wire Protocol** – This element helps communication between the language binding and browser drivers and is done using JSON over REST services.
- **Browser drivers** – The browser drivers help execute the automated test scripts on the actual browsers installed on the system. We need to download a browser driver specific to each browser installed on our system.

- **Real browsers** – These are the real browsers installed on the system available to download from their official websites like Chrome and Firefox.

The communication between browser drivers and real browsers is done through HTTP. The drivers send the instructions to the real browsers. The browser then executes the steps and returns the report to the driver. This is then shown to the tester. All this is done with the help of an HTTP server which is also seen in the image.

Locators and Their Types

Locators are those attributes of the elements that give it a unique representation. Due to this, they are the preferred choice in UI testing. Selenium can grasp these elements using locators and take required action upon them. In Selenium UI testing, a tester may use one of the following locators:

ID

The ID attribute is attached to an HTML element as "id" which should be unique for an element in the same DOM. If the web page has more than one element with a similar id, JS code generally returns the first element, which can create confusion in testing. Therefore, it is a practice to keep this unique, and for testers wishing to find a unique element, "id" fits the purpose.

Selenium Code for Locating an Element with ID

```
driver.findElement(By.id(""))
```

Class

The class attribute in HTML is similar to the ID attribute, but here, more than one element may have the same class name. The purpose of the class is to pool a certain set of elements with specific characteristics. For example, a class with the name "highlight" can be defined and used with all the elements that may need highlighting with different colors on the web page.

Selenium Code for Locating an Element with Classname

```
driver.findElement(By.className(""))
```

Name

The name attribute is attached to the element to reference it in programming languages. The usage of "name" differs according to the element it is used in. It is popularly used in the input tag, but there is no such standard.

Selenium Code for Locating an Element with Name

```
driver.findElement(By.name("name_attribute_value"));
```

Xpath

XPath stands for XML path language. It is used for navigating through the DOM tree with the help of expressions. For example, if there is an element "city" with multiple

child elements named “Paris”, then to select the first “Paris” element we can write it as: /city/Paris[1].

Selenium Code for Locating an Element with Xpath

driver.findElements(By.xpath(“xpath_expression”));

CSS Selector

Another method to locate an element using Selenium is by using CSS Selectors. Their working method is somewhat similar to XPath and is very efficient. It is generally used when there is not any direct relation to the element with ID, class, and name.

Selenium code for locating an element with XPath

driver.findElement(By.cssSelector(“CSS_Selector”))

Link Text

A link text, as its name suggests, is the text attached to the link on a web page. So when we want to write some text clicking on which will redirect us to another link that may or may not be on the same page, we make use of an anchor tag. The anchor tag contains the link, while the text mentioned in it is what a user will see on the web page.

```
<html>
    <head>
        <tite>Sample</tite>
    </head>
    <body>
        <a href=”http://www.google.com” >Click Here</a>
        <br>
        <a href=”http://www.yahoo.com” >Click Here</a>
    </body>
</html>
```

Selenium Code for Locating an Element with Linktext

driver.findElement(By.linkText(“”))

Selenium Code for Locating an Element with Partiallinktext

Element(By.partialLinkText(“”))

Example Login.html

```
<!DOCTYPE html>
<html>
<head>
<title>Login Page</title>
<script>
function validateLogin()
```

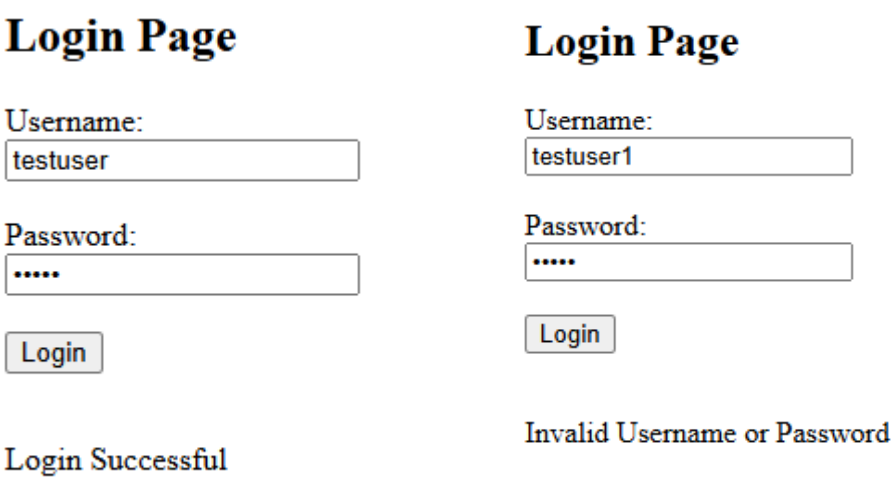
```
{
        var username = document.getElementById("username").value;
        var password = document.getElementById("password").value;
        if(username == "testuser" && password == "12345")
        {
                document.getElementById("message").innerHTML = "Login Successful";
        }
        else
        {
                document.getElementById("message").innerHTML = "Invalid Username or
        Password";
        }
        return false;
}
</script>
</head>
<body>
        <h2>Login Page</h2>
        <form onsubmit="return validateLogin()">
                <label>Username:</label><br>
                <input type="text" id="username" name="username"><br><br>
                <label>Password:</label><br>
                <input type="password" id="password" name="password"><br><br>
                <button id="loginBtn" type="submit">Login</button>
        </form>
        <br>
        <p id="message"></p>
</body>
</html>
```

Selenium code for testing:

LoginTest.java

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class LoginTest
{
```

```
    public static void main(String[] args) throws InterruptedException
    {
        // Step 1: Launch Chrome Browser
        WebDriver driver = new ChromeDriver();
        // Step 2: Open the login page (local HTML file)
        driver.get("D:/Welcome/login.html");
        // Step 3: Enter Username
        driver.findElement(By.id("username")).sendKeys("testuser");
        // Step 4: Enter Password
        driver.findElement(By.id("password")).sendKeys("12345");
        // Step 5: Click Login button
        driver.findElement(By.id("loginBtn")).click();
        // Step 6: Wait for message to appear
        Thread.sleep(1000);
        // Step 7: Read the result message
        String message = driver.findElement(By.id("message")).getText();
        System.out.println("Login Result: " + message);
        // Step 8: Verify result
        if(message.equals("Login Successful"))
        {
            System.out.println("Test Passed");
        }
        else
        {
            System.out.println("Test Failed");
        }
        // Step 9: Close browser
        driver.quit();
    }
}
```



Example 2: StudentForm.html

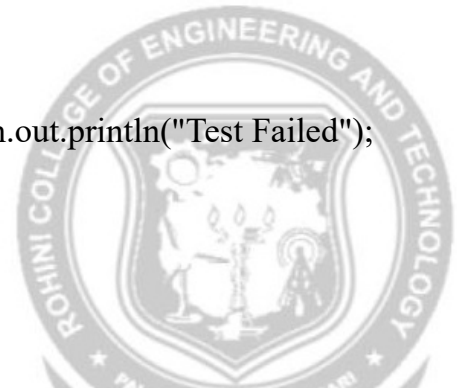
```
<html>
<body>
<h2>Student Form</h2>
<form onsubmit="showMessage(); return false;">
Name: <input type="text" id="name"><br><br>
Email: <input type="text" id="email"><br><br>
<button id="submitBtn" type="submit">Submit</button>
</form>
<p id="message"></p>
<script>
function showMessage()
{
document.getElementById("message").innerHTML="Form submitted successfully";
}
</script>
</body>
</html>
```

FormTest.java

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class FormTest
{
        public static void main(String[] args) throws InterruptedException
```

```
    {
        WebDriver driver = new ChromeDriver();
        driver.get("file:///Users/yourname/Documents/Welcome/form.html");
        driver.findElement(By.id("name")).sendKeys("Preethi");
        driver.findElement(By.id("email")).sendKeys("preethi@gmail.com");
        driver.findElement(By.id("submitBtn")).click();
        Thread.sleep(1000);
        String message = driver.findElement(By.id("message")).getText();
        System.out.println("Message: " + message);
        if(message.equals("Form submitted successfully"))
        {
            System.out.println("Test Passed");
        }
        else
        {
            System.out.println("Test Failed");
        }
        driver.quit();
    }
}
```

Output:



Student Form

Name: Preethi

Email: preethik.rcet@gmail.com

Submit

Form submitted successfully
