COX

Selenium Concept

February 14, 2020

BSS

JVM

Automation Testing:

- **Automation Testing or Test Automation** is a software testing technique that performs using special automated testing software tools to execute a test case suite. Tool Used Selenium/Java.

- Architecture

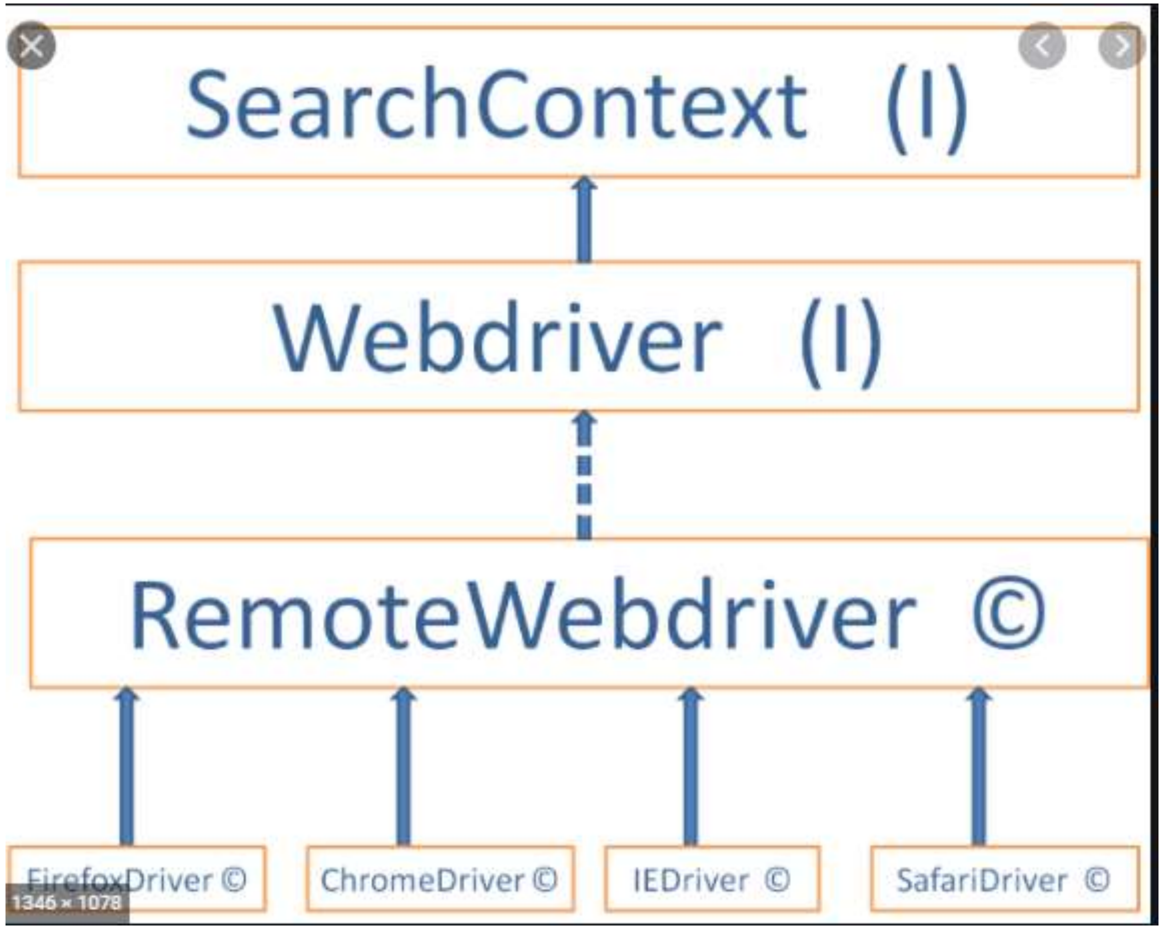


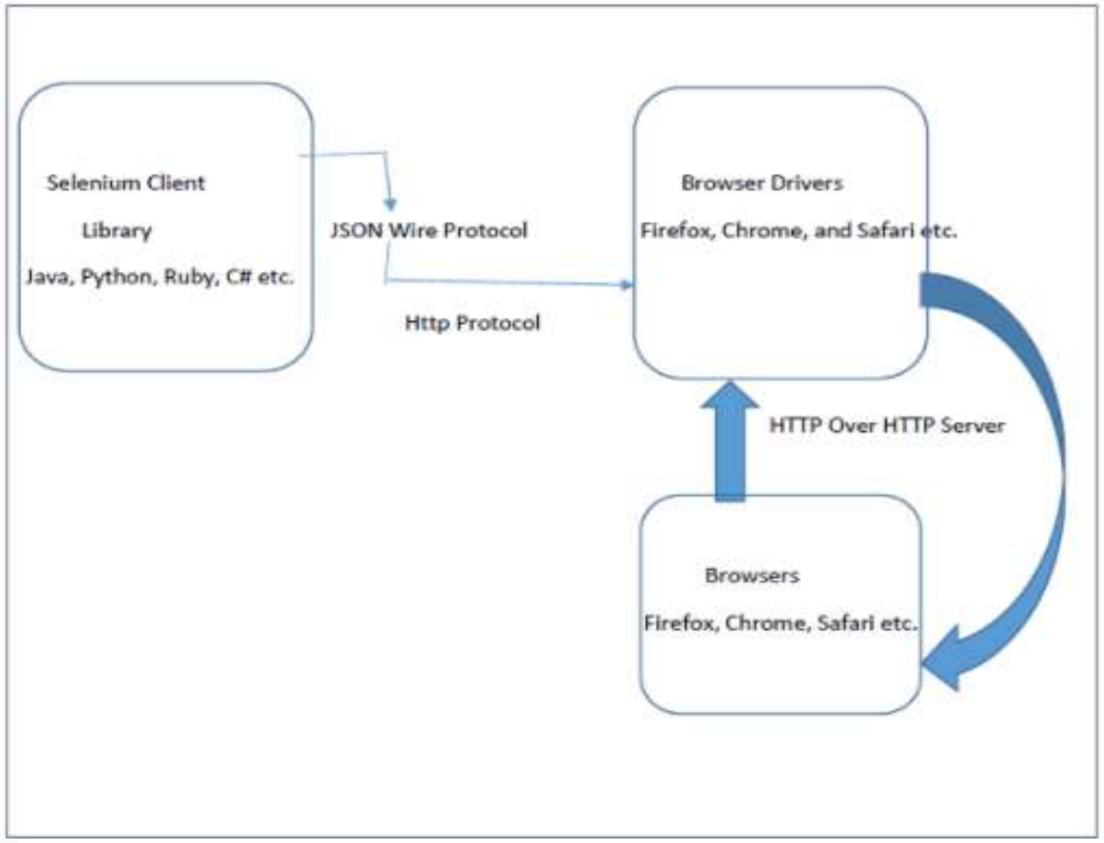
Fig : JVM Architecture

COX

Selenium Web Driver architecture in a simplified diagram is described below:

Let us now understand the Selenium Web Driver Architecture.
Selenium WebDriver API enables interaction between browsers and browser drivers. This architecture consists of four layers namely the Selenium Client Library, JSON Wire Protocol, Browser Drivers and Browsers.

- Selenium Client Library consists of languages like Java, Ruby, Python, C# and so on. After the test cases are triggered, entire Selenium code will be converted to Json format.
- JSON stands for Javascript Object Notation. It takes up the task of transferring information from the server to the client. JSON Wire Protocol is primarily responsible for transfer of data between HTTP servers. Generated Json is made available to browser drivers through http Protocol.
- Each browser has a specific browser driver. Browser drivers interact with its respective browsers and execute the commands by interpreting Json which they received from the browser. As soon as the browser driver gets any instructions, they run them on the browser. Then the response is given back in the form of HTTP response.



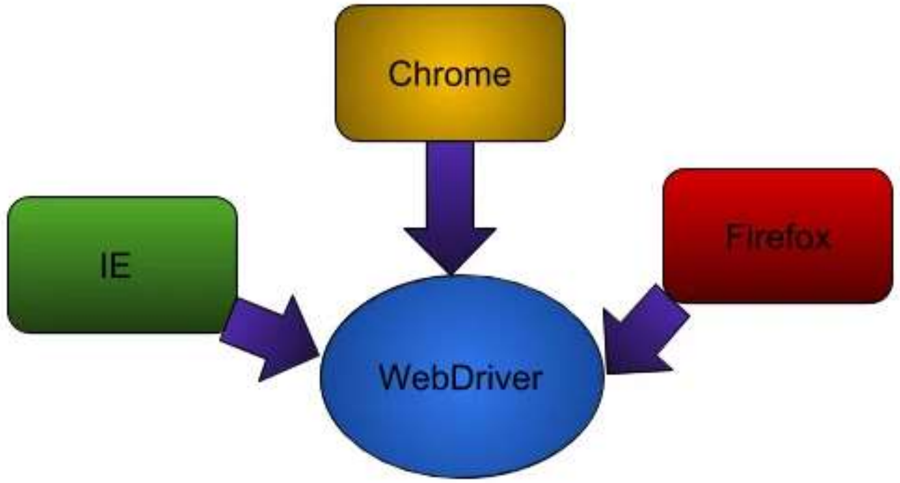
- *JRE = JVM + libraries to run Java application.*
- *JDK = JRE + tools to develop Java Application.*

What is Selenium WebDriver Interface?

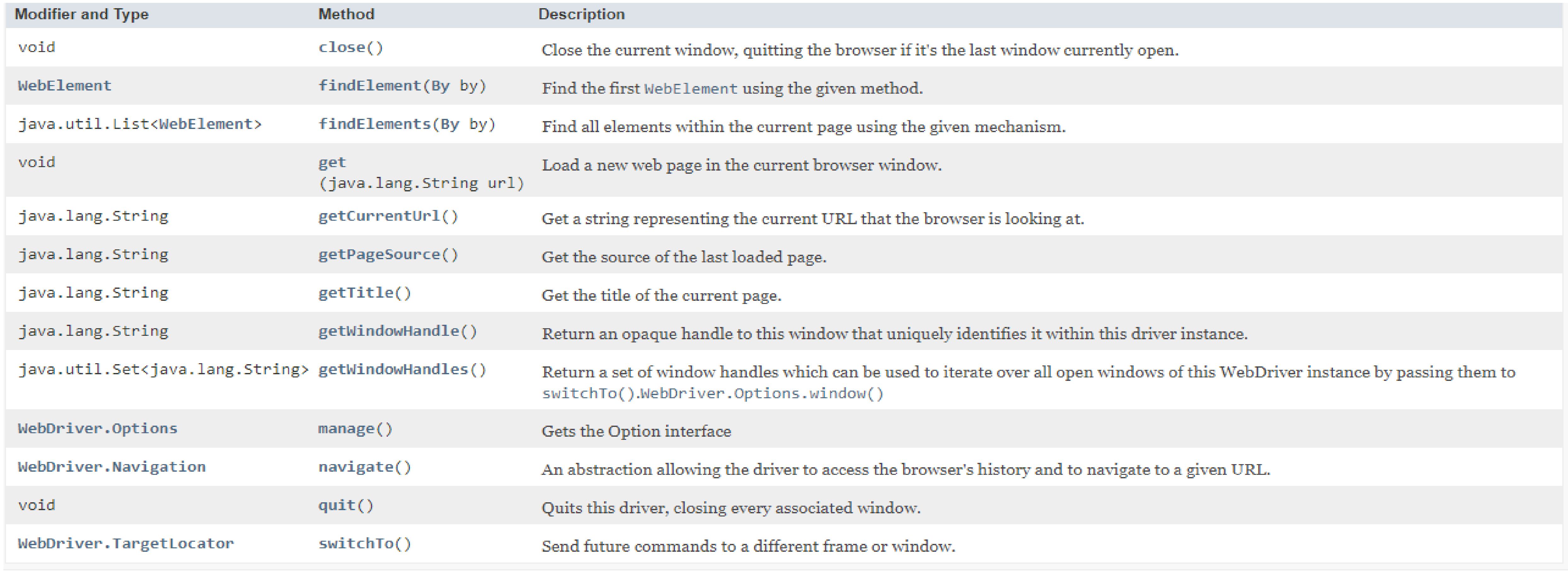
Selenium WebDriver is an interface that defines a set of methods. However, implementation is provided by the browser specific classes. Some of the implementation classes are `AndroidDriver`, `ChromeDriver`, `FirefoxDriver`, `InternetExplorerDriver`, `IPhoneDriver`, `SafariDriver` etc.

Tl
pa

The WebDriver main functionality is to control the browser. It even helps us to select the HTML page elements and perform operations on them such as click, filling a form fields etc.



Methods of WebDriver Interface?



Modifier and Type	Method	Description
`void`	`close()`	Close the current window, quitting the browser if it's the last window currently open.
`WebElement`	`findElement(By by)`	Find the first `WebElement` using the given method.
`java.util.List<WebElement>`	`findElements(By by)`	Find all elements within the current page using the given mechanism.
`void`	`get (java.lang.String url)`	Load a new web page in the current browser window.
`java.lang.String`	`getCurrentUrl()`	Get a string representing the current URL that the browser is looking at.
`java.lang.String`	`getPageSource()`	Get the source of the last loaded page.
`java.lang.String`	`getTitle()`	Get the title of the current page.
`java.lang.String`	`getWindowHandle()`	Return an opaque handle to this window that uniquely identifies it within this driver instance.
`java.util.Set<java.lang.String>`	`getWindowHandles()`	Return a set of window handles which can be used to iterate over all open windows of this WebDriver instance by passing them to `switchTo().WebDriver.Options.window()`
`WebDriver.Options`	`manage()`	Gets the Option interface
`WebDriver.Navigation`	`navigate()`	An abstraction allowing the driver to access the browser's history and to navigate to a given URL.
`void`	`quit()`	Quits this driver, closing every associated window.
`WebDriver.TargetLocator`	`switchTo()`	Send future commands to a different frame or window.

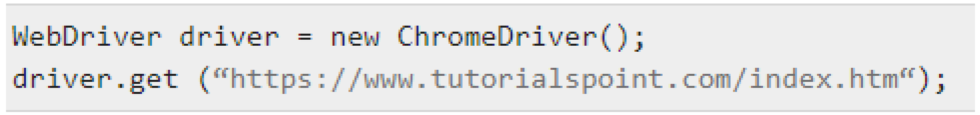
Methods of JavaScript executor?

- a) executeScript.
- b) executeAsyncScript.

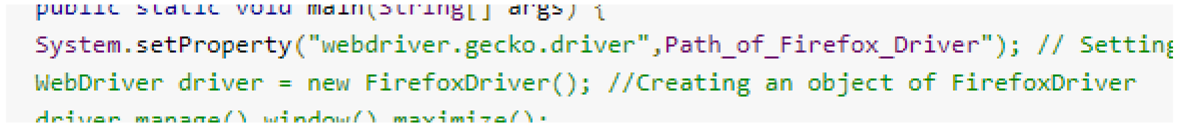
Methods of TakeScreenshot?

- getScreenshotAs **method**

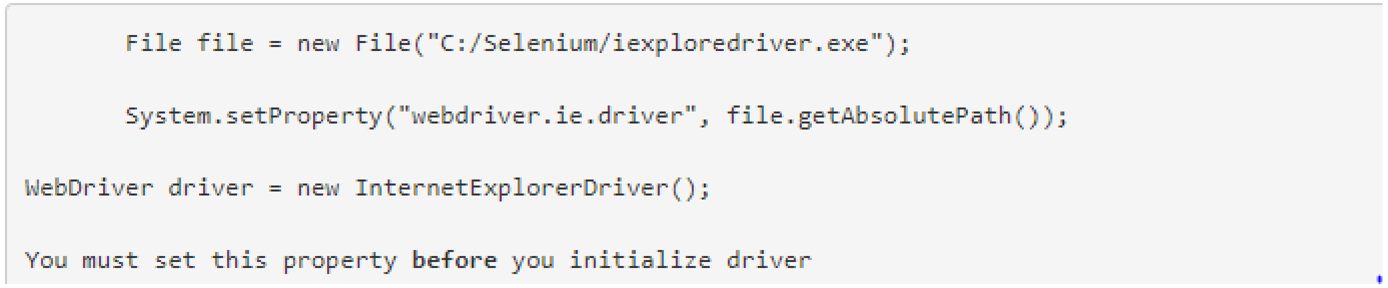
EXAMPLES



```
WebDriver driver = new ChromeDriver();
driver.get ("https://www.tutorialspoint.com/index.htm");
```



```
public static void main(String[] args) {
System.setProperty("webdriver.gecko.driver",Path_of_Firefox_Driver"); // Setting
WebDriver driver = new FirefoxDriver(); //Creating an object of FirefoxDriver
driver.manage().window().maximize();
```



```
        File file = new File("C:/Selenium/iexploredriver.exe");

        System.setProperty("webdriver.ie.driver", file.getAbsolutePath());

WebDriver driver = new InternetExplorerDriver();

You must set this property before you initialize driver
```

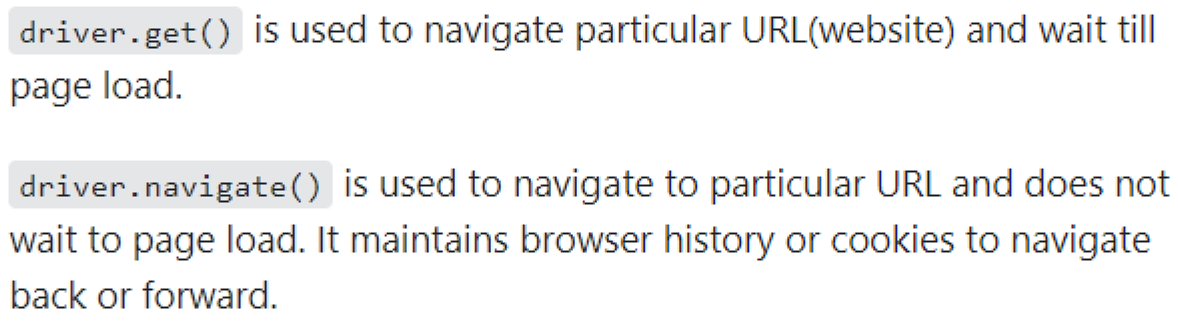
PROGRAM

PROGRAM

```
package Demo;

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

public class Class1 {

		public static void main(String[] args) {
			System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-
workspace\\AutomationDemo\\Driver\\chromedriver.exe");
			WebDriver driver=new ChromeDriver();

		driver.navigate().to("https://www.google.com/search?q=webdriver+in+selenium&source=hp&ei=LvCVYMj1MMjd9QPsta2oCg&iflsig=AINFCb
YAAAAAYJX-
PoCOJ9NnS_XpqJ4lo2_lIN9EPvx7&oq=webdrive&gs_lcp=Cgdnd3Mtd2l6EAEYATIFCAAQsQMyAggAMgUIABCxAzICCAAyAggAMggIABCxAxCDATICCAAyAggA
MgIIADICCAA6CAguELEDEIMBUKCsBFjuxgRg3NMEaABwAHgAgAH6AYgB6AmSAQUwLjYuMpgBAKABAaoBB2d3cy13aXo&sclient=gws-wiz");
			driver.manage().window().maximize();
    String Title=driver.getTitle();
    System.out.println(Title);
    String URL=driver.getCurrentUrl();
    System.out.println("Current URL" + URL);
    driver.navigate().back();
    driver.navigate().forward();
    driver.navigate().refresh();




		}

}
```

Get method & navigate method

Difference between Get() & Navigate()



`driver.get()` is used to navigate particular URL(website) and wait till page load.

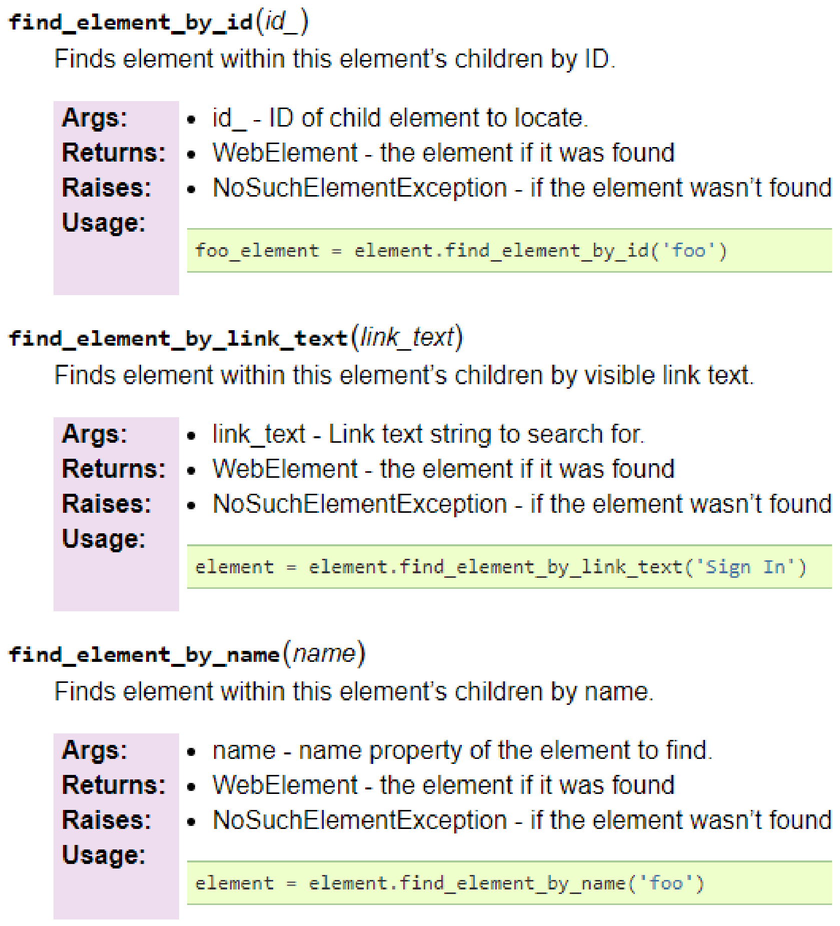
`driver.navigate()` is used to navigate to particular URL and does not wait to page load. It maintains browser history or cookies to navigate back or forward.

Delete Cookies

Driver.manage().deleteAllcookies()

WebElement

- WebElements which are based on different properties like ID, Name, Class, XPath, CSS Selectors, link Text, etc
- Locators:



`find_element_by_id`(*id_*)

Finds element within this element's children by ID.

Args:
- id_ - ID of child element to locate.

Returns:
- WebElement - the element if it was found

Raises:
- NoSuchElementException - if the element wasn't found

Usage:

```
foo_element = element.find_element_by_id('foo')
```

`find_element_by_link_text`(*link_text*)

Finds element within this element's children by visible link text.

Args:
- link_text - Link text string to search for.

Returns:
- WebElement - the element if it was found

Raises:
- NoSuchElementException - if the element wasn't found

Usage:

```
element = element.find_element_by_link_text('Sign In')
```

`find_element_by_name`(*name*)

Finds element within this element's children by name.

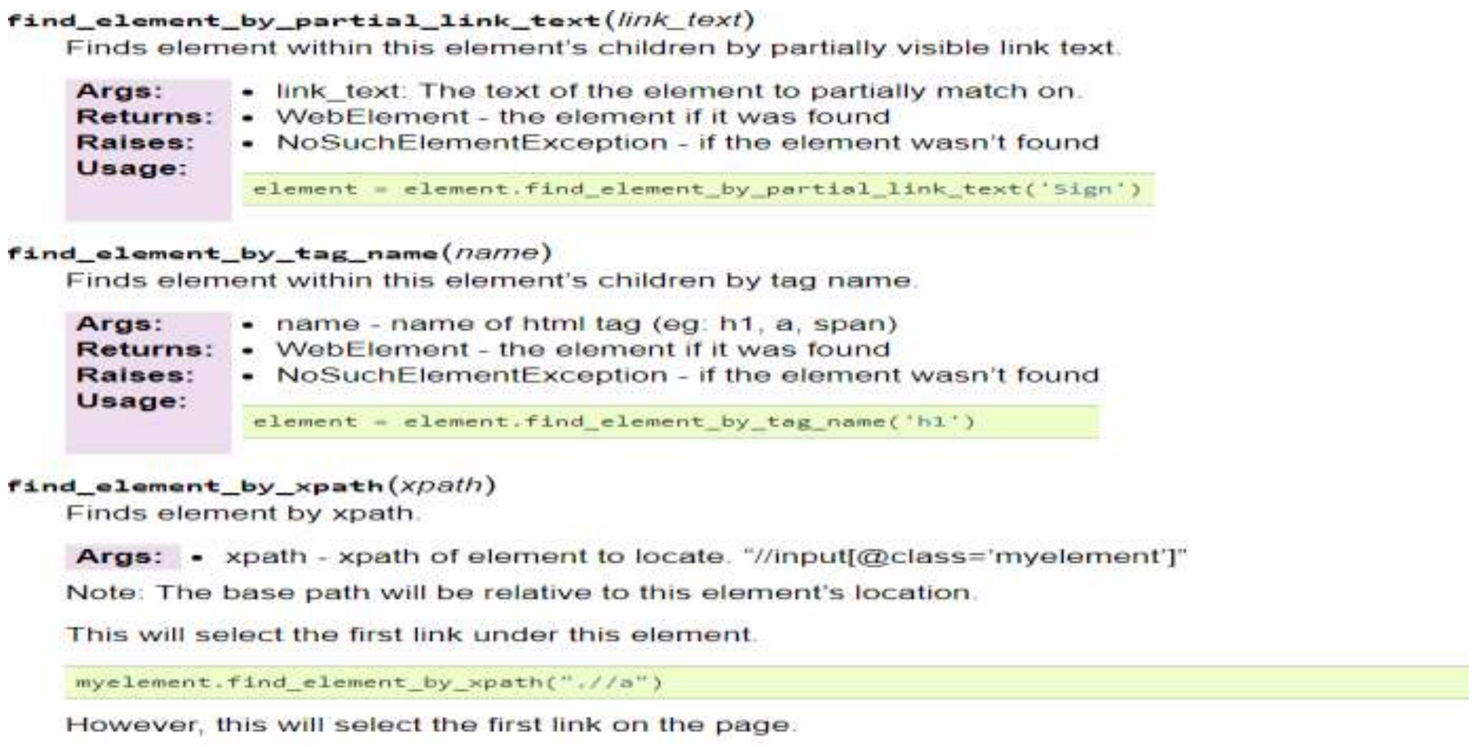
Args:
- name - name property of the element to find.

Returns:
- WebElement - the element if it was found

Raises:
- NoSuchElementException - if the element wasn't found

Usage:

```
element = element.find_element_by_name('foo')
```



`find_element_by_partial_link_text`(*link_text*)

Finds element within this element's children by partially visible link text.

Args:
- link_text: The text of the element to partially match on.

Returns:
- WebElement - the element if it was found

Raises:
- NoSuchElementException - if the element wasn't found

Usage:

```
element = element.find_element_by_partial_link_text('Sign')
```

`find_element_by_tag_name`(*name*)

Finds element within this element's children by tag name.

Args:
- name - name of html tag (eg: h1, a, span)

Returns:
- WebElement - the element if it was found

Raises:
- NoSuchElementException - if the element wasn't found

Usage:

```
element = element.find_element_by_tag_name('h1')
```

`find_element_by_xpath`(*xpath*)

Finds element by xpath.

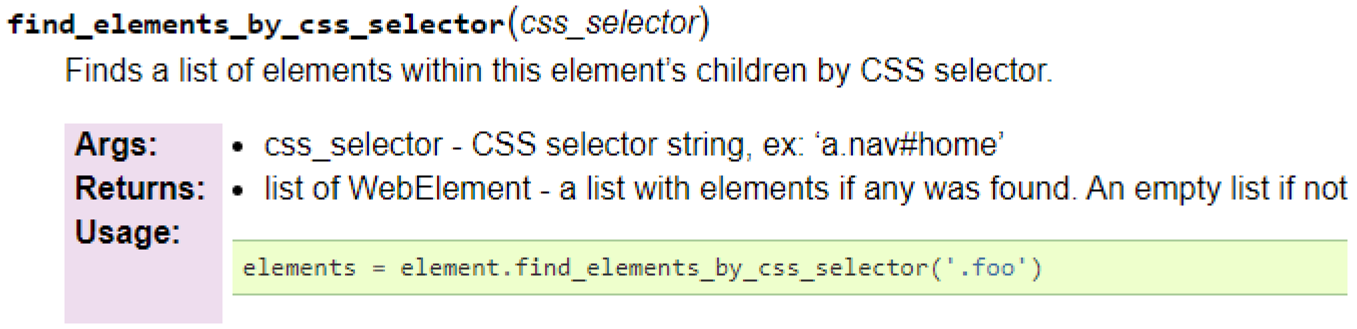
Args:
- xpath - xpath of element to locate. "//input[@class='myelement']"

Note: The base path will be relative to this element's location.

This will select the first link under this element.

```
myelement.find_element_by_xpath(".//a")
```

However, this will select the first link on the page.



`find_elements_by_css_selector`(*css_selector*)

Finds a list of elements within this element's children by CSS selector.

Args:
- css_selector - CSS selector string, ex: 'a.nav#home'

Returns:
- list of WebElement - a list with elements if any was found. An empty list if not

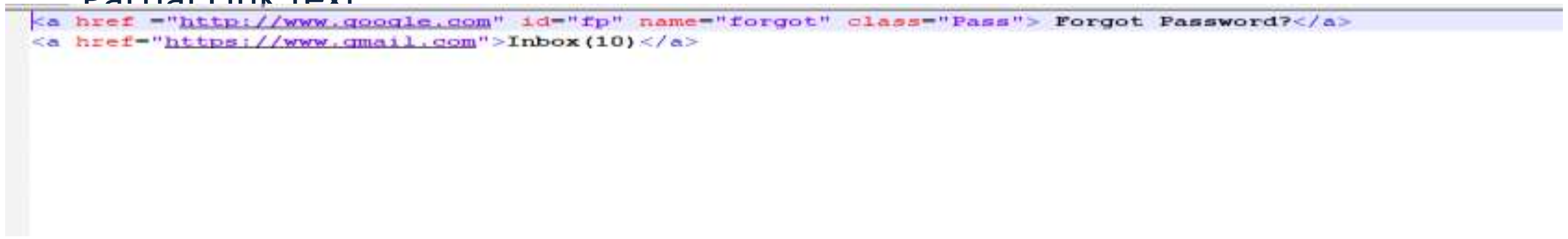
Usage:

```
elements = element.find_elements_by_css_selector('.foo')
```

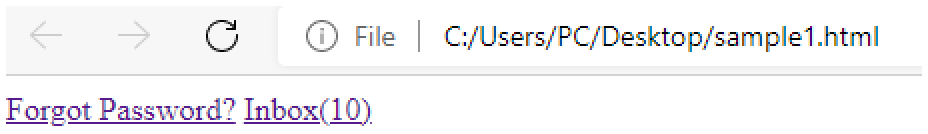
STATEMENT TYPES

WEBELEMENT

- We can handle the single element by using Find-Element Method
- In Find Element method , if the specified locator matching with multiple element,then it returns the address of first matching element.
- In Find Element method, if specified locator not matching with any element then it will throw No Such Element exception.
- We can use linkText, if specified element is link
- If specified element is link and if it partially dynamic then we can identify that element by using Locator Partial Link text



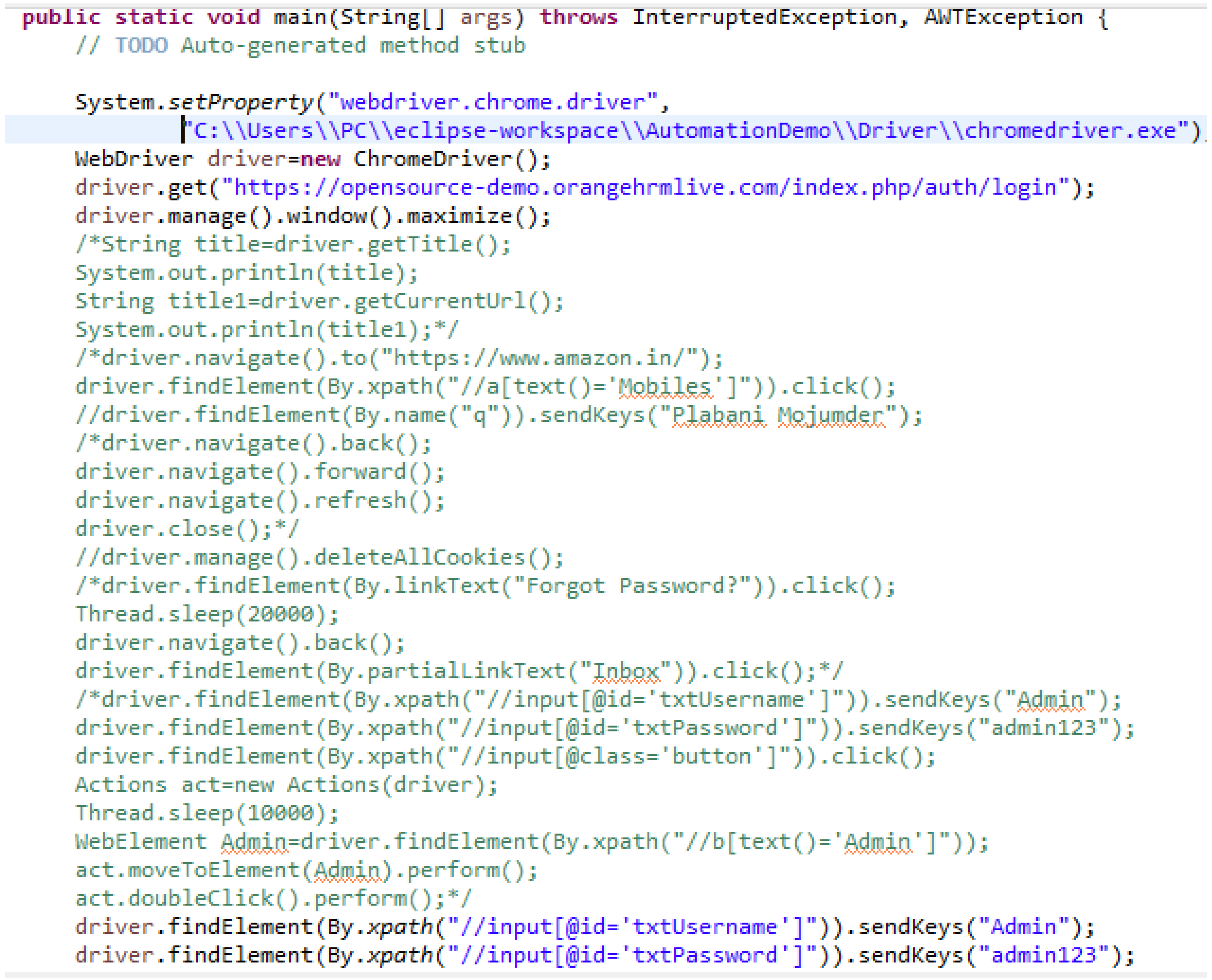
```
<a href ="http://www.google.com" id="fp" name="forgot" class="Pass"> Forgot Password?</a>
<a href="https://www.gmail.com">Inbox(10)</a>
```



File | C:/Users/PC/Desktop/sample1.html

Forgot Password? Inbox(10)

PROGRAM



```
public static void main(String[] args) throws InterruptedException, AWTException {
    // TODO Auto-generated method stub

    System.setProperty("webdriver.chrome.driver",
            "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe")
    WebDriver driver=new ChromeDriver();
    driver.get("https://opensource-demo.orangehrmlive.com/index.php/auth/login");
    driver.manage().window().maximize();
    /*String title=driver.getTitle();
    System.out.println(title);
    String title1=driver.getCurrentUrl();
    System.out.println(title1);*/
    /*driver.navigate().to("https://www.amazon.in/");
    driver.findElement(By.xpath("//a[text()='Mobiles']")).click();
    //driver.findElement(By.name("q")).sendKeys("Plabani Mojumder");
    /*driver.navigate().back();
    driver.navigate().forward();
    driver.navigate().refresh();
    driver.close();*/
    //driver.manage().deleteAllCookies();
    /*driver.findElement(By.linkText("Forgot Password?")).click();
    Thread.sleep(20000);
    driver.navigate().back();
    driver.findElement(By.partialLinkText("Inbox")).click();*/
    /*driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("Admin");
    driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin123");
    driver.findElement(By.xpath("//input[@class='button']")).click();
    Actions act=new Actions(driver);
    Thread.sleep(10000);
    WebElement Admin=driver.findElement(By.xpath("//b[text()='Admin']"));
    act.moveToElement(Admin).perform();
    act.doubleClick().perform();*/
    driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("Admin");
    driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin123");
```

XPATH

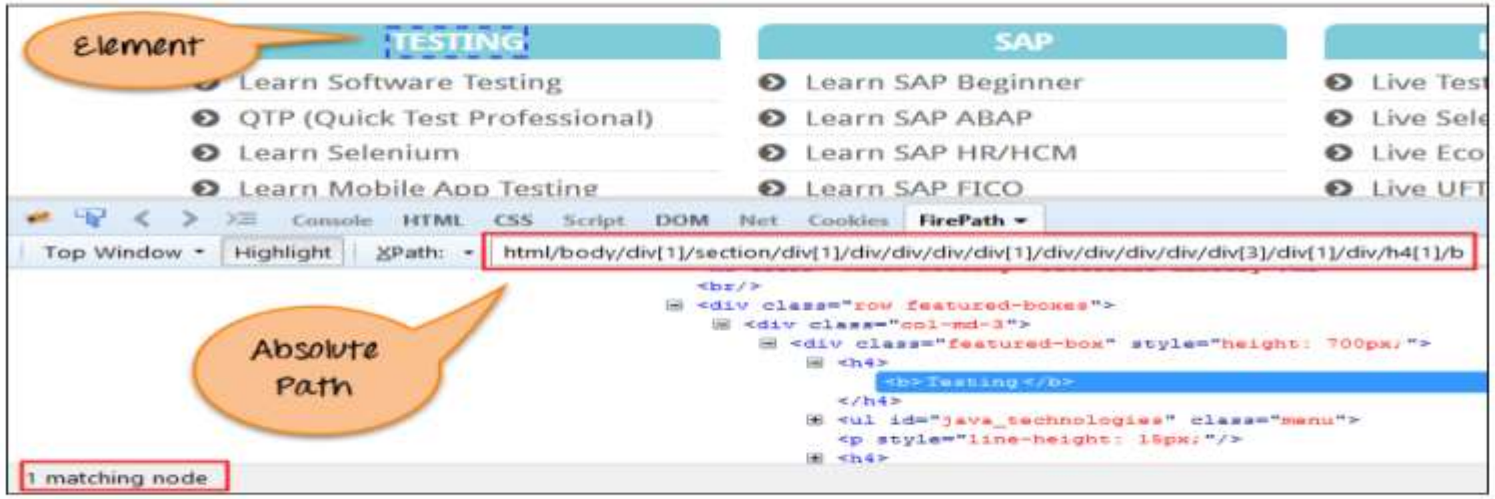
CSS SELECTOR:
Driver.findelement(By.CSSelector(“input[type=‘Password’])).sendkeys(“Plabani123”)

Xpath

- **Absolute Xpath:**The **absolute xpath** has the complete path beginning from the root to the element which we want to identify. An **absolute xpath** starts with the / symbol.
- **Relative Xpath:**The relative **xpath** starts by referring to the element that we want to identify and not from the root node. A relative **xpath** starts with the // symbol**.**

Absolute XPath:

```
/html/body/div[2]/div[1]/div/h4[1]/b/html[1]/body[1]/div[2]/div[1]/div[1]/h4[1]/
b[1]
```

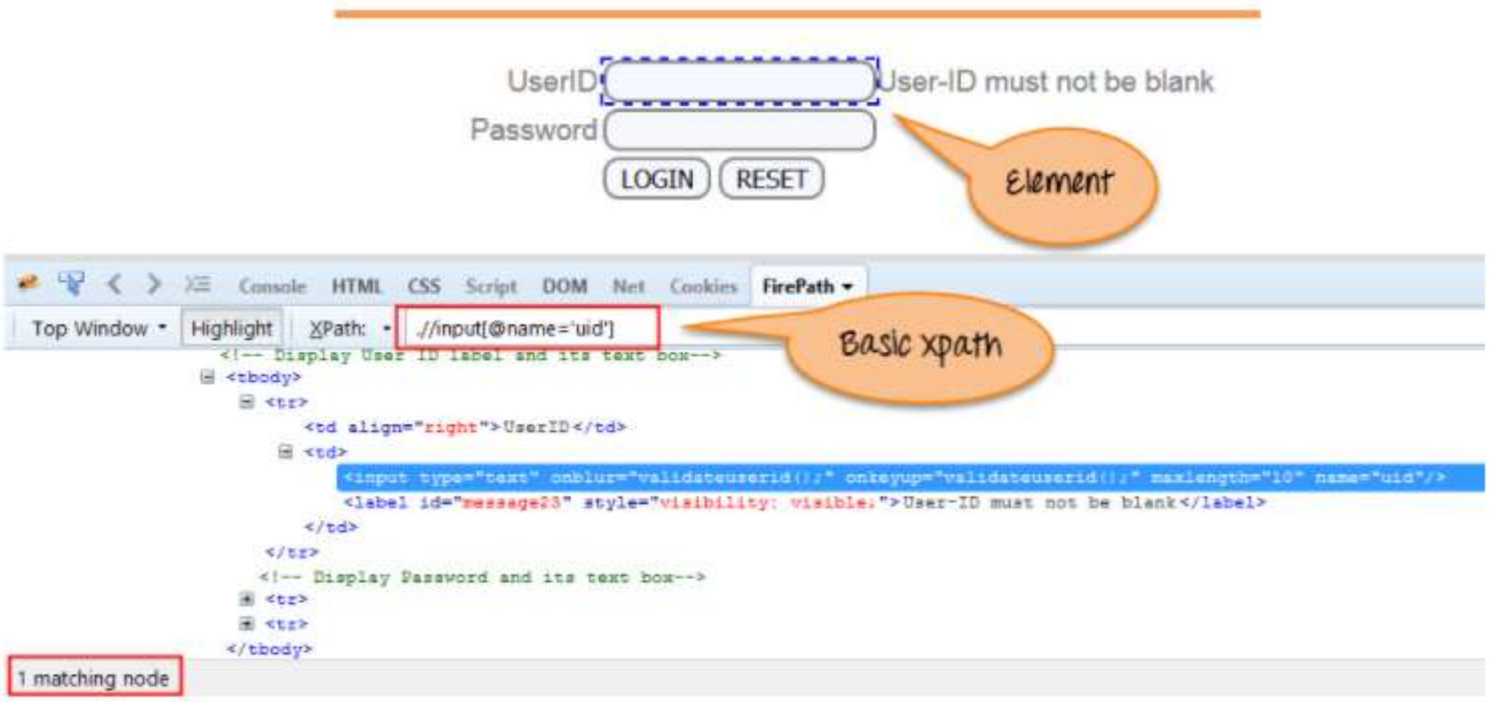


```
html/body/div[1]/section/div[1]/div/div/div[1]/div/div/div/div[3]/div[1]/div/h4[1]/b
```

```
<br/>
<div class="row featured-boxes">
  <div class="col-md-3">
    <div class="featured-box" style="height: 700px;">
      <h4>
          <b>Testing</b>
      </h4>
      <ul id="java_technologies" class="menu">
      <p style="line-height: 15px;"/>
      <h4>
```

Absolute XPath

Relative Xpath



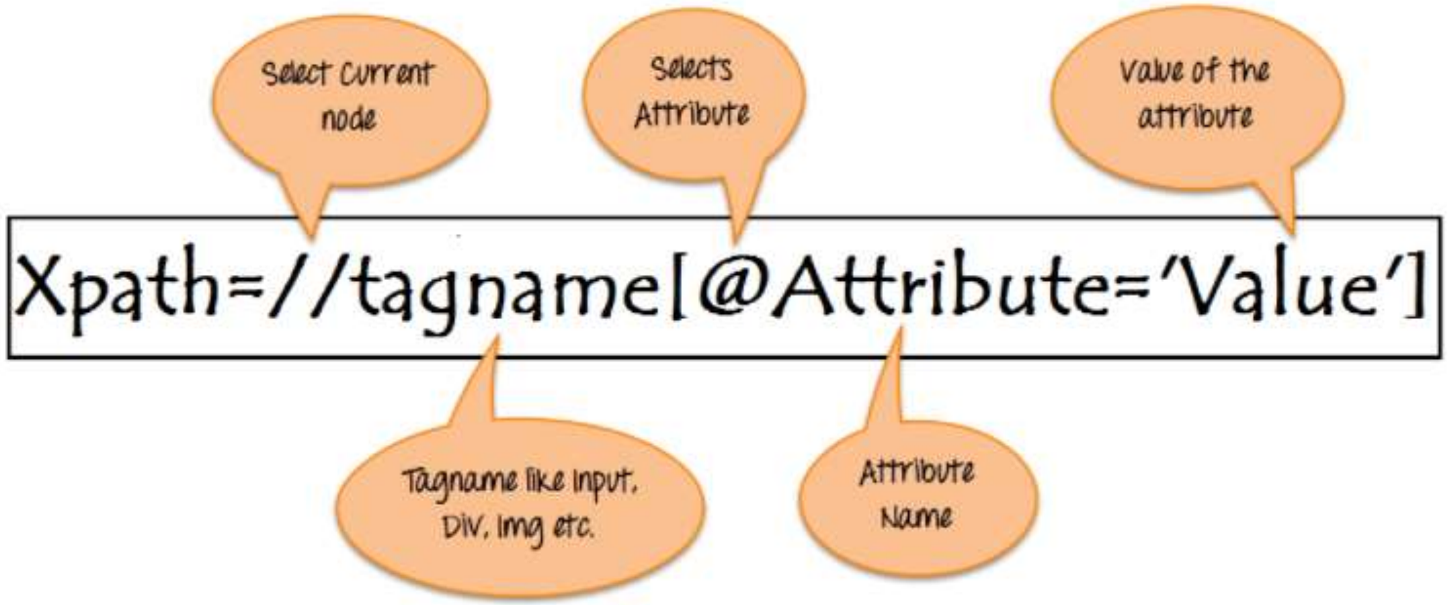
```
.//input[@name='uid']
```

```
<!-- Display User ID label and its text box-->
<tbody>
  <tr>
      <td align="right">UserID</td>
    <td>
        <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
        <label id="message23" style="visibility: visible;">User-ID must not be blank</label>
      </td>
  </tr>
  <!-- Display Password and its text box-->
  <tr>
  <tr>
</tbody>
```

1 matching node

Basic XPath

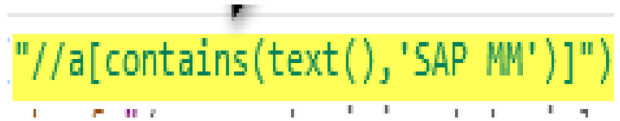
Xpath by Attribute



Xpath by text
//a[text()="Ask Question"]
//a[.="Ask Question"]

XPATH BY CONTAINS

If specified element partially dynamic,then we can identify that element by using Xpath by Contains



```
"//a[contains(text(),'SAP MM')]")
```

Traversing

Navigating from one element to another element using xpath is called as Traversing

Xpath axes

following: This function will return the immediate element of the particular component.

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

Preceding: This function will return the preceding element of the particular element.

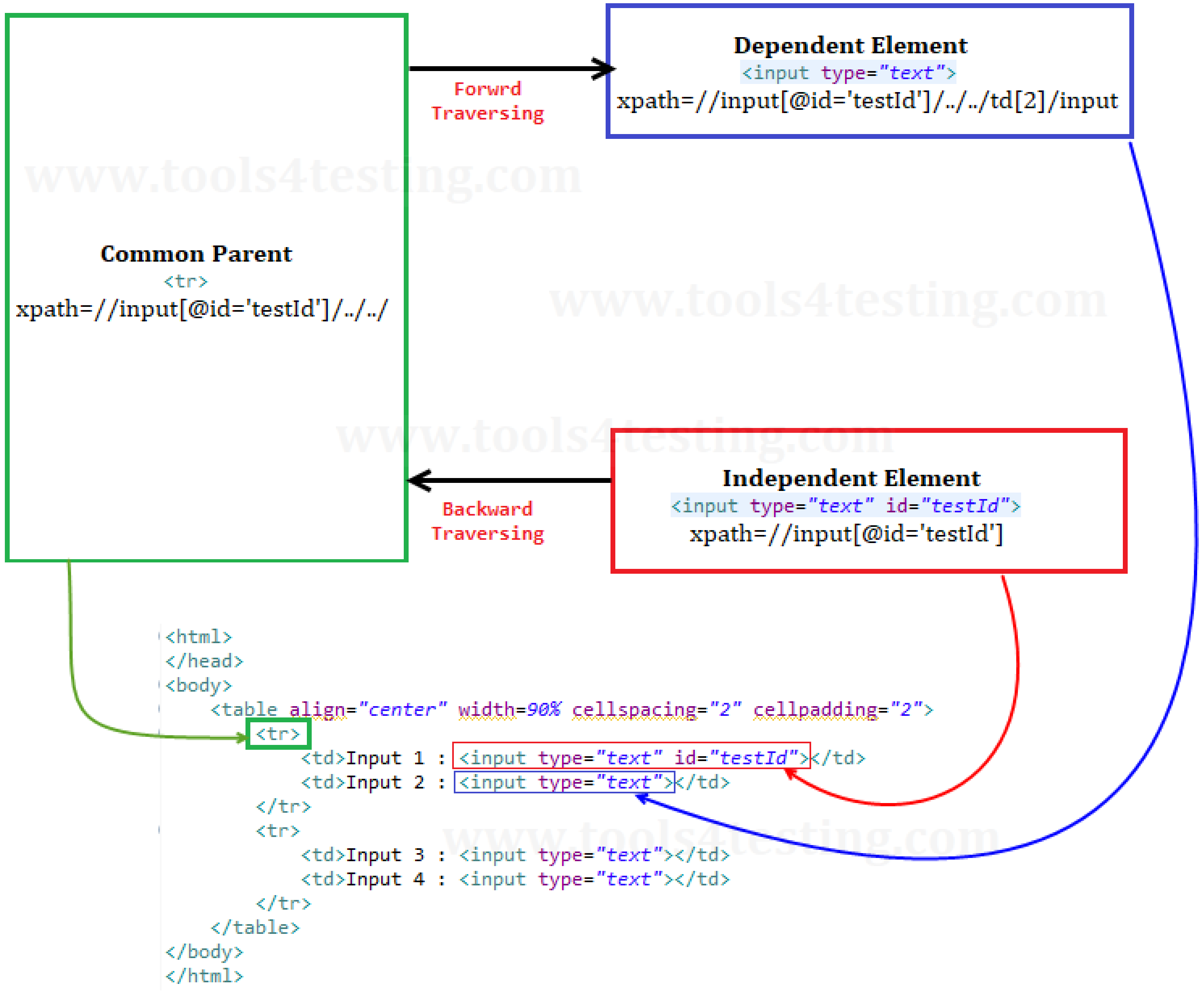
Xpath= //*[@type='submit']//preceding::input

Child: /

Parent:/..

Descendant://

Independent-Dependent Xpath

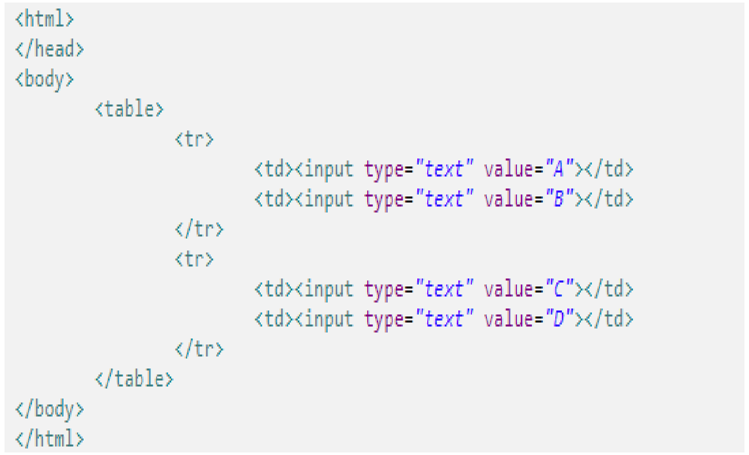


Dependent Element
<input type="text">
xpath=//input[@id='testId']/../../td[2]/input
Forwrd Traversing
Common Parent
<tr>
xpath=//input[@id='testId']/../../
Backward Traversing
Independent Element
<input type="text" id="testId">
xpath=//input[@id='testId']

```
<html>
</head>
<body>
    <table align="center" width=90% cellspacing="2" cellpadding="2">
        <tr>
            <td>Input 1 : <input type="text" id="testId"></td>
            <td>Input 2 : <input type="text"></td>
        </tr>
        <tr>
            <td>Input 3 : <input type="text"></td>
            <td>Input 4 : <input type="text"></td>
        </tr>
    </table>
</body>
</html>
```

XPATH BY INDEX

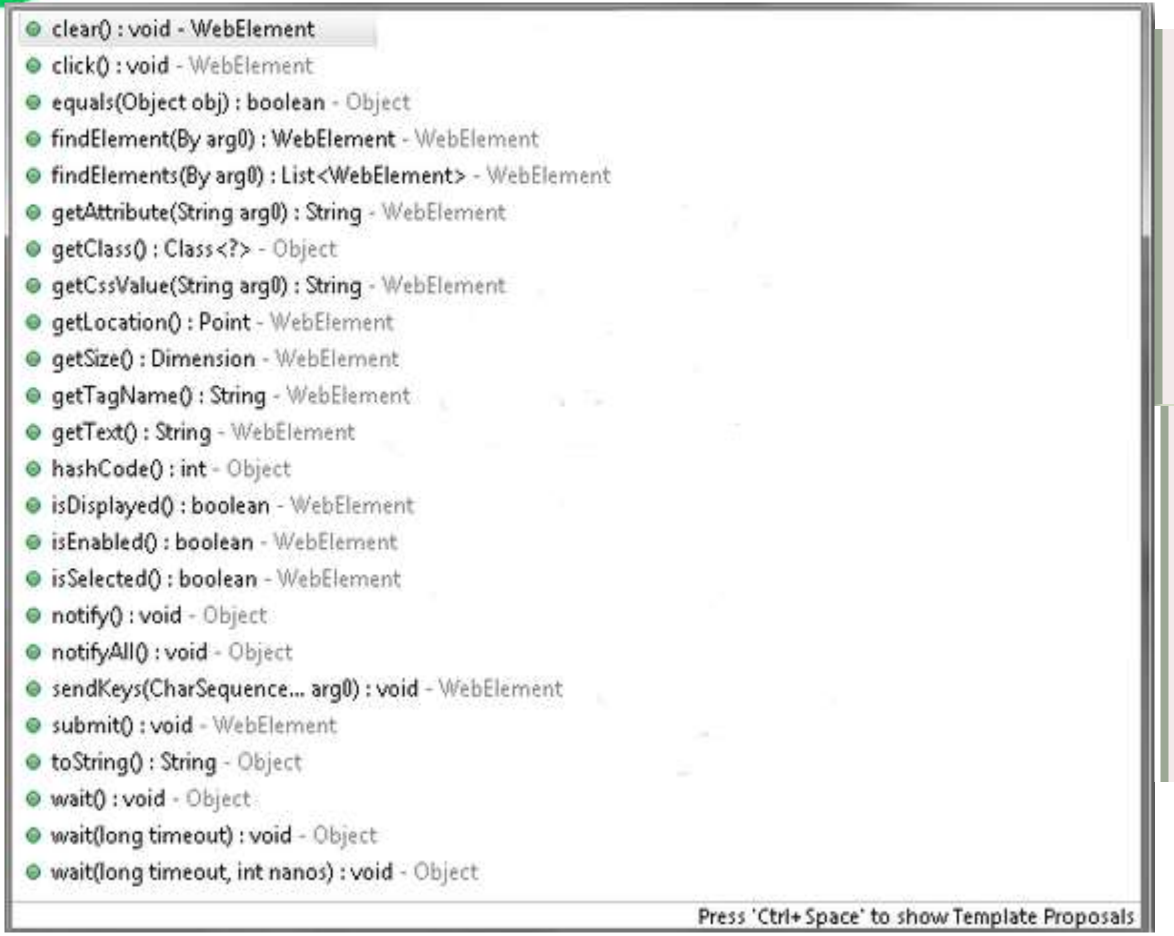
Even After using Independent/Depender
that element by using group index



```
<html>
</head>
<body>
        <table>
                <tr>
                        <td><input type="text" value="A"></td>
                        <td><input type="text" value="B"></td>
                </tr>
                <tr>
                        <td><input type="text" value="C"></td>
                        <td><input type="text" value="D"></td>
                </tr>
        </table>
</body>
</html>
```

//input	ABCD
(//input)[1]	A
(//input)[2]	B
(//input)[3]	C
(//input)[4]	D
(//input)[last()]	D
(//input)[last()-1]	C
//input[1]	AC
(//input[1])[1]	A
(//input[1])[2]	C
(//input[1])[last()]	C
//input[2]	BD

WEB-ELEMENT



```
clear() : void - WebElement
click() : void - WebElement
equals(Object obj) : boolean - Object
findElement(By arg0) : WebElement - WebElement
findElements(By arg0) : List<WebElement> - WebElement
getAttribute(String arg0) : String - WebElement
getClass() : Class<?> - Object
getCssValue(String arg0) : String - WebElement
getLocation() : Point - WebElement
getSize() : Dimension - WebElement
getTagName() : String - WebElement
getText() : String - WebElement
hashCode() : int - Object
isDisplayed() : boolean - WebElement
isEnabled() : boolean - WebElement
isSelected() : boolean - WebElement
notify() : void - Object
notifyAll() : void - Object
sendKeys(CharSequence... arg0) : void - WebElement
submit() : void - WebElement
toString() : String - Object
wait() : void - Object
wait(long timeout) : void - Object
wait(long timeout, int nanos) : void - Object
Press 'Ctrl+Space' to show Template Proposals
```

```
WebElement element = driver.findElement(By.id("UserName"));
boolean status = element.isDisplayed();

//Or can be written as

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

```
WebElement element = driver.findElement(By.id("UserName"));
boolean status = element.isEnabled();

//Or can be written as

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

Handling Multiple Elements

- We can handle multiple element by using FindElemets method
- The return type of findelements() method is List<WebElement>

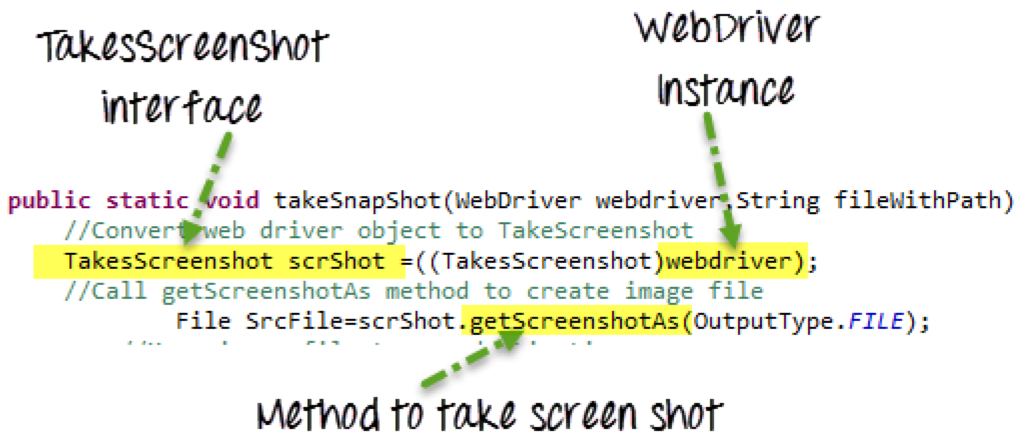
```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;


public class Handling_AutoSuggestion {

	public static void main(String[] args) throws InterruptedException {
		// TODO Auto-generated method stub
		System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
		WebDriver driver=new ChromeDriver();
		driver.get("https://www.google.com/");
		driver.findElement(By.name("q")).sendKeys("Qspiders");
		//Thread.sleep(10000);
		String xp="//span[contains(text(),'QSpiders')]";
		java.util.List<WebElement> allsuggestion=driver.findElements(By.xpath(xp));
		Thread.sleep(10000);

		int count =allsuggestion.size();
		Thread.sleep(10000);
		System.out.println(count);
		Thread.sleep(10000);

		for(int i=0;i<count;i++)
		{
			WebElement suggestion=allsuggestion.get(i);
			String text=suggestion.getText();
			System.out.println(text);

		if(text.equals("QSpiders BTM"))
		{
			suggestion.click();
			break;
		}
	}
```

Diff Between Find Element & Find Elements:

findElement	**findElements**
Returns the first matching web element if multiple web elements are discovered by the locator	Returns a list of multiple matching web elements
Throws **NoSuchElementException** if the element is not found	Returns an empty list if no matching element is found
Detects a unique web element	Returns a collection of matching elements

TakeScreenShot

To copy files from one folder to another folder we use API called commons-IO,



```
public static void takeSnapShot(WebDriver webdriver,String fileWithPath)
    //Convert web driver object to TakeScreenshot
    TakesScreenshot scrShot =((TakesScreenshot)webdriver);
    //Call getScreenshotAs method to create image file
            File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);
```

TakeScreenshot

```
package com.automation.practice;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Project_TakeScreenshot {

		public static void main(String[] args) throws IOException {
				// TODO Auto-generated method stub
				System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-
workspace\\AutomationDemo\\Driver\\chromedriver.exe");
				WebDriver driver=new ChromeDriver();
				driver.get("https://www.google.com/");

				TakesScreenshot t=(TakesScreenshot)driver;
				File src=t.getScreenshotAs(OutputType.FILE);
				File dest=new File("C:\\Users\\Default\\Downloads\\Screenshot.png");
				FileUtils.copyFile(src, dest);

		}

}
```

SYNCHRONIZATION

Matching the speed of selenium with the speed of application, is called as Synchornization

- Implicit wait
- Explicit wait
- Thread.sleep()

Implicit

```
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Sample_WaitImpli {

		public static void main(String[] args) {
				// TODO Auto-generated method stub
				System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
				WebDriver driver=new ChromeDriver();
				driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
				driver.get("https://opensource-demo.orangehrmlive.com/");
				driver.findElement(By.id("txtUsername")).sendKeys("Admin");
				driver.findElement(By.id("txtPassword")).sendKeys("admin123");
	driver.findElement(By.id("btnLogin")).click();

		}

}
```

Wait

Explicit

```
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.Test;

public class Explicit_wait {

		@Test
		public void test()
		{
				System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-
workspace\\AutomationDemo\\Driver\\chromedriver.exe");
				WebDriver driver=new ChromeDriver();
				driver.get("https://opensource-demo.orangehrmlive.com/");
				//driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
				WebDriverWait wait=new WebDriverWait(driver, 10);
				try
				{
				wait.until(ExpectedConditions.titleContains("actiTime"));
				System.out.println("passed");
				}
				catch (Exception e) {
						System.out.println("Failed");

				}

		}

}
```

HANDLING LISTBOX/DROPDOWNBOX

If the ListBox is developed by using, select tag, then we can handle the Listbox by using "Select Class"

Methods of Select class

- Selectbyindex(int)
- SelectbyValue(string)
- SelectByVisiblecheck(string)

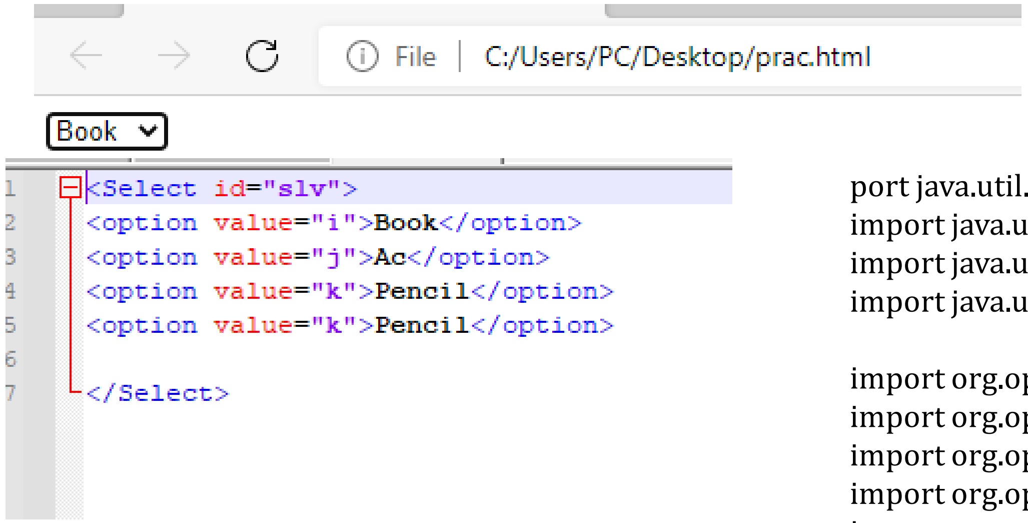
Deselect the option(Applicable for Multiselect tag)

- DeselctAll()
- DeselectbyIndex()
- DeselectByValue()

Get methods

- getoptions()-→List<WebElement>
- getAllselectedoption()
- getFirstSelectedOptions—List<WebElement>

Select



C:/Users/PC/Desktop/prac.html

Book

```
<Select id="slv">
<option value="i">Book</option>
<option value="j">Ac</option>
<option value="k">Pencil</option>
<option value="k">Pencil</option>

</Select>
```

```
port java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;


public class PracSelect {

		public static void main(String[] args) throws InterruptedException {
			// TODO Auto-generated method stub
			System.setProperty("webdriver.chrome.driver",
"C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
			WebDriver driver=new ChromeDriver();
			driver.get("file:///C:/Users/PC/Desktop/prac.html");
			WebElement selectobj=driver.findElement(By.id("slv"));
			Select sel=new  Select(selectobj);
		/*0read.sleep(10000);
			//sel.selectByIndex(1);
			sel.selectByValue("j");
			Thread.sleep(10000);

			sel.selectByVisibleText("Pencil");*/
```

SELECT LAST OPTION

```
public class PracSelect {

public static void main(String[] args) throws InterruptedException
{
System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
WebDriver driver=new ChromeDriver();
driver.get("file:///C:/Users/PC/Desktop/prac.html");
WebElement selectobj=driver.findElement(By.id("slv"));
Select sel=new  Select(selectobj);
/*Thread.sleep(10000);
//sel.selectByIndex(1);
sel.selectByValue("j");
Thread.sleep(10000);

sel.selectByVisibleText("Pencil");*/
List<WebElement> alloptions=sel.getOptions();
int count=alloptions.size();
for(int i=0;i<count;i++)
{
sel.selectByIndex(i);
}
```

Remove Duplicate

```
package com.automation.practice;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;


public class PracSelect {

        public static void main(String[] args) throws InterruptedException
        {
                System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
                WebDriver driver=new ChromeDriver();
                driver.get("file:///C:/Users/PC/Desktop/prac.html");
                WebElement selectobj=driver.findElement(By.id("slv"));
                Select sel=new  Select(selectobj);
        /*Thread.sleep(10000);
                //sel.selectByIndex(1);
                sel.selectByValue("j");
                Thread.sleep(10000);

                sel.selectByVisibleText("Pencil");*/
                List<WebElement> alloptions=sel.getOptions();
                /*int count=alloptions.size();
                for(int i=0;i<count;i++)
                {
                        sel.selectByIndex(i);
                }
```

Continuation

```
ArrayList<String> alltext=new ArrayList();
for (WebElement option:alloptions)
{String text=option.getText();
alltext.add(text);
}

HashSet<String> alltextcopy=new HashSet(alltext);
if(alltext.size()==alltextcopy.size())
                                        {
                    System.out.println("duplicate present");
                                        }


else
{
                    System.out.println("no duplicate");
}
```

Actions

Actions

- Actions is a class which implements Action interface
- In selenium, Actions class is use to handle mouse and keyboard action.

1. Move to Element
2. Double click
3. Drag and Drop
4. Context click

```
Actions act=new Actions(driver);
Thread.sleep(10000);
WebElement Admin=driver.findElement(By.xpath("//b[text()='Admin']"));
act.moveToElement(Admin).perform();
act.doubleClick().perform();
```

CONTEXT CLICK

```
driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("Admin");
driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin123");
driver.findElement(By.xpath("//input[@class='button']")).click();
WebElement image=driver.findElement(By.xpath("//img[@alt='OrangeHRM']"));
Actions act=new Actions(driver);
	act.contextClick(image).perform();
```

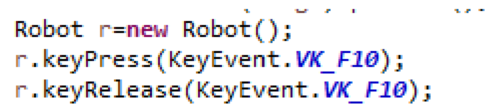
ROBOT CLASS

Robot class is used to (generate native system input events) take the control of mouse and keyboard. Once you get the control, you can do any type of operation related to mouse and keyboard through with java code.
There are different methods which robot class uses. Here in the below example we have used *'keyPress'* and *'keyRelease'* methods.
keyPress - takes keyCode as Parameter and Presses here a given key.
keyrelease - takes keyCode as Parameterand Releases a given key
Both the above methods *Throws - IllegalArgumentException, if keycode is not a valid key.*

EXAMPLE-1:

```
Robot robot = new Robot();


robot.keyPress(KeyEvent.VK_CONTROL);

robot.keyPress(KeyEvent.VK_V);

robot.keyRelease(KeyEvent.VK_V);

robot.keyRelease(KeyEvent.VK_CONTROL);

robot.keyPress(KeyEvent.VK_ENTER);

robot.keyRelease(KeyEvent.VK_ENTER);
```

EXAMPLE-2:



```
Robot r=new Robot();
r.keyPress(KeyEvent.VK_F10);
r.keyRelease(KeyEvent.VK_F10);
```

COMPOSITE ACTION

To perform a **composite action**, the click() method can also be used in combination with some other methods like moveToElement(), or moveByOffset(). It is also used to perform an **action** independently.

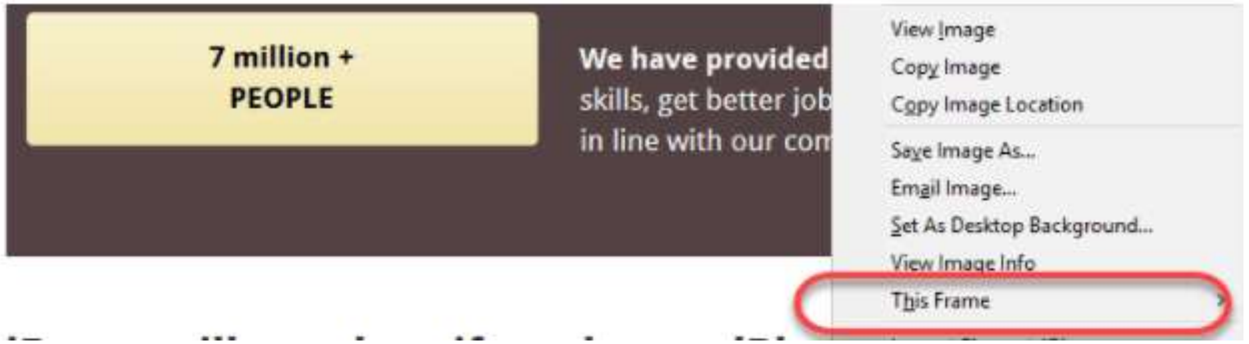
Syntax:

```
actions.moveToElement(element).click();.
```

```
Syntax:
    actions.moveToElement(element).doubleClick().perform();
```

FRAME

- We cannot detect the frames by just seeing the page or by inspecting.
- Right click on the element, If you find the option like 'This Frame' then it is an iframe.(Please refer the diagram)
- 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.



We can even identify total number of iframes by using below snippet.

```
Int size = driver.findElements(By.tagName("iframe")).size();
```

How to Switch Frame:

Basically, we can switch over the elements and handle frames in Selenium using 3 ways.

By Index
By Name or Id
By Web Element

- **Switch to the frame by index:**

```
driver.switchTo().frame(0);
driver.switchTo().frame(1);
```

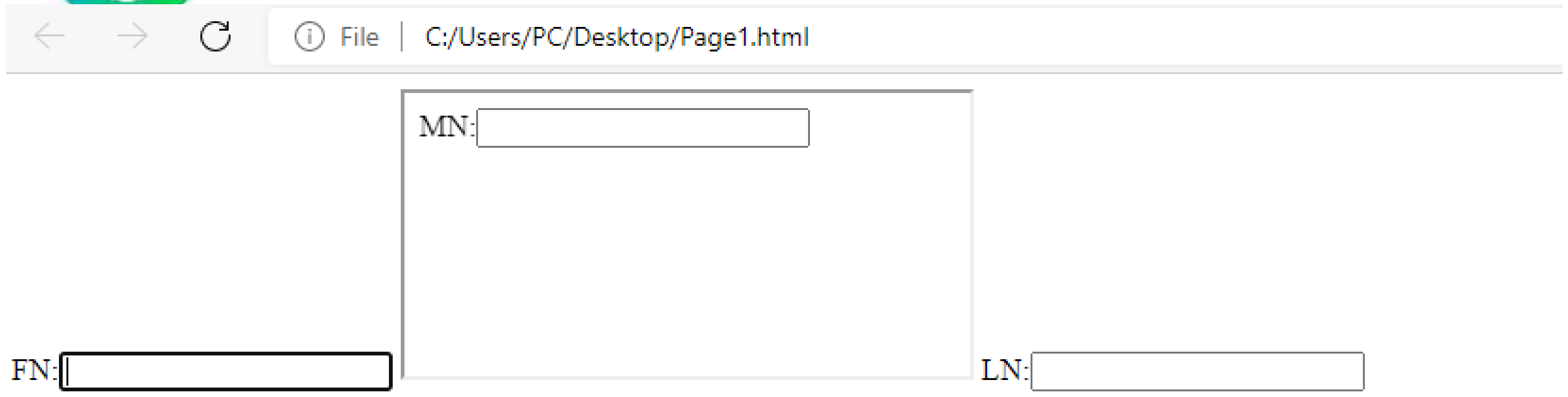
- **Switch to the frame by Name or ID:**

```
driver.switchTo().frame("iframe1");
driver.switchTo().frame("id of the element");
```

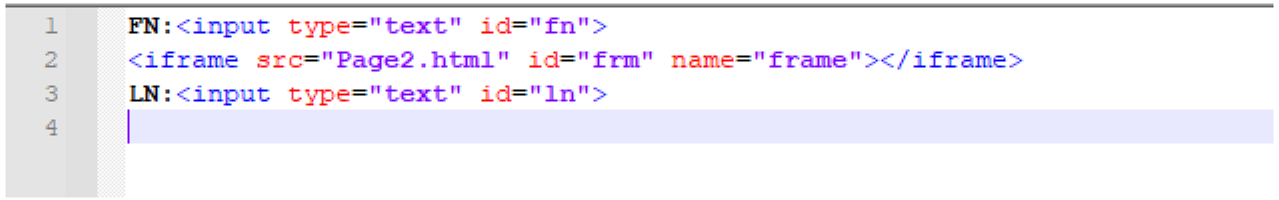
- **Switch to the frame by Web Element:**

```
driver.switchTo().frame(WebElement);
```

FRAME -PROGRAM

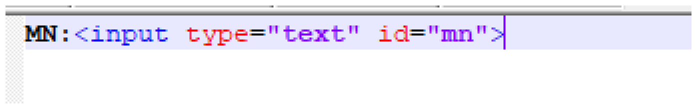


Page1.html



```
FN:<input type="text" id="fn">
<iframe src="Page2.html" id="frm" name="frame"></iframe>
LN:<input type="text" id="ln">
```

Page2.html



```
MN:<input type="text" id="mn">
```

PROGRAM

```
package com.automation.practice;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Frame {
public static void main(String[] args) throws InterruptedException
{
System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-
workspace\\AutomationDemo\\Driver\\chromedriver.exe");
WebDriver driver=new ChromeDriver();
driver.get("file:///C:/Users/PC/Desktop/Page1.html");
driver.findElement(By.id("fn")).sendKeys("Plabani");
driver.switchTo().frame(0);
driver.findElement(By.id("mn")).sendKeys("j");
driver.switchTo().defaultContent();
driver.findElement(By.id("ln")).sendKeys("b");


}


}
```

JAVA SCRIPT POPUP

the simplest **way to handle JavaScript popup/alert** using selenium is by **using the Alert interface**.
To access the popup/alert dialog in Selenium Webdriver, use the following line of code

webDriver.switchTo().alert()

The alert interface provides following methods to handle/interact with such Javascript popups/dialogs:

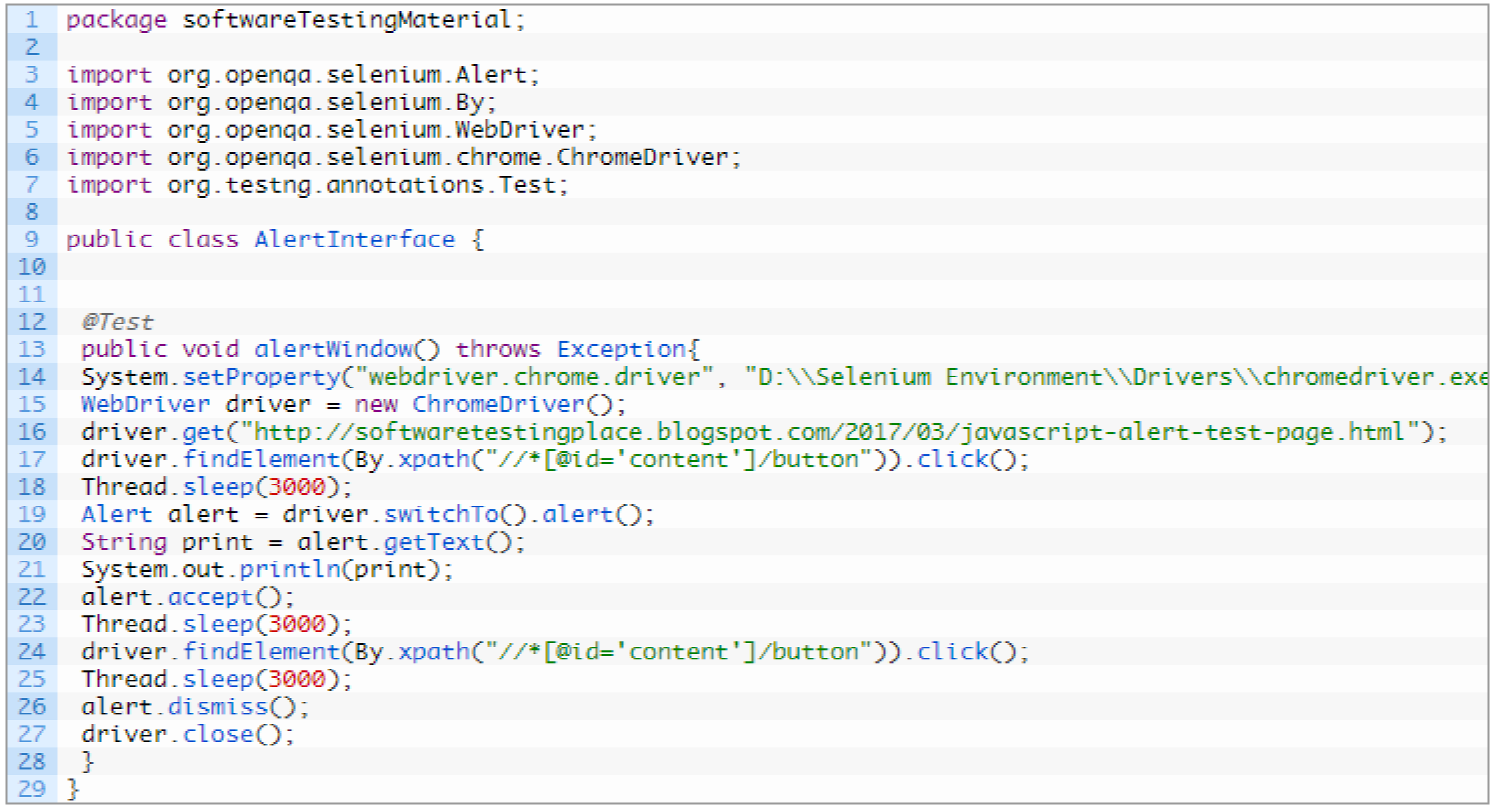
accept(): To accept an popup/alert

dismiss(): To decline an popup/alert

getText(): To get the text written on the popup/alert

sendKeys(keysToEnter): To enter some text on the popup/alert's input box

Example:



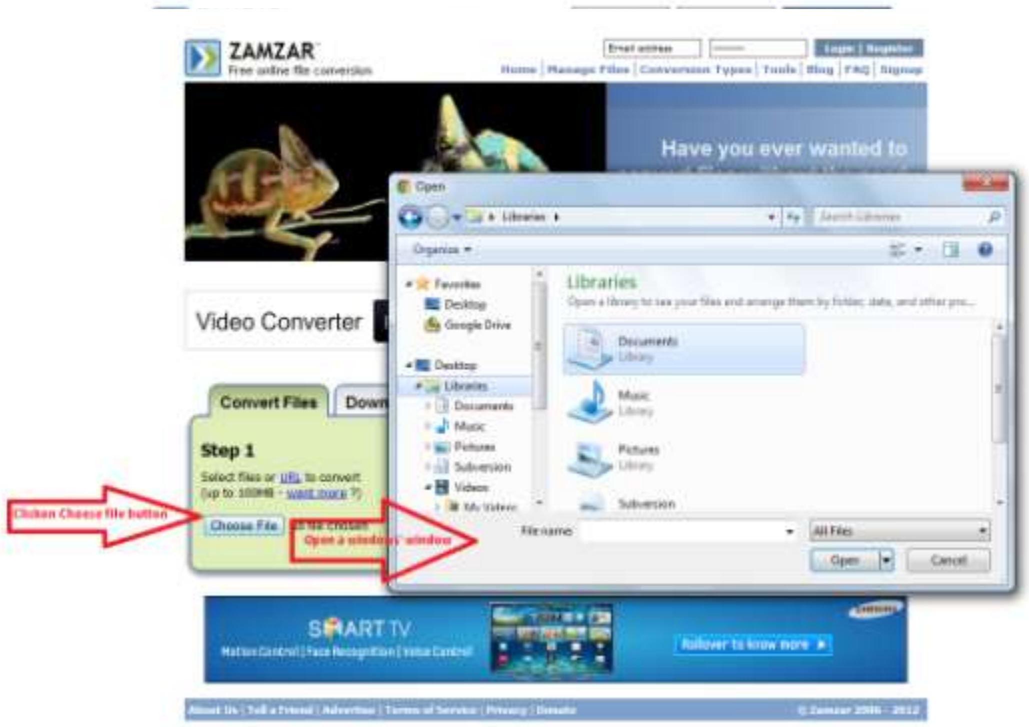
```java
package softwareTestingMaterial;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;

public class AlertInterface {


 @Test
 public void alertWindow() throws Exception{
 System.setProperty("webdriver.chrome.driver", "D:\\Selenium Environment\\Drivers\\chromedriver.exe
 WebDriver driver = new ChromeDriver();
 driver.get("http://softwaretestingplace.blogspot.com/2017/03/javascript-alert-test-page.html");
 driver.findElement(By.xpath("//*[@id='content']/button")).click();
 Thread.sleep(3000);
 Alert alert = driver.switchTo().alert();
 String print = alert.getText();
 System.out.println(print);
 alert.accept();
 Thread.sleep(3000);
 driver.findElement(By.xpath("//*[@id='content']/button")).click();
 Thread.sleep(3000);
 alert.dismiss();
 driver.close();
 }
}
```

FILE UPLOAD POPUP

Uploading files in **WebDriver** is done by simply using the sendKeys() method on the (type=**file)**-select input field to enter the path to the **file** to be **uploaded**

Example:

```
WebElement fileInput = driver.findElement(By.name("uploadfile"));
 fileInput.sendKeys("C:/path/to/file.jpg")
```



File download popup-program

File download possible using Click method & Robot class

Click Method

```
package Demo;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Click_class {

		public static void main(String[] args) throws InterruptedException, AWTException {
			// TODO Auto-generated method stub

			System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
			WebDriver driver=new ChromeDriver();
			driver.get("https://www.selenium.dev/downloads/");
			driver.manage().window().maximize();
			JavascriptExecutor j=(JavascriptExecutor)driver;
			Thread.sleep(2000);
			String Scrolldown="window.scrollBy(0,1000)";
			j.executeScript(Scrolldown);
			String xp="//td[text()='Java']/..//a[text()='Download']";
			//driver.findElement(By.xpath(xp)).click();
```

File Download Using ROBOT CLASS-PROGRAM

```
package Demo;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Robot_class {

        public static void main(String[] args) throws InterruptedException, AWTException {
                // TODO Auto-generated method stub

                System.setProperty("webdriver.chrome.driver", "C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe")
                WebDriver driver=new ChromeDriver();
                driver.get("https://www.selenium.dev/downloads/");
                driver.manage().window().maximize();
                JavascriptExecutor j=(JavascriptExecutor)driver;
                Thread.sleep(2000);
                String Scrolldown="window.scrollBy(0,1000)";
                j.executeScript(Scrolldown);
                Thread.sleep(3000);
                Robot r=new Robot();
                r.keyPress(KeyEvent.VK_ENTER);
                r.keyRelease(KeyEvent.VK_ENTER);
```

What is window handles in selenium

It is a unique identifier that holds the address of all the windows. Think of it as a pointer to a window, which returns the string value.
It is assumed that each browser will have a unique window handle. This window handle function helps to retrieve the handles of all windows.
Syntax
get.windowhandle(): This method helps to get the window handle of the current window. Return type String
get.windowhandles(): This method helps to get the handles of all the windows opened . Return type set<String>

Switch Windows:

Driver.switchTo.window()

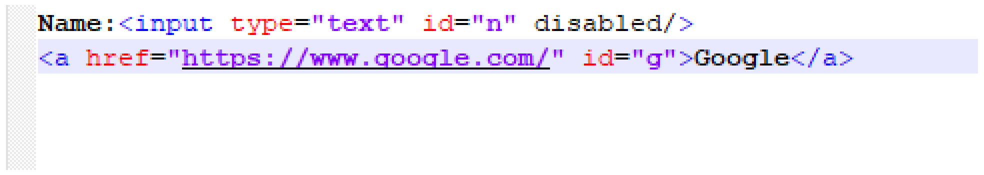
WHAT IS WINDOW HANDLES IN SELENIUM-PROGRAM

```
mport java.util.Iterator;
import java.util.Set;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class WindowHandle_Demo {
public static void main(String[] args) throws Exception {


System.setProperty("webdriver.chrome.driver","Path to the driver");
WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();
// Load the website
driver.get("http://www.naukri.com/");
// It will return the parent window name as a String
String parent=driver.getWindowHandle();
Set<String>s=driver.getWindowHandles();
// Now iterate using Iterator
Iterator<String> I1= s.iterator();
while(I1.hasNext())String child_window=I1.next();
if(!parent.equals(child_window))
{
driver.switchTo().window(child_window);

System.out.println(driver.switchTo().window(child_window).getTitle());

driver.close();
}

}
//switch to the parent window
driver.switchTo().window(parent);

}

}
```

WHAT IS JAVA SCRIPT EXECUTOR

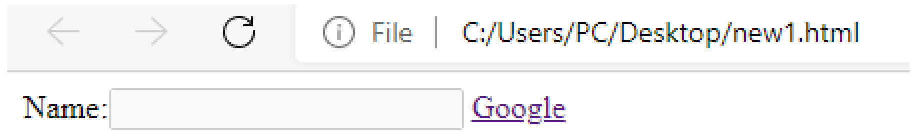
JavaScriptExecutor is an Interface that helps to execute JavaScript through Selenium Webdriver.
JavaScriptExecutor provides two methods "executescript" & "executeAsyncScript" to run javascript on the selected window or current page

HTML code:



```
Name:<input type="text" id="n" disabled/>
<a href="https://www.google.com/" id="g">Google</a>
```

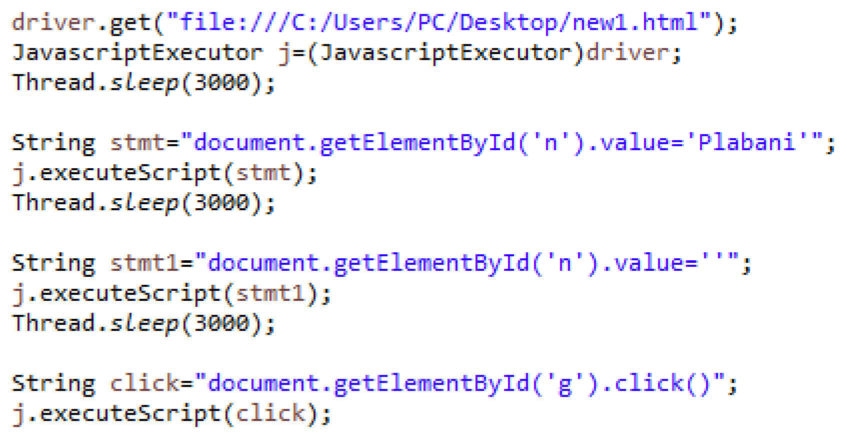
Website



```
File | C:/Users/PC/Desktop/new1.html
```

Name: Google

Program



```
driver.get("file:///C:/Users/PC/Desktop/new1.html");
JavascriptExecutor j=(JavascriptExecutor)driver;
Thread.sleep(3000);

String stmt="document.getElementById('n').value='Plabani'";
j.executeScript(stmt);
Thread.sleep(3000);

String stmt1="document.getElementById('n').value=''";
j.executeScript(stmt1);
Thread.sleep(3000);

String click="document.getElementById('g').click()";
j.executeScript(click);
```

Page Object Model

- **Page Object Model**, also known as **POM**, is a design **pattern** in Selenium that creates an **object** repository for storing all web elements.

It is useful in reducing code duplication and improves test case maintenance. In **Page Object Model**, consider each web **page** of an application as a class file.

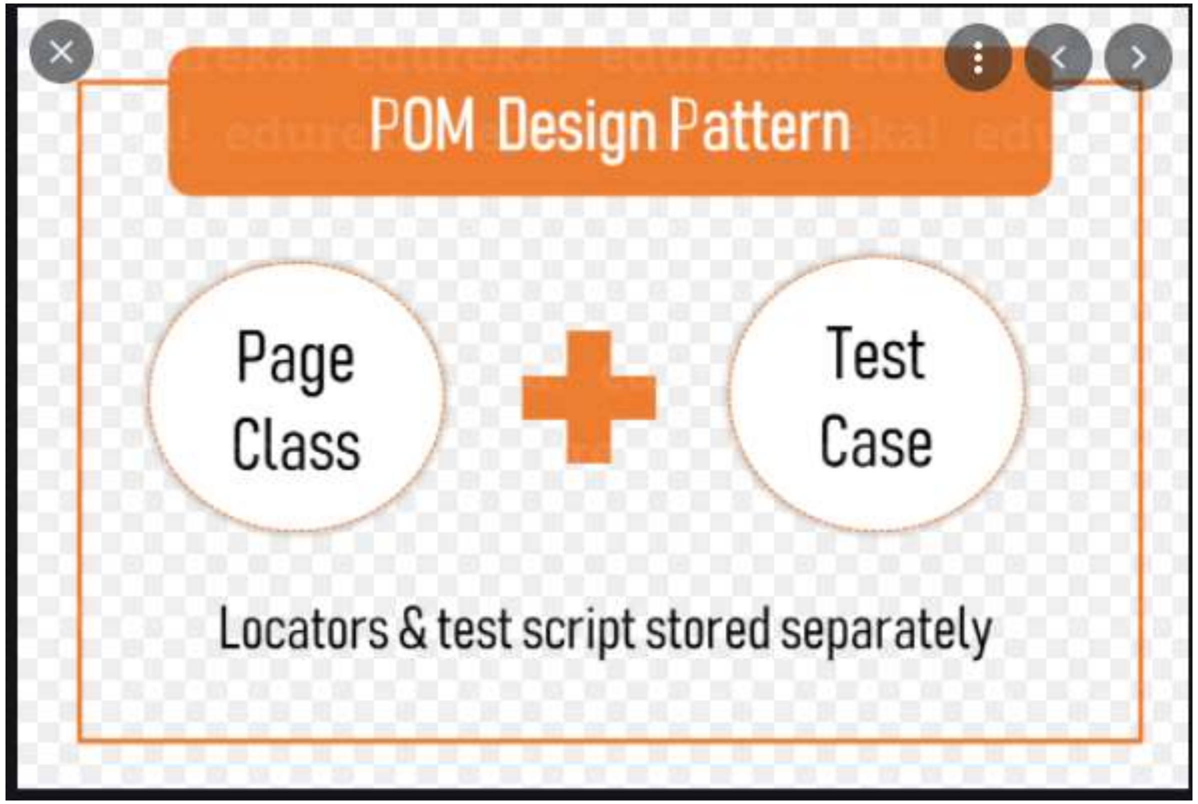
- How do we Declare element is POM Class?

→ By using @FindBy annotation

- How do we initialize the element in POM class?

→PageFactory.initElement(driver,this)

Page Object Model call it as Object Repository



BSS

PROGRAM-POM

Page Object Model:

```
package com.automation.practice;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class Page_Object_Model {

		@FindBy(id="txtUsername")
		private WebElement unTB;
		@FindBy(id="txtPassword")
		private WebElement pwTB;
		@FindBy(id="btnLogin")
		private WebElement btnLogin;

		public Page_Object_Model(WebDriver driver)
		{
			PageFactory.initElements(driver, this);
		}
		public void enterUsernme(String un)
		{
			unTB.sendKeys(un);
		}
		public void enterPassword(String pw)
		{
			pwTB.sendKeys(pw);
		}
		public void clickLogin()
		{
			btnLogin.click();
		}

}
```

Test Class

```
package com.automation.practice;

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

public class Page_object_model_main {

		public static void main(String[] args) {
			// TODO Auto-generated method stub
			System.setProperty("webdriver.chrome.driver",
	"C:\\Users\\PC\\eclipse-workspace\\AutomationDemo\\Driver\\chromedriver.exe");
			WebDriver driver=new ChromeDriver();
			driver.get("https://opensource-demo.orangehrmlive.com/");
			Page_Object_Model page=new Page_Object_Model(driver);
			page.enterUsernme("Admin");
			page.enterPassword("admin23");
			page.clickLogin();

		}

}
```

COX

ADVANTAGES-POM CLASS

- It makes ease in maintaining the code (flow in the UI is separated from verification)
- Makes code readable (Methods get more realistic names)
- Makes the code reusable (object repository is independent of test cases)
- The Code becomes less and optimised
- All the actions should be done in POM class
- All the verification should be done in TestClass