Selenium

## Selenium WebDriver- Definitions:-

*Selenium is an open-source (free) automated testing framework for validating web applications across multiple browsers and platforms.*

* *It is Java Interface containing n number of abstract methods and variables.*

*Web driver Interface contains multiple browser classes.*

* *It is also an API containing library of ‘n’ no of classes and methods by which we can communicate between Client n Server. Thus, it acts as an intermediary between Client n Server.*

***Advantages-***

A diagram of software development

Description automatically generated

* Open source and freely available.
* Support multiple languages🡪 java .Python, c#. Ruby…
* Support multiple OS🡪Windows, Linux. MacOS
* Support multiple browsers🡪 Cross Browser Testing🡪 chrome, FF, Edge, IE..
* Selenium can integrate third party tools🡪 TestNG, maven, Jenkins, Git & GitHub etc.

**Disadvantages-**

* Automate only web based app(do not automate-Standalone/Desktop based App eg, Calculator, MS Office etc.
* Does not have dedicated support and team for any issues faced.
* No reporting facility by default-reports
* Cannot support working with the excel sheets
* Do not support Image based testing.
* Does not support testing Captcha.

**To overcome these disadvantages, we have 3rd party tools eg:-**

* Does not support window- based app🡪 AutoIT Tool, sikuli
* Does not have dedicated support and team for any issues faced.--> Forums available.
* No reporting facility by default-reports🡪 TestNG , Extents Report.
* Cannot support working with the excels sheets🡪 Apache poi.
* Do not support Image based testing-->Certain Level🡪e.g. Pixels, positions etc.

## Components of Selenium?

There are 4 components of Selenium-

* Selenium IDE(Integrated Devlopment Environment)
* Selenium RC(Remote Control)
* Selenium WebDriver
* Selenium Grid

1. **Selenium IDE**🡪 plug-in for FF browser only, record and play back tool.

🡪Limitation- only for FF, cannot automate complex TC’s.

1. **RC**🡪Automate TC but interact with browser in-directly
   * Eg : TC🡪RC(server)-🡪 browser and vice-versa.
   * Limitation🡪

1.Thus, does not communicate directly with the browser., thus time consuming and lengthy process.

2.If RC server slows or goes down TC execution stops.

1. **Web-Driver**🡪It is an API .Thus, communicate directly with Browser.
2. **Selenium Grid**- Used to execute our TC on remote machines and distributed env.

* Selenium v-1 - IDE, RC, GRID
* Selenium v-2 - IDE, WebDriver, GRID🡪 WebDriver introduced and became popular instead of RC (completely removed).
* Selenium v-3 - WebDriver, GRID

## Hierarchy of Selenium 4 Webdriver

A diagram of a driver

Description automatically generated

**Search Context Interface**

Search Context is a super or top most interface in WebDriver hierarchy. This interface consists of only two methods findElement​() and findElements(). This interface is extended by both WebDriver and WebElement interfaces.

**WebDriver Interface**

WebDriver API🡪Packages🡪Classes🡪Methods(Packaged them in the jar files)🡪we cannot see the implementation of these methods but still use them.

Interface WebDriver{

m1(){}

m2(){}

m3(){}

}

class FireFoxDriver implemented WebDriver

{

m1(){body customized to ff}

m2(){body customized to ff}

m3( (){body customized to ff}

}

class Edge implemented WebDriver

{

m1(){body customized to Edge}

m2(){body customized to Edge }

m3( (){body customized to Edge }

}

class Chrome implemented WebDriver

{

m1(){body customized to Chrome}

m2(){body customized to Chrome}

m3( (){body customized to Chrome}

}

}

**RemoteWebDriver Class**

RemoteWebDriver, which is a fully implemented class where all abstract methods of WebDriver and SearchContext interface implemented.

Also, two other interfaces JavascriptExecutor and TakesScreenshot abstract methods are implemented in RemoteWebDriver class.

**Javascript Executor Interface**

JavaScript Executor is an Interface that helps to execute JavaScript through Selenium Webdriver.

JavaScript Executor provides two methods "executescript" & "executeAsyncScript" to run javascript on the selected window or current page.

**TakesScreenshot Interface**

TakesSchreenshot capture a screenshot and store it in different ways.

**Browser Specific Classes**

Browser specific classes provides control on a browser and provides different methods to test browser. Browser specific driver classes which extend RemoteWebDriver:

1. InternetExplorerDriver
2. FirefoxDriver
3. ChromiumDriver
4. OperaDriver
5. Safari Driver

Classes which extends ChromiumDriver are:

1. ChromeDriver
2. EdgeDriver

## Selenium Webdriver Architecture:

**Selenium 3-JSON wire protocol.**

A picture containing diagram, line, plan, sketch

Description automatically generated

**Selenium4- W3C Protcol**

Reason-JSON wire protocol requires encoding/decoding(encryption/decryption) while all other components follow W3C standards.

## Selenium Webdriver Architecture:

A picture containing diagram, sketch, line, plan

Description automatically generated

**Advantages🡪**

All follow W3C Standards 🡪stability in TC execution.

Additional features added eg-Action API etc.

Overcome challenges-All browser TC execution consistent.

## Selenium Setup:

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

## Download Drivers

A screenshot of a browser

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

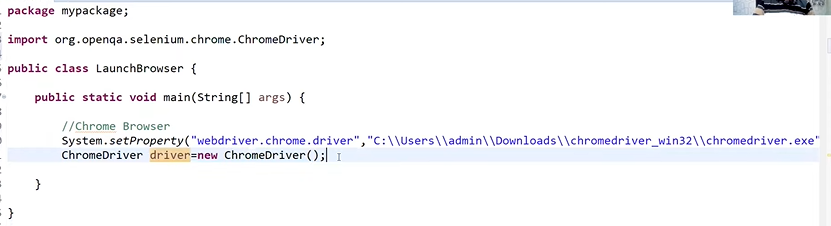
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## How to Launch Browsers in WebDriver?



WebDriver is an interface implemented by different browser classes.

WebDriver 🡪implemented by🡪 Chrome driver, Gecko driver, edge driver

Chrome driver Class will implement WebDriver Methods.

ChromeDriver driver = new ChromeDriver();

Once we create the object of the Chrome driver, then the default *constructor* of ChromeDriver class gets invoked which will launch the Chrome browser automatically.

We can write the same statement in diff way:

WebDriver driver= new ChromeDriver();

Note:

* We can create obj of *interface* but we cannot instantiate it.
* So instead of instantiating WebDriver interface we instantiate the browser class which we intend to use.
* Since Webdriver is an interface implemented by different browser classes , thus whatever objects are created for Browser class can hold WebDriver variables.
* So if we use WebDriver driver, then we can use the already initialized driver (as common object variable) for all browsers we want to automate
* e.g. Mozilla, Chrome, InternetExplorer, Safari.

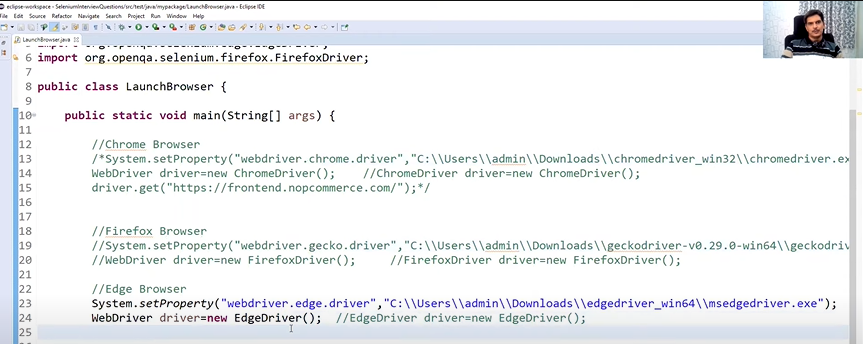
WebDriver driver = new FirefoxDriver();

driver = new ChromeDriver();

driver = new FirefoxDriver();

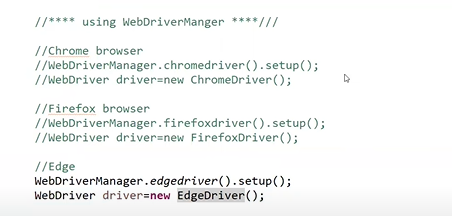
driver = new SafariDriver();

Launching different browsers



**Another way: no need to download drivers manually.**

Use-WebDriverManager dependency(API).-->add in pom.xml



## WebDriver Methods

1. Fetching a web page

There are two methods to fetch a web page:

* Using Get method-🡪 driver.get("-----URL----")
* Using Navigate method 🡪 driver.navigate().to("----URL-----");

2 .Sending user inputs

1. driver.findElement(By.id("---------")).sendKeys("-----Input----");

3. Clearing User inputs

The clear() method is used to clear the user inputs from the text box.

1. driver.findElement(By.name("---------")).clear();

4. Fetching data over any web element

To fetch the text written over a web element for performing some assertions and debugging. We use getText() method to fetch data written over any web element.

1. driver.findElement(By.id("---------")).getText();

5. Performing Click event

The click() method is used to perform click operation on any web element.

1. driver.findElement(By.id("-----------")).click();

6. Navigating backward in browser history

1. driver.navigate().back();

7. Navigating forward in browser history

1. driver.navigate().forward();

8. Refresh/ Reload a web page

1. driver.navigate().refresh();

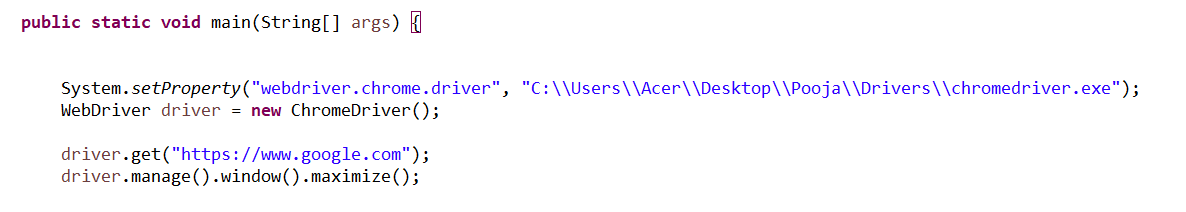
9. Closing Current Browser

1. driver.close();

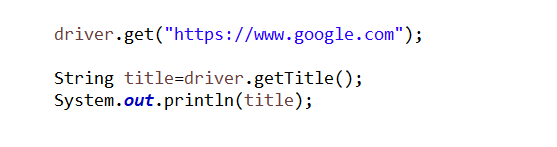
10. Closing Browser and other all other windows associated with the driver

1. driver.quit();

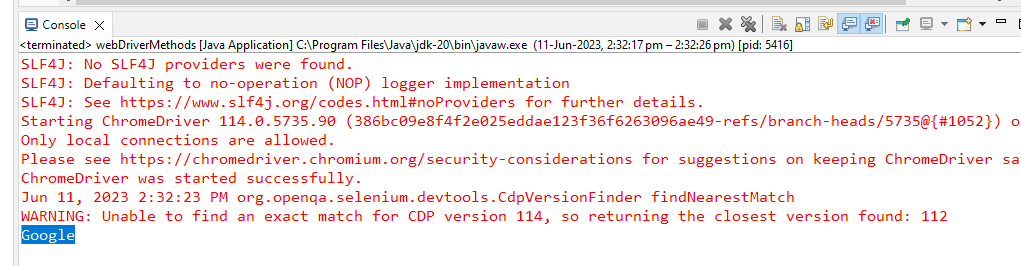
## How to Open URL?



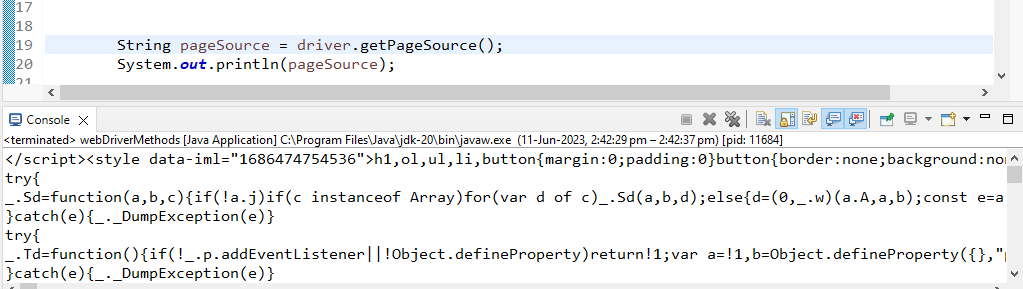
## How to Capture title/Current URL/ pageSource(html) of webpage ?

1. **Title:**

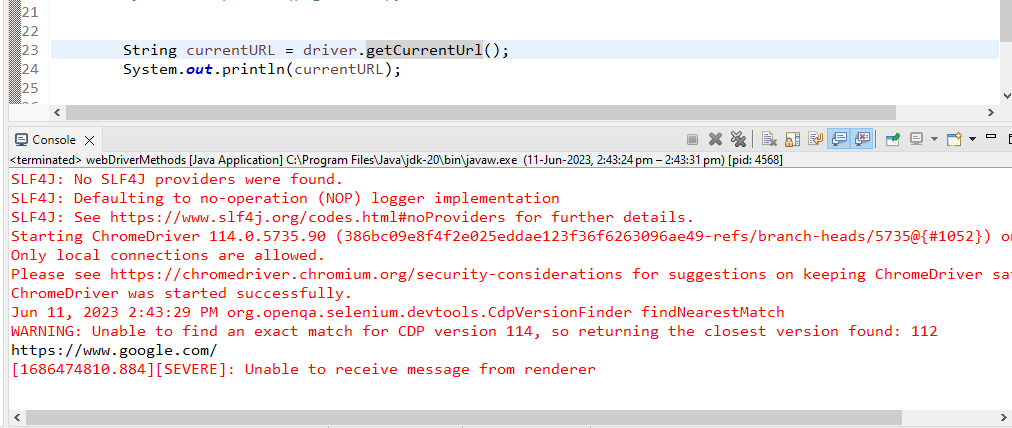
Console:



1. **PageSource(html):**

****

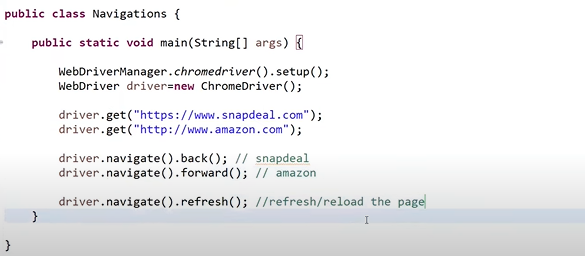
1. **Current URL:**

****

## Navigations on WebPage?

1. How to Navigate Back & Forward?

2. How to Refresh Page?



## **Practice Exercise - 1**

1. Launch a new Chrome browser.
2. Open desired webPage
3. Get Page Title name
4. Print Page Title on the Eclipse Console.
5. Get Page URL and verify if it is a correct page opened

String actualUrl = driver.getCurrentUrl();

if (actualUrl.equals(url)){

System.out.println("Verification Successful - The correct Url is opened.");

}

else {

System.out.println("Verification Failed - An incorrect Url is opened.");

1. Get Page Source (HTML Source code) on the Eclipse Console.
2. Close the Browser

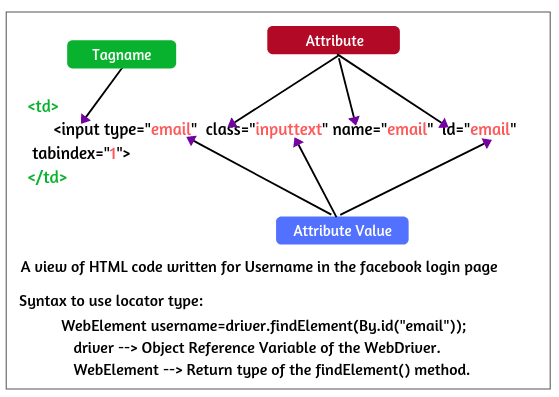
## What is WebElement?

* WebElement represents an ***HTML element***. HTML documents are made up by of HTML elements.
* HTML elements are written with a ***start*** tag, with an ***end*** tag, with the ***content*** in between: ***<tagname> content </tagname>***
* The HTML element is everything from the start tag to the end tag.
* HTML elements can be nested (elements can contain elements).
* WebElement can be of any type, like it can be a ***Text, Link, Radio Button, Drop Down, WebTable*** or any HTML element.

## Html

A screenshot of a computer

Description automatically generated



# Locators

We can perform a variety of actions on the WebElements but to perform these actions we first need to interact with a web page so that we can use WebElement Commands/actions.

WebElement elementName = driver.findElement

(By.LocatorStrategy("LocatorValue"));

| ***Locator*** | ***Description*** |
| --- | --- |
| ***id*** | finds elements by ID attribute. The search value given should match the ID attribute.  WebElement element = driver.findElement(By.id("elementId")); |
| ***name*** | Finds or Locates elements based on the NAME attribute. The name attribute is used to match the search value.  WebElement element = driver.findElement(By.name("elementName")); |
| ***class name*** | Finds elements that match the class name specified. Note that compound classes are not allowed as strategy names.  WebElement element = driver.findElement(By.className("className")); |
| ***tag name*** | Finds or Locates elements having tag names that match the search value.  WebElement element = driver.findElement(By.tagName("tagName")); |
| ***link text*** | Here the visible text whose anchor elements are to be found is matched with the search value.  WebElement element = driver.findElement(By.linkText("linkText")); |
| ***partial link text*** | Here also we match the visible text with the search value and find the anchor value. If we are matching multiple elements, only the first entry will be selected.  WebElement element =driver.findElement(By.partialLinkText("partialLinkText")); |
| ***CSS selector*** | Matches CSS selector to find the element.  WebElement element = driver.findElement(By.cssSelector("cssSelector")); |
| ***XPath*** | Matches XPath expression to the search value and based on that the element is located.  WebElement element = driver.findElement(By.xpath("xpathExpression")); |

# List of WebElement Commands/Actions

All interesting operations to do with interacting with a page will be performed through this ***WebElement Interface***.



## findElement()

***findElement*** command of ***WebDriver*** returns ***WebElement***.



So, to get the WebElement object write the below statement:

***WebElement element = driver.findElement(By.id("UserName"));***

When we type ***element dot***, Eclipse's intelli-sence will populate the complete list of actions just like the above image.

Q:What is the difference between findElement() and findElements() in Selenium WebDriver?

A screenshot of a computer

Description automatically generated

findElements(); can return one as well as multiple elements, but even when returning single element ,it will still store it in List<WebElement>.

For iteration of each element from list<WebElement> , we need create a loop statements.

## Clear Command

***clear( ) : void*** - If this element is a text entry element, this will clear the value. This method accepts nothing as a parameter and returns nothing.

***Command - element.clear();***

WebElement element = driver.findElement(By.id("UserName"));

element.clear();

//Or can be written as

driver.findElement(By.id("UserName")).clear();

## SendKeys Command

***sendKeys(CharSequence... keysToSend ) : void*** –

This simulates typing into an element, which may set its value. This method accepts CharSequence as a parameter and returns nothing.

***Command – element.sendKeys("text");***

## Click Command

Accepts nothing as a parameter and returns nothing.

***Command - element.click();***

Clicking is perhaps the most common way of interacting with web elements like text elements, links, radio boxes and many more.

WebElement element = driver.findElement(By.linkText("------"));

element.click();

//Or can be written as

driver.findElement(By.linkText("-----")).click();

## IsDisplayed Command

***isDisplayed( ) : boolean*** - This method determines if an element is currently being displayed or not. This accepts nothing as a parameter but returns a boolean value(true/false).

***Command - element.isDisplayed();***

WebElement element = driver.findElement(By.id("UserName"));

boolean status = element.isDisplayed();

//Or can be written as

boolean staus = driver.findElement(By.id("UserName")).isDisplayed();

***Note***:This will return ***true*** if the element is present on the page and throw a  ***NoSuchElementFound*** exception if the element is not present on the page.

## IsEnabled Command

***isEnabled( ) : boolean***- This determines if the element currently is ***Enabled or not?*** This accepts nothing as a parameter but returns boolean value(true/false).

***Command - element.isEnabled();***

This will generally return true for everything there aresometimes many disabled input elements in the web pages.

WebElement element = driver.findElement(By.id("UserName"));

boolean status = element.isEnabled();

## IsSelected Command

***isSelected( ) : boolean*** - Determine whether or not this element is selected or not. This accepts nothing as a parameter but returns boolean value(true/false).

***Command - element.isSelected();***

This operation only applies to input elements such as ***Checkboxes, Select Options***, and ***Radio Buttons***. This returns ***True*** if the element is currently selected or checked, ***false*** otherwise.

WebElement element = driver.findElement(By.id("Sex-Male"));

boolean status = element.isSelected();

//Or can be written as

boolean staus = driver.findElement(By.id("Sex-Male")).isSelected();

## Submit Command

\*\*\*submit( ) : void- \*\*\*This method works well/better than the click() if the current element is a form, or an element within a form. This accepts nothing as a parameter and returns nothing.

***Command - element.submit();***

If this causes the current page to change, then this method will wait until the new page is loaded.

WebElement element = driver.findElement(By.id("SubmitButton"));

element.submit();

//Or can be written as

driver.findElement(By.id("SubmitButton")).submit();

## GetText Command

***getText( ) : String-*** This method will fetch the visible (i.e. not hidden by CSS) innerText of the element. This accepts nothing as a parameter but returns a String value.

***Command - element.getText();***

WebElement element = driver.findElement(By.xpath("anyLink"));

String linkText = element.getText();

## getTagName Command

***getTagName( ) : String***- This method gets the tag name of this element. This accepts nothing as a parameter and returns a String value.

***Command - element.getTagName();***

WebElement element = driver.findElement(By.id("SubmitButton"));

String tagName = element.getTagName();

//Or can be written as

String tagName = driver.findElement(By.id("SubmitButton")).getTagName();

## getAttribute Command

***getAttribute***(String Name) : ***String***- This method gets the value of the given attribute of the element. This accepts the String as a parameter and returns a String value.

***Command - element.getAttribute();***

Attributes are Ids, Name, Class etc, and using this method you can get the value of the attributes of any given element.

WebElement element = driver.findElement(By.id("SubmitButton"));

String attValue = element.getAttribute("id"); //This will return "SubmitButton"