**Selenium (Jason Huggins-2004)**

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.

It focuses on automating web-based applications.

It has four components:

**Selenium IDE, RC, WebDriver and Grid.**

**Selenium WebDriver (Simon Stewart-2006)**

It is used to automate web-based applications using different programming languages.

It directly **interacts with the browser**.

**Note:**

* Selenium RC has intermediate RC server in between your selenium commands and browser.
* Webdriver supports **headless HtmlUnit browser.**

**Different locaters**

Locating elements in WebDriver is done by using the "**find Element (By. *Locator* ())**" method.

* **By.id**
* **By.name**
* **By. xpath**
* **By.css Selector**
* **By. link Text**
* **By. class Name**

By.cssSelector () **does not** support the **"contains" feature**.

**Closing and Quitting Browser Windows**

**close ():** It closes only the browser window that WebDriver is currently controlling.

**quit ():** It closes all windows that WebDriver has opened **including all the pop-ups**.

**Frames:**

We can identify the frames/iframes using methods given below:

* **Right click** on the element, If you find the option like **'view frame source /This Frame**' then it is an frame/iframe.
* Right click on the page and click 'View Page Source' and Search with the ‘iframe’, if you can find any tag name with the 'iframe' then it is meaning to say the page consisting an iframe.

**How to switch over the elements in frames:**

1. By id
2. By name
3. By WebElement

To move back to the parent frame, you can either use **switchTo ().parentFrame ()** or if you want to get back to the main (or most parent) frame, you can use **switchTo ().defaultContent ().**

**Example:**Suppose if there are 100 frames in the page, and there is no ID available, in this case, we just don't know from which iframe required element is being loaded then we should follow below steps:

1. Find total number of frames.

2. Iterate each frame to check whether web element is present or not using for loop.

3. Print 1 for found frame else print 0.

4. Return to main frame.

**Concept of Nested Frames (Frames inside Frames):**

If there are 2 frames placed one inside other than follow below step:

* 1. Go to outer frame first using id or xpath then locate require element on it.
  2. Go back to inner frame.

**Switching Between Pop-up Windows:**

To access the elements within the alert:

* We must use the **"switchTo ().alert ()"** method and then retrieve its message using the **"get Text ()"** method.
* Automatically close the alert box using the **"switchTo ().alert ().accept ()"** method.
* **alert().dismiss() and alert().sendkeys()**

**Notes:**

* **isSelected ()** method is used to know whether the Checkbox is toggled on or off.
* **Select option from Dropdown using Selenium Webdriver:** Select by value or select by visible text.
* **By.linkText ()** and **By.partialLinkText ()**are both case-sensitive
* By.cssSelector and By. Xpath is used to access the image link
* Handling special keyboard and mouse events are done using the **Advanced User Interactions API**. It contains the **Actions** and the **Action** classes. Example click (), clickAndHold (), doubleclick (), contextclick () etc.
* **waitFor()** is used to wait for 30 seconds.

**X-PATH:**

It is a syntax or language for finding any element on the web page using XML path expression.

Xpath=//tagname [@attribute='value']

Xpath=//input [@name='uid']

**Types of X-path**

**1) Absolute XPath:** You can select the element from the root node.

**2) Relative XPath:** For Relative Xpath the path starts **from the middle of** the HTML DOM structure. It starts with the **double forward slash** (//), which means it can search the element **anywhere** at the webpage.

**Contains:** It is used when the value of any attribute changes dynamically.

Xpath=.//\* [contains (@name,'btn')]

**AND OR:**

Xpath=//input [@type='submit' AND @name='btnLogin'

**Start-with function** is used to find the element whose **attribute changes dynamically**.

Example: Id=" message12"

Id=" message345"

Id=" message8769"

Xpath=//label [starts-with (@id,'message')

**Parent:** Parent node of the given node. **Self:** node itself.

Xpath=//\*[@id='rt-feature']//parent::div

**Ancestor:** Parent or grandparent of the given node.

Xpath=//\*[text()='Enterprise Testing']//ancestor::div

**Following-siblings:** It will find the element after the current node.

xpath=//\*[@type='submit']//following-sibling::input

**Child:**

Xpath=//\*[@id='java\_technologies']/child::li

**Preceding**: All node comes before given node.

Xpath=//\*[@type='submit']//preceding::input

**Following:** Select all the element following current node.

Xpath=//\*[@type='text']//following::input

**Descendant:**

Xpath=//\*[@id='rt-feature']//descendant::a

**Handling Dynamic Web Tables Using Selenium WebDriver:**

Using X-Path to Locate Web Table Elements

* Find parent tag like div class.
* Find table, find table head th if you want column number and column td for row size.

Fetch number of rows and columns from Dynamic Web Table

Fetch cell value of a particular row and column of the Dynamic Table

**Get Maximum of all the Values in a Column of Dynamic Table**

Get all the values of a Dynamic Table.

**Number Format class:**

It is used to format and parse the number.

Example 2457,00 to 2457.00

NumberFormat f =NumberFormat.*getNumberInstance*();

Number num = f.parse(max);

**Implicit Wait & Explicit Wait in Selenium**

* **Implicit Wait:**

The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception".

driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS) ;

* **Explicit Wait:**

The explicit wait is used to tell the Web Driver to wait for certain conditions (**Expected Conditions**) or the maximum time exceeded before throwing an "**ElementNotVisibleException**" exception.

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( "xpath ")));

* **Fluent Wait:**

The fluent wait is used to tell the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an "ElementNotVisibleException" exception.

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(30, TimeUnit.SECONDS)

.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);