**SELENIUM WEB DRIVER**

**Selenium is open source Web based Automation Testing Tool**

|  |
| --- |
| **Syllabus:**   1. Complete Core java 2. Selenium Webdriver- Major focus (20 hours+ content) 3. Selenium Grid 4. Live projects on Selenium 5. Interview questions discussion with solutions 6. TestNG 7. Maven 8. ANT 9. Jenkins 10. Log4j 11. Real time challenges in developing framework and how to address them 12. Extent Test Execution reports 13. Cucumber 14. Data driven framework 15. Hybrid Framework 16. Page object Model Framework 17. DataBase testing with Selenium 18. Performance testing with Selenium 19. Mobile Automation basics only! 20. CLoud Automation with Sauce labs 21. AutoIT- Desktop Automation integration with Selenium 22. Bonus lecture- Bugzilla defect tool management. |

**What is used before SELENIUM comes into the Picture?**

QTP invented by HP.

It is PAID version. (Have to thousands of dollars to get a license)

**Supporting Browsers:**

Google Chrome, Mozilla Firefox, Internet Explorer, Safari

**Supporting Platforms:**

Windows, Mac OS, Linux

**Languages supported by Selenium:**

Java, C#, Python, JS, PHP, Ruby (Java + Selenium is the very popular.)

Selenium Ide – is to record and to playback. (Mostly not used)

Selenium Web Driver – is the popular one to perform all the regression testing which is web based.

**Architecture of Selenium Web Driver:**

Diagram, application

Description automatically generated

**CLIENT SERVER BROWSERS**

Code which we return 🡪 Converted to JSON 🡪 Chrome Driver, 🡪 Any browsers

Firefox Driver,

IE Driver

**How can we send ‘converted JSON’ to the Server (Browser Drivers)?**

Download the respective Browser driver from the internet.

Include the downloaded driver into the *selenium code* by giving the path of the driver file.

**Responsibility of Browser (Chrome etc.,) Driver:**

After getting the JSON, chrome driver is responsible to start the actual browser and perform all the operations.

And the response back from the browser and send it back to the client or selenium code.

**Setting Up a WebDriver using Java Maven Project:**

**Problem in using the Standalone drivers:**

**Problem:** If I want to execute the code written using Chrome driver in different browser. We should change ‘Chrome Driver’ to respective browser driver. Because methods present in the ‘Chrome Driver’ might not present in ‘Firefox Driver’.

**Solution:** To solve the above issue Selenium provide the ‘WebDriver’ interface. This ‘WebDriver’ interface will have all the **Common Methods** that are supported all the browsers each browser has its own implementation in their class.

For eg: chromeDriver.get(); firefoxDriver.get();

|  |
| --- |
| String browserName = “Chrome”;  if (browserName == “Chrome”) {  WebDriver chromeDriver = new ChromeDriver();  } |

**How the browser getting launched when we write the code in selenium?**

Selenium Code 🡺 Interpreted by **chromedriver.exe 🡺** Browser will launch

chromedriver.exe 🡪 provided by Chrome guys.

Note: version of chromedriver.exe ⬄ version of Chrome in your system.

**How can we configure the downloaded ‘chromedriver.exe’ in our selenium code?**

System.setProperty(“webdriver.chrome.driver”, “PATH\_OF\_EXE\_FILE”);

|  |
| --- |
| WebDriver driver = null;    if (browserName.equalsIgnoreCase("Chrome")) {  System.setProperty("webdriver.chrome.driver", "C:\\Users\\lmanoh572\\Documents\\Selenium\_Web\_Driver\_UDEMY\\chromedriver\\chromedriver.exe");  driver = new ChromeDriver();  } else if (browserName.equalsIgnoreCase("Firefox")) {  System.setProperty("webdriver.gecko.driver", "C:\\Users\\lmanoh572\\Documents\\Selenium\_Web\_Driver\_UDEMY\\firefoxdriver\\geckodriver.exe");  driver = new FirefoxDriver();  } else if (browserName.equalsIgnoreCase("Edge")) {  System.setProperty("webdriver.edge.driver", "C:\\Users\\lmanoh572\\Documents\\Selenium\_Web\_Driver\_UDEMY\\edgedriver\\msedgedriver.exe");  driver = new EdgeDriver();  } else {  // Most of the developers can't want to run in Internet Explorer because of its inconsistency and slowness  driver = new InternetExplorerDriver();  } |

**How the selenium will find the HTML elements and perform actions on it?**

Actions includes: click, type some texts etc.,

To perform all the actions selenium should locate the place where should it perform specific action.

Using LOCATORS selenium can easily identify the HTML elements and perform operations on it.

Locators:

* ID
* Xpath (Special)
* CSS Selector (Special)
* Tag Name
* Name
* Class Name
* Link Text
* Partial Link Text

To practice with Locators: <https://rahulshettyacademy.com/locatorspractice/>

*Why CSS Selector and Xpath is special?*

We don’t have any direct notation like ID or class name etc.,We are going to use the whole tag to construct the Unique value.

CSS Selector:

For eg:

<input id=”inputUsername” class=”” />

tagName.className 🡺input.

tagName#id 🡺 input#inputUsername

*What if no id or class or name?*

General Syntax: Tagname[attribute=”value”]

Note: Make sure attribute is Unique.

Xpath:

For eg:

<input id=”inputUsername” class=”” />

//TagName[@attribute[“value”]] 🡺 //input[id=”inputUsername”]

Note:

First Comes First Serve if we have more than one element that contains specific class.

*How can we find the searching class is unique or not?*

SelectorsHub or Chropath [Chrome Extensions]

**SelectorsHub:**

* Used to validate the XPATH and CSS Selectors.
* And it also provides the auto suggestions and debugging mechanisms to write the XPATH or CSS Selectors.

A screenshot of a computer

Description automatically generated with medium confidence

**Synchronisation:**

1. Implicit wait  
    Define the waiting time globally. So that if any element is not present execution will wait for the time mentioned.  
    Pros: No need to add waiting time at many places. So, code is well organised.  
    Cons: Performance Issue
2. Explicit wait  
    If we need waiting time specific element. We can specify using the Explicit wait.  
    Pros: No performance issue. Since we include the waiting time in wherever needed.  
    Cons: More code.
   1. WebDriverWait:  
        
      WebDriverWait w = new WebDriverWait(driver, 5); // 5 seconds  
      w.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(“”)));
   2. Fluent Wait:   
        
      Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5))

.ignoring(NoSuchElementException.class);

WebElement foo = wait.until(new Function<WebDriver, WebElement>() {

public WebElement apply(WebDriver driver) {

return driver.findElement(By.id("foo"));

}

});

Difference between ‘WebDriverWait’ and ‘FluentWait’

'WebDriverWait' is keep on polling into the Browser to identify the presence of element.

Whereas, in **'FluentWait'** we can give the Polling period also. If we  
give polling period as '2 seconds' polling to the browser will happen  
only for every 2 seconds.

1. Thread.sleep()  
    It is to PAUSE the execution.

**Interview Questions:**

1.Features of Selenium

2.Difference between **close()** and **quit()** methods

Close() -> to close the current window

Quit() -> to close all the windows opened by the selenium. For eg: in some places we will redirect to some other pages that also get closed when we use the quit() method.\

3. Difference between Implicit wait & Explicit wait.