Selenium

What is Automation?

Doing any task using a tool or system without manual intervention is called Automation . You can also call it as replication of human effort.

Advantages of Automation.

* It is faster.
* Saves Time
* Reduces effort.
* Increases the Quality.

Disadvantages of Automation.

* Initial investment is high.
* Required Skilled Man power.

How we can achieve Automation?

We can achieve automation with the help of some tool and software like QTP,SELENIUM.

Why we will choose SELENIUM?

* **Language and Framework Support**
* **Open Source Availability**
* **Multi-Browser Support**
* **Support Across Various Operating Systems**
* **Ease Of Implementation**

What is Selenium?

**Selenium is free(open Source) testing suite containing tools each with different approach for test automation of web application.**

**History of Selenium.**

Selenium core –JavaScript on Notepad.

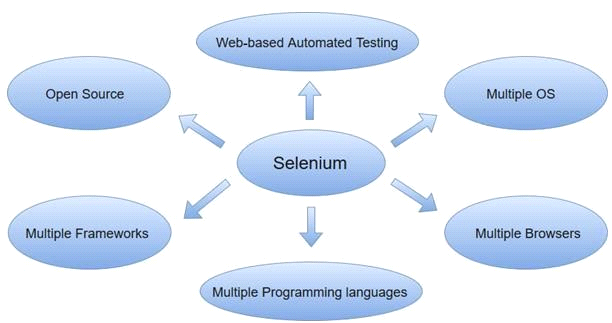
Selenium IDE-No Need to have programming language with record and play feature and Firefox add on.

Selenium Remote control-it is basically for compatibility testing.

Selendroid-for android application.

Selenium WebDriver-For Web Server.

Appium-For Both Window and ios.



Advantages of Selenium

1.It Support 14 lag.

2.Free Open Source

3.Access from Any where

Disadvantages of Selenium

1.It does not support window application.

2.We cannot automate captcha and OTP and animation.

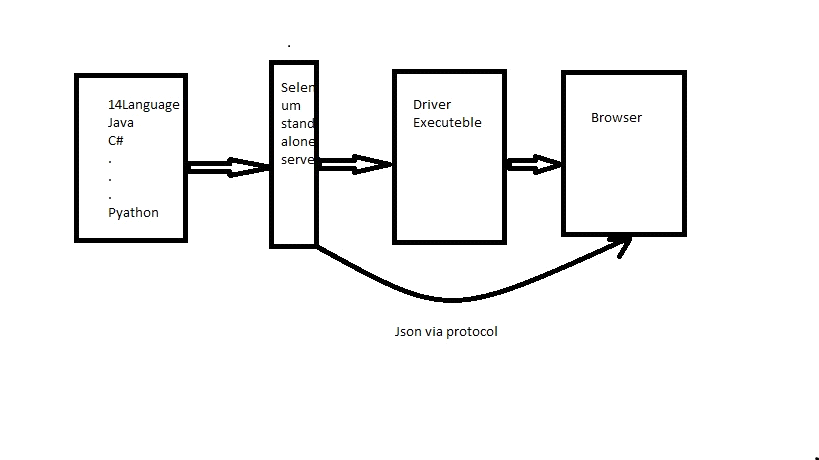
What is OpenSource?

Open source mean that we can download it for free and we can see source code and modification in that source code.

**Selenium Architecture**

Selenium Stand alone server combined with 14 language(java,c#) which is known as language binding or client binding . Selenium Stand alone server perform real actions on

Browser by Driver executable using Json(javaScript object notation) via protocol over http



How to launch a browser?

In order to launch a browser we have to write some set of line in main method.

In order to launch ChromeBrowser.

Public class DemoChromeLaunch{

Public static void main(String[] args){

System.setProperty(“webdriver.chrome.driver”,”path of chromeDriver Exc.”);

ChromeDriver driver = new ChromeDriver();

}

}

In order to launch FireFoxBrowser.

Public class DemoFireFoxLaunch{

Public static void main(String[] args){

System.setProperty(“webdriver.gecko.driver”,”path of geckoDriver Exc.”);

FirefoxDriver driver = new FirefoxDriver();

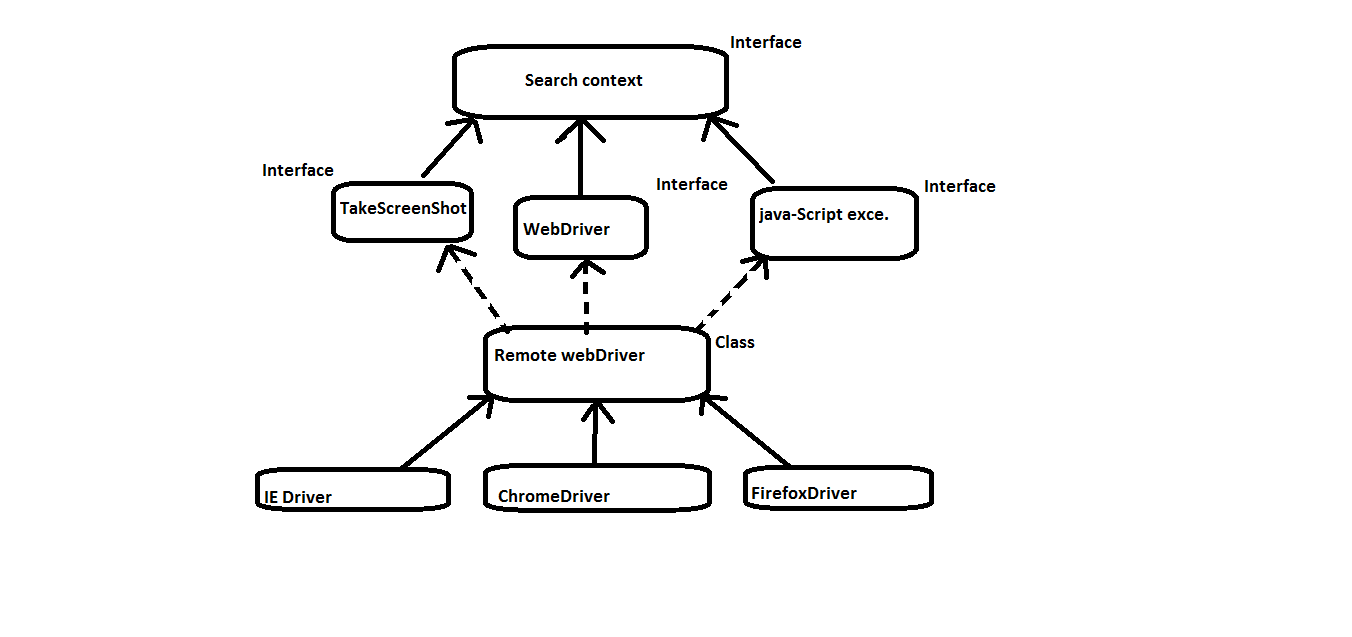
}

}

**Why we are writing WebDriver instead of FirefoxDriver and ChromeDriver?**

**we are performing up-casting over here in order to avoid unnecessary method ,because if we are creating an object of any driver class so it will give me the method which we will not use. And in future if we want to change from one browser to any browser we can change it easily because WEBDRIVER is the parent of all the driver class.**

**Java-Selenium Architecture**



In java selenium architecture super most interface is search context

After than we have three more interface 1.takescreenShot 2.Webdriver 3.javaScriptexec. All are interface so they will have only Abstract method

With no body no implementation ,So we will provide implementation in RemoteWebDriver class and we will use these method in DriverClass.

**METHODS OF WEBDRIVERS.**

* close()
* get()
* getTitle()
* getpagesource()
* getcurrenturl()
* getwindowhandle()
* getwindowhandles ()
* manage()
* navigate ()
* quite ()
* switchTo()

-------------------------------------------------------------------------------------------------------------

All are non Static method so we have call the method with the help of Webdriver object

* close - To close the browser.

Driver.close(). Return type is void.

* get - To load the url.

Driver.get(“String args”). Return type is void.

* gettitle - To get the current page title.

Driver.gettitle() return type is String.

* getpagesource - For current pageSource.

Driver.getpagesource() .Return type is String.

* getcurrentUrl - For current page url.

Driver.getcurrenturl(). Return type is String.

* getwindowhandle - to get parent windowhandle.

Driver.getwindowhandle() .Return type is String.

* getwindowhandles - to get parent and child windowhandle

Driver.getwindowhandles() .Return type is set<String>.

* manage - Return type is option interface

An interface for managing stuff you would do in a browser menu.

Driver.manage();

* navigate - An abstraction allowing the driver to access the browser's history and to navigate to a given

return A {@link org.openqa.selenium.WebDriver.Navigation} that allows the selection of what to do next.

Driver.navigate();

* quit - To close the browser

Driver.quit() - Return type is void.

* switchTo-for switching to frame or window

Driver.switchTo() - Return A TargetLocator which can be used to select a frame or window.

----------------------------------------------------------------------------------------------------------

**Locators**

We can locate the web or application element like Text Field and Button.

Types of Locators.

* ID----1st
* Name—2nd
* ClassName
* TagName —3rd
* LinkText
* PartialLinkText
* Css Selector
* X-path----4th

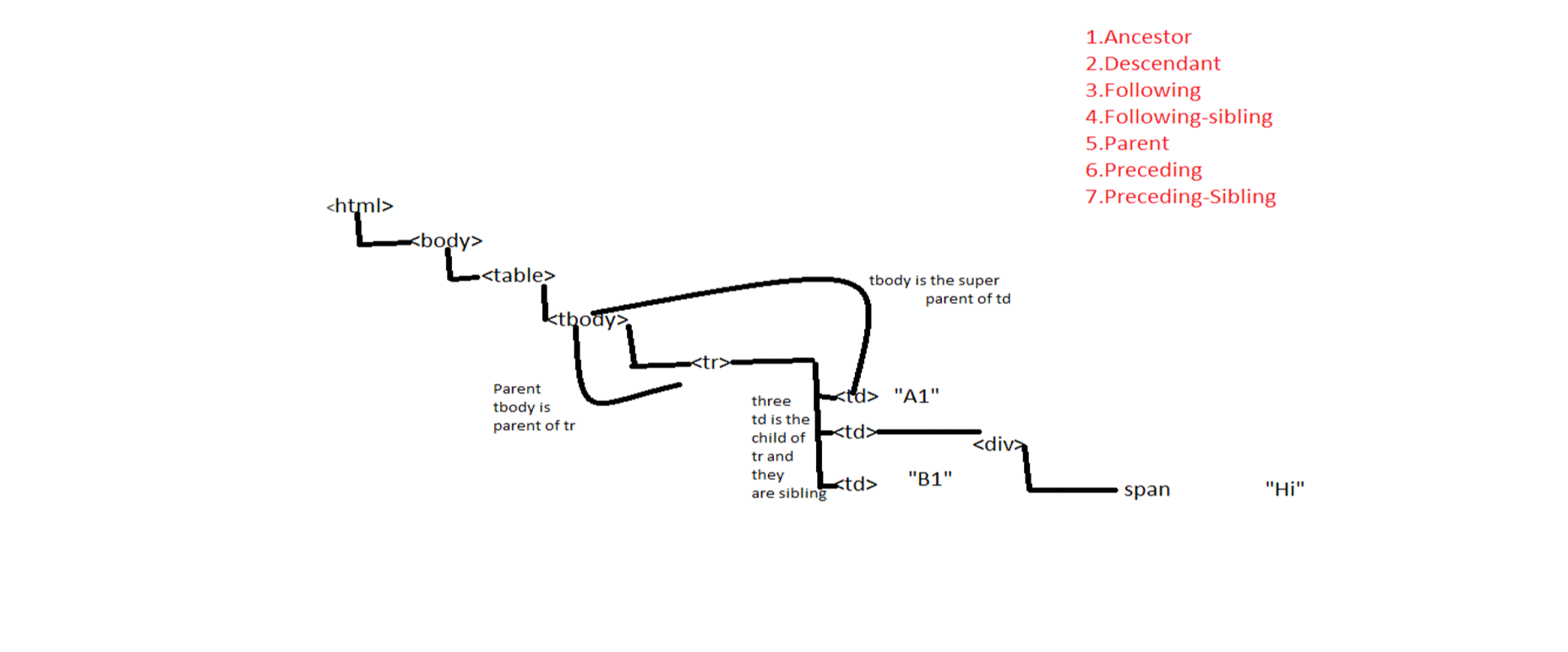
All locators are present in BY Class and they are Static method which will take String as an Argument.

X-PATH

When there are no locator is present then we will go for X-Ptah

There are two type of x-Path.

1. Reletive. (//)

2. Absolute(/)

+++––

Syntax:

//TagName [@AtributeName=’AtributeValue’]

|  |  |
| --- | --- |
| Axis Name | Description |
| 1. Ancestor | Select all the ancestor(parent,grandParent)of the element. |
| 2. Descendant | Select all the descendant(child,grandChild)of the element. |
| 3. Following | Select all the element that follow choosing tag of current element. |
| 4. Following-sibling | Select all the sibling after the current-element. |
| 5. Parent | Select the parent of the current element. |
| 6. Preceding | Select all the element that are before current element. |
| 7. Preceding-Sibling | Select all sibling that are before current element. |

FindElement and FindElements in Selenium WebDriver?

FindElement and FindElements are non-Static method which are present in Search-context .since the Search-Context is the parent of WebDriver so we can call these method with the help of webdriver object .Both method will accept By class as an argument.

List<WebElement> elementName=driver.findElements(By.LocatorStrategy("LocatorValue")) WebElement loginLink = driver.findElement(By.LocatorStrategy("LocatorValue"));

Differences between FindElement and FindElements.

|  |  |
| --- | --- |
| **Find Element** | **Find Elements** |
| Returns the first most web element if there are multiple web elements found with the same locator | Returns a list of web elements |
| Throws exception NoSuchElementException if there are no elements matching the locator strategy | Returns an empty list if there are no web elements matching the locator strategy |
| It will only find one web element | It will find a collection of elements whose match the locator strategy. |
| Not Applicable | Each Web element is indexed with a number starting from 0 just like an array |

**METHODS OF WEBELEMENTS.**

**Webelement is an interface present in selenium.**

* clear ()
* click()
* getAttribute()
* getcssvalue()
* getLocation()
* getText()
* getRect()

8.sendKeys()

* clear - It will clear the text in the webelements.

Return type is void.

* click - It will click on the webelements

Return type is void.

* getAttribute - If the element is currently selected or checked, false otherwise.

It will take String as an argument return type is String.

* getcssvalue() - It will return String and the argument it will take it is also String.
* getLocation() - It will return point class object which will give location of a webelement in the form of x-axis and y-axis.
* getText() - It will return the inner text of a webelement.

Return type is String.

* getRect() - Return type is Rectangle class object.

How to maximize a browser?

To maximize a browser we have to go for method chaining.

driver.manage().window().maximize();

How to load URL?

In Two ways we can load the url

* driver.get(“url”);
* driver.navigate.to(“url”);

Difference Between Get and to?

How to close browser?

In Two ways we can close browser

* driver.close();
* driver.quit();

Difference Between quit and close?

**ACTIONs CLASS**

In order to do action events, you need to use org.openqa.selenium.interactions Actions class. The user-facing API for emulating complex user gestures. Use the selenium actions class rather than using the Keyboard or Mouse directly. This API includes actions such as drag and drop, clicking multiple elements.

Webdriver driver;

Actions a= new Action(driver);

It will take webdriver as an argument .

Webelement ele;

**Methods of Actions class.**

* moveToElement()
* contextClick()
* dragAndDrop() ,dragAndDropBy()
* doubleClick()

For Mouse over Action we have moveToElement(), it will take webelements as an argument.

a.moveToElement(ele).build().perform();

build = The build() method is used to compile all the listed actions into a single step.

Perform = A convenience method for performing the actions without calling build() first

For Right Click we have contextClick(), it will take webelement as an argument.

a.contextClick(ele).build().perform();

For Drage and Drop we have drageanddropBy it will take three arguments webelement, x and y axis.

a.dragAndDropBy(ele, x, y).build().perform();

For Double Click we have doubleClick(), it will take webelement as an argument.

a.doubleClick(ele).build().perform();

**ROBOT CLASS**

To perform keyboard Actions we go for Robot Class like open new tab, new window.

it is coming from java.awt.

**Robot r = new Robot();**

**r.keyPress(KeyEvent);**

**r.keyRelease(KeyEvent);**

KeyPress and KeyRelease will ask for int kye code as an argument for that we will pass KeyEvent with static member of KeyEvent we will pass to get keyCode.

**Select Class.**

Any tag which is developed using Select (<Select></select>) to perform some action on that we go for Select class.

Select s= new Select(Webelement);

We can select the element by index,value,visible text.

**s.selectByIndex(int i);** //it will take int as an argument .

**s.selectByValue(String str);** //it will take String as an argument.

**s.selectByVisibleText(String str);//** it will ask for inner text in the form of String argument

To get all option in list box we go for

**s.getOptions();** //it will return List<webelement>.

To get all option we can go for this method

**isMultiple();** // it will return Boolean value.

To select multiple option we can go for

**s.getAllSelectedOptions();** // it will return List<webelement>.

To know which option get selected 1st we go for

**s.getFirstSelectedOption().getText()**; // it will return String.

**How to Switch to frames**.

To Switching to frames we have to go for driver.switchTo().frame();

It is overloaded, we can switch in three ways

1st by index driver.switchTo().frame(0); //where we have to pass index in int.

2nd by name

driver.switchTo().frame(“frame name”); // it will take String as an argument where String is an name of frame .

3rd by Webelement

driver.switchTo().frame(Webelement); //it will take webelement as an argument where webelement is the frame .

**How to get the location of a webelement.**

To get the location of a particular webelement we have to find the webelement and store in to one variable then using that variable we have to call getlocation() method .

Point loc = ele.getLocation();

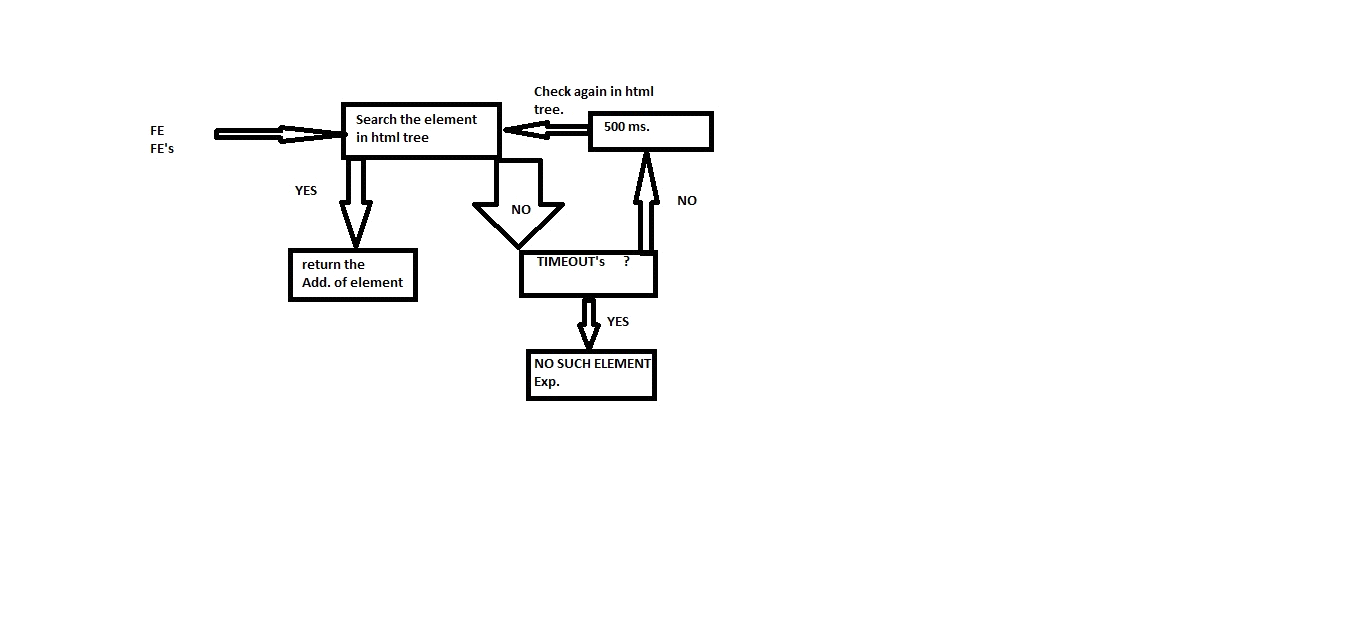
int x = loc.getX(); int y = loc.getY();

System.out.println(x); System.out.println(y);

System.out.println(loc);

**Synchronization.**

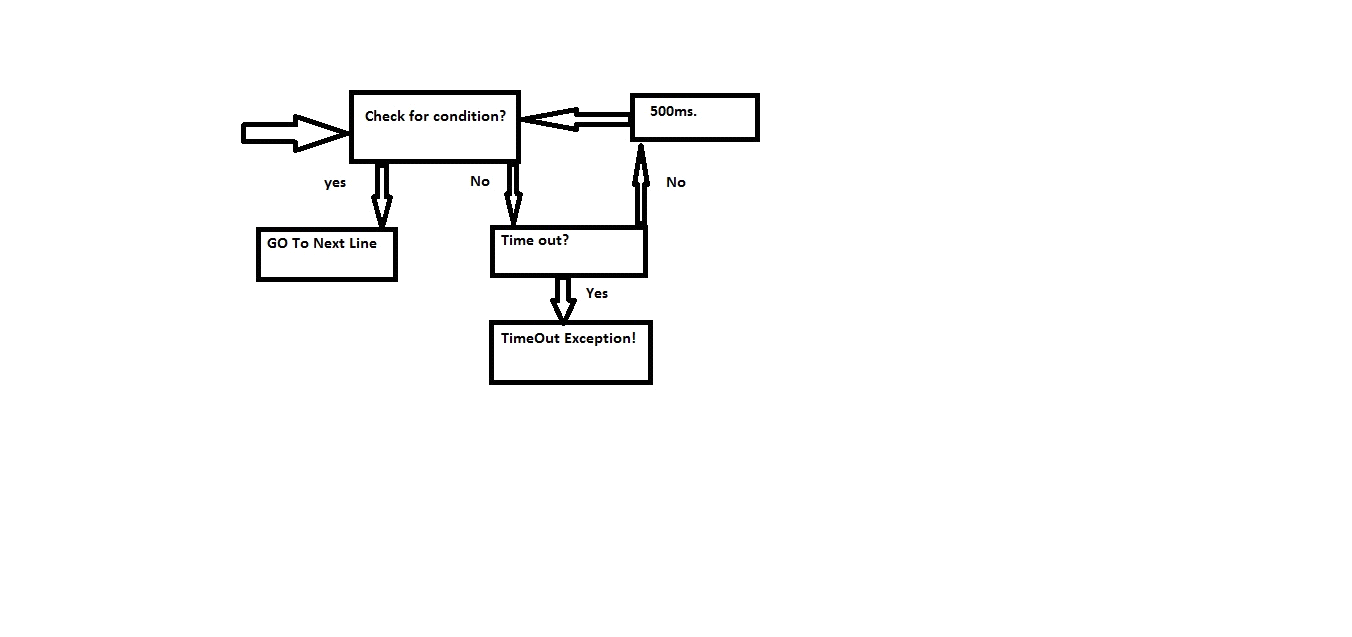
IMPLICIT WAITS



IMPLICIT WAITS will go with FindElement and FindElements, and will check the element in html tree if it find the element then it will return the Address of the webelement, if no then it will check for the time out if time out is over then it will return the NoSuchElementExcp. Else it will check for the element in html tree again After 5ms .

Here time unit will be in days,hr,sce,min,mSec.microSec. driver.manage().timeouts().implicitlyWait(long time,TimeUnit);

Explicit Waits.



Explicit wt. will check for the condition if condition is true the it will go to next line ,if condition is not true then it will go and check for time out if time out is over then it will give timeout Excp. Else after 500ms. It will go and check the condition again.

WebDriverWait ww = new WebDriverWait(driver, 15); ww.until(ExpectedCondition (ele));

until is non Static methods which is present in webDriverWait class which will take ExpectedConditions class as an argument.

**Handling PopUps.**

We have four type of popups.

1.Window popup

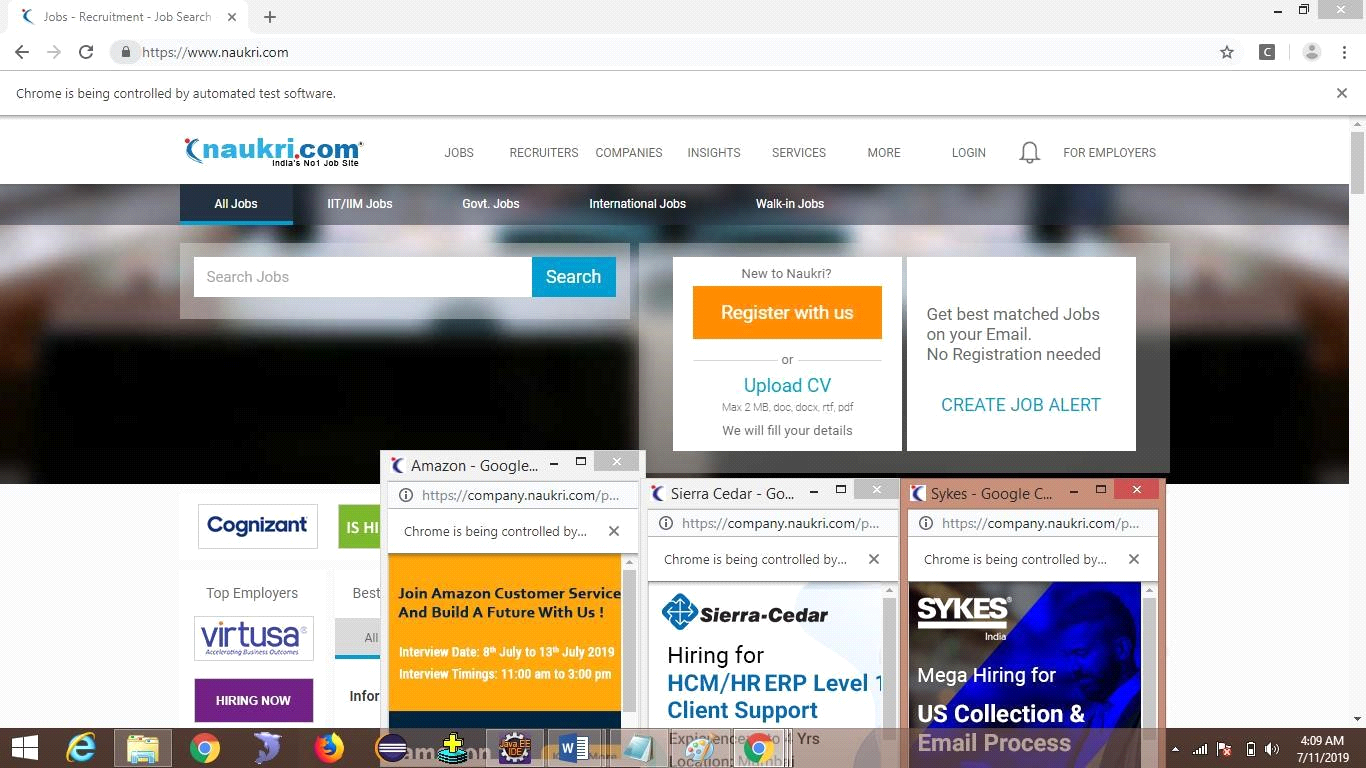
2.Notification popup

3.Alert popup

4.upload popup

Window popup

We can handle window popup by using getwindowhandle() and getwindowhandles() both are present in webdriver.



With parent browser 3 child browser also came.

Code to close only child browser

driver.get("https://www.naukri.com/");

String parent = driver.getWindowHandle();

System.out.println(parent);

Set<String> Win = driver.getWindowHandles(); Win.remove(parent); for (String CWin : Win) { Thread.sleep(2000); System.out.println(CWin); driver.switchTo().window(CWin);

driver.close();

}

Code to close browser in revers order driver.get("https://www.naukri.com/");

String parent = driver.getWindowHandle();

System.out.println(parent);

Set<String> Win = driver.getWindowHandles();

ArrayList<String> a = new ArrayList<>(Win); System.out.println(a);

for (int i = a.size() - 1; i >= 0; i--) { driver.switchTo().window(a.get(i)); driver.close();

}

}

Since set is not index based so we have to convert set to ArrayList the we have to close it.

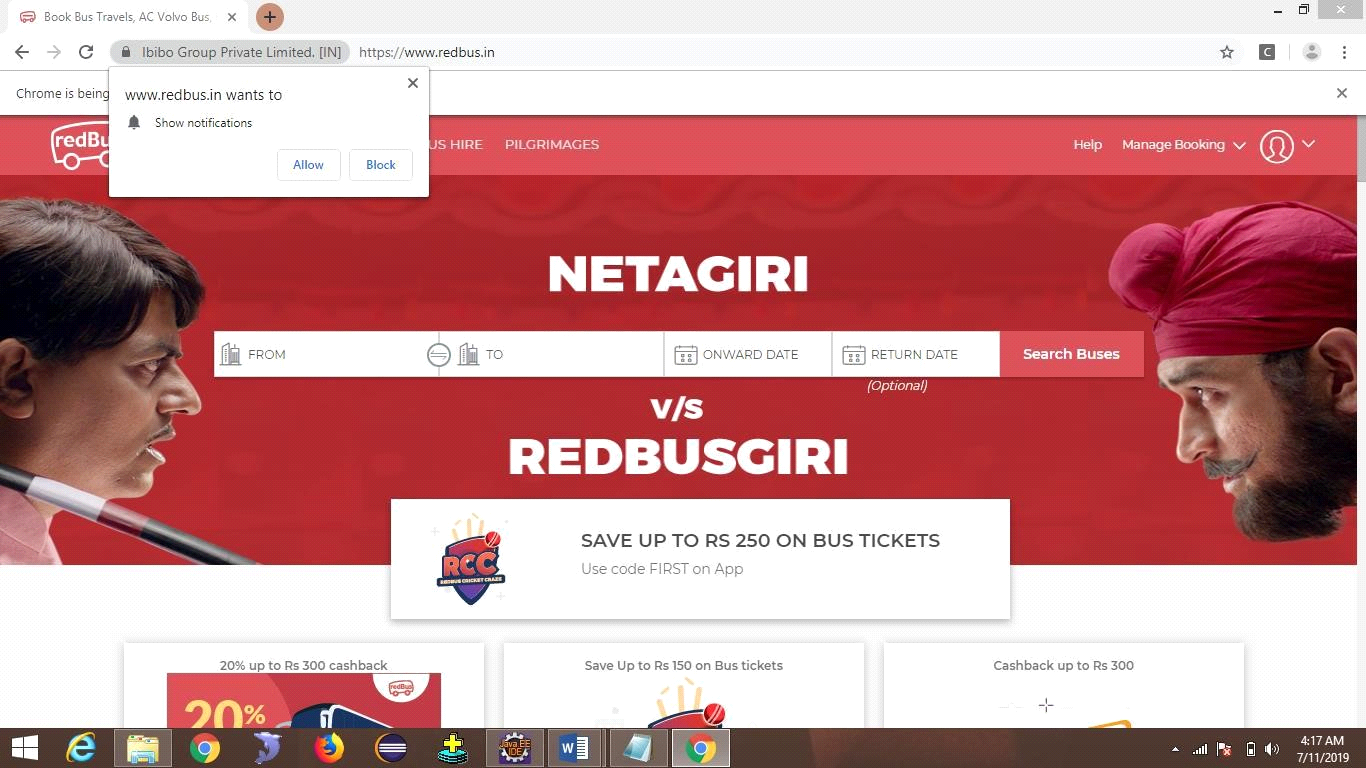
**Notification popup**

Code to close notification popup.

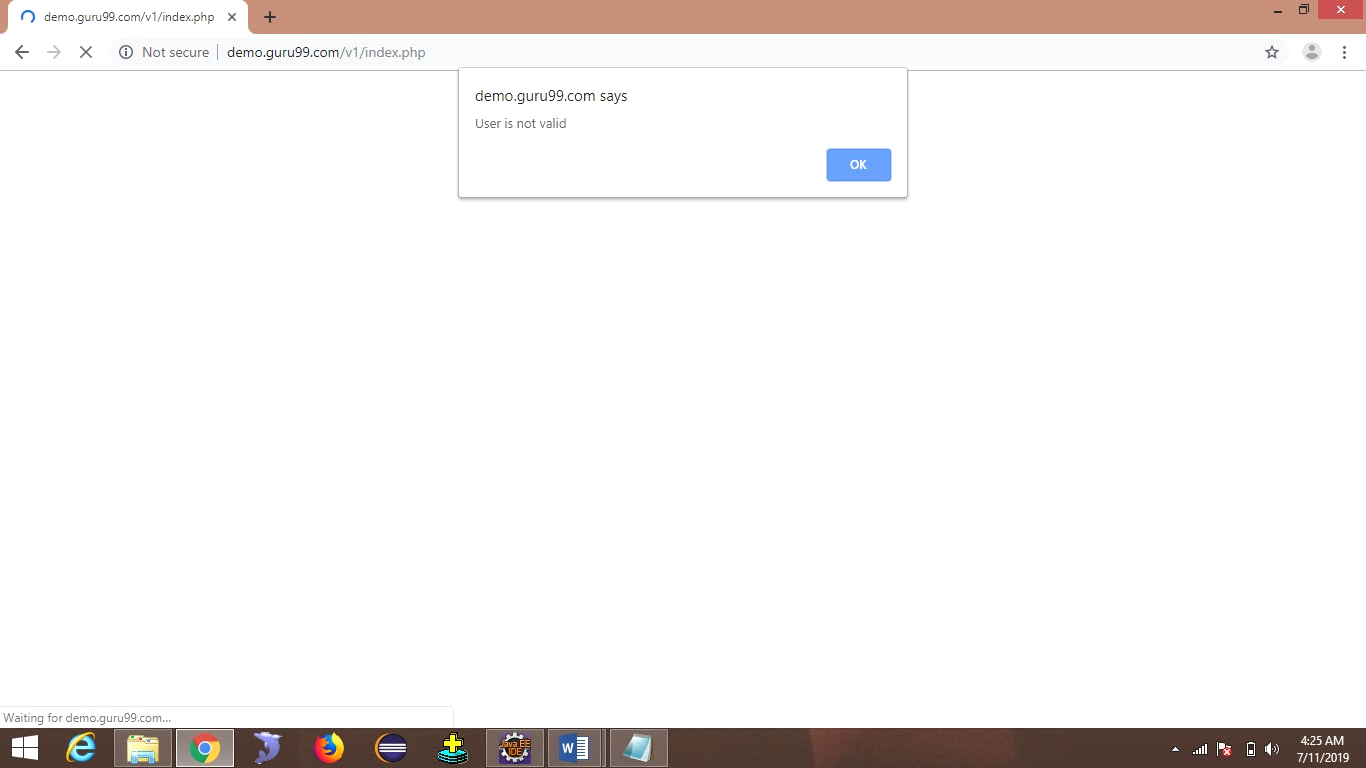
ChromeOptions co = new ChromeOptions();

co.addArguments("--disable-notifications"); WebDriver driver = new ChromeDriver(co);

to handle notification popup we have to make the object of chromeOption class and we have to add the argument and than we have to pass the object of chromeOption in chromeDriver().



**Alert popup**



To handle the alert popup we have to switch to the Alert ,using driver.switchTo().alert(); it will return Alert class object.

Alert a = driver.switchTo().alert();

It will come with four options

a.accept()//to accept it

a.dismiss()//to reject it

a.sendKeys()//to send some data.

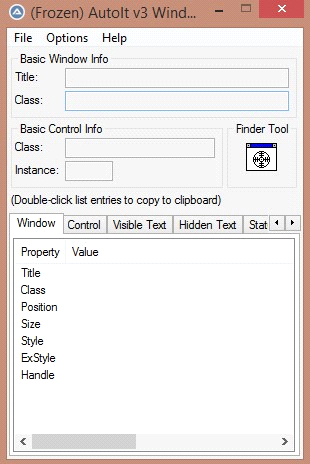
a.getText()//to fetch the text

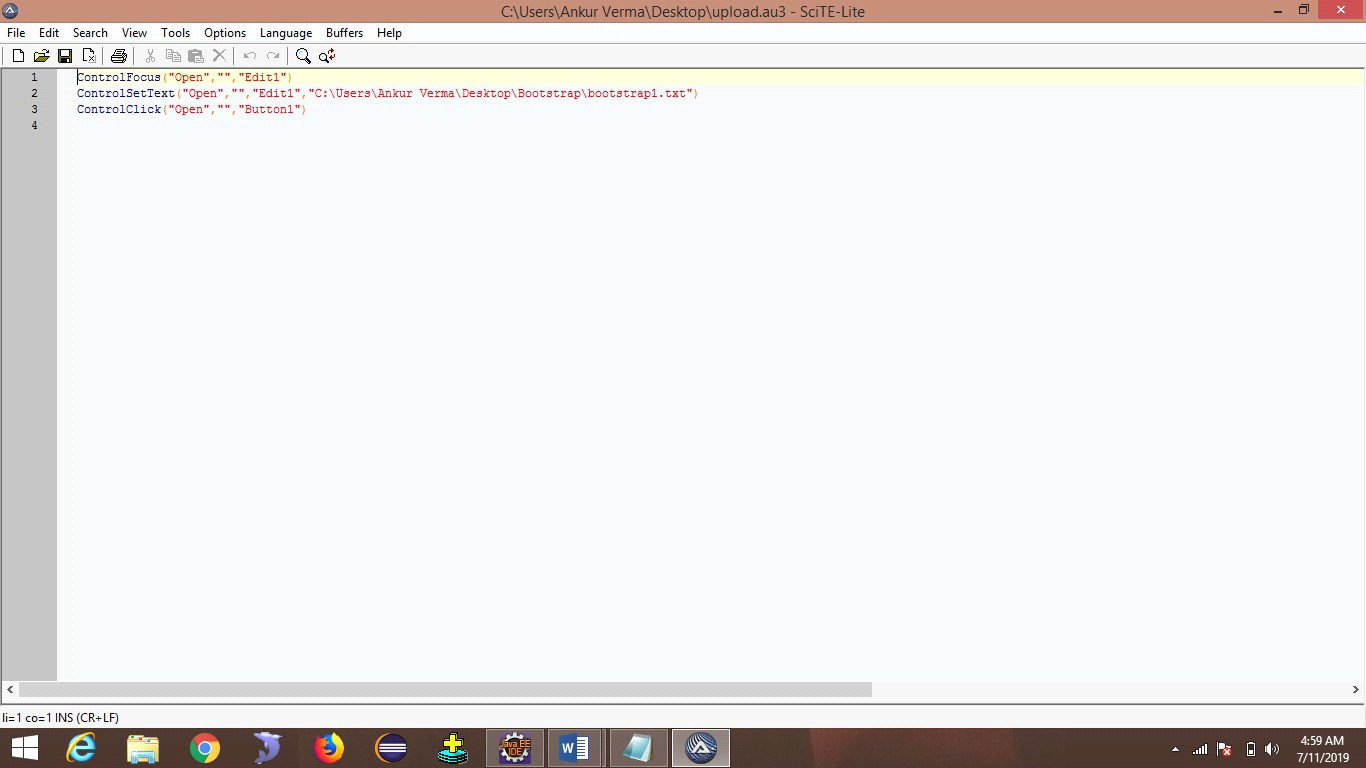
**Upload popup**

To handle this kind of popup we have to take help of AutoIt we have to install this software.

We have to move the finder tool to the path and the open button and we have to take the id and title.

Then we have to write the AutoIt Script.





ControlFocus("Open","","Edit1")

ControlSetText("Open","","Edit1","C:\Users\Ankur Verma\Desktop\Bootstrap\bootstrap1.txt")

ControlClick("Open","","Button1")

The we have to compile the script and we will get Exe file we have to mention that path in script. driver.get("http://demo.guru99.com/test/upload/");

WebElement upload = driver.findElement(By.id("uploadfile\_0")); upload.click();

Runtime.getRuntime().exec("C:\\Users\\AnkurVerma\\Desktop\\uplod.exe");

**Take Screen Shot.**

To take the screenshot we have to cast the driver to takescreenShot and get the use of getScreenshotAs() it will take OutputType.File, outputtype is an Interface and File is static final member .return type of this method is File class object

Then we have to make the object of File class and we have to pass the path of location where we want to save our screenshot, along with name and extension.

And we have make the use of Files class which is present in google.common.io.file packages and we have to use static method called copy which will take two argument from file and to file to copy img.

To get the screenshot refresh the folder.

TakesScreenshot ts = (TakesScreenshot) driver;

File ScreenS = ts.getScreenshotAs(OutputType.FILE);

File ss = new File("loc”+ name + ".png");

Files.copy(ScreenS, ss);



**Java Script Executor**.

To perform the scroll operation we have to go for Java Script Executor.

Scroll operation –like scroll-up, scroll-down, scroll-left, scroll-right.

To make use of java-script-executor we have to cast the webdriver object to javaScriptExecutor.

**WebDriver driver= new ChromeDriver();**

**JavascriptExecutor js = (JavascriptExecutor) driver;**

In javaScriptExecutor we have one method called executeScript() in side this we have to write the script.

**scroll-Down js.executeScript("window.scrollBy(0,1000)");**

In this we have to write the index up-to where we want to scroll in y-axis and we have to make x-axis 0.

**scroll-up js.executeScript("window.scrollBy(0,-500)");**

If want to go up we have to put –ve axis value in place of y-axis.

**scroll-right**

**js.executeScript("window.scrollBy(1000, 0)");**

In this we have to write the index up-to where we want to scroll in x-axis and we have to make y-axis 0.

**scroll-left**

**js.executeScript("window.scrollBy(-500, 0)");**

If want to go up we have to put –ve axis in place of x-axis.

**scroll-Till a webelement js.executeScript("arguments[0].scrollIntoView()", ele);**

To scroll till a webelement we have store the webelement in variable and we have to pass the variable in the method.

**scroll-fullDown js.executeScript("window.scrollTo(0,document.body.scrollHeight)");**

**scroll-fullright js.executeScript("window.scrollTo(document.body.scrollWidth,0)");**

-------------------------------------------------------

**Read Data From Xml.**

To read the data from xml file we have to make use of poi-jar’s .we have to add those jar to build path. After that we have make the object the fileInputStream and pass the path of xml file, after that we have to call a static method create() which is present in WorkBookFactory class and it will take fileInputStream object as an argument and it will return Workbook interface object.after that we have to call a non static method getSheet which will take String sheet name as an argument and it will return sheet interface object. In Sheet interface we have one method called getRow which will take int as argument from which row we want to read the data,it will return Row interface object.we have to take two for loop to read data from xml because it in the form of Two-D array.

FileInputStream fis = new FileInputStream("./Softwares/read.xlsx");

Workbook w = WorkbookFactory.create(fis);

Sheet sh = w.getSheet("Sheet1"); Row firstRow = sh.getRow(0); int rowcount = sh.getPhysicalNumberOfRows();

for (int i = 0; i < rowcount; i++) { for (int j = 0; j < firstRow.getLastCellNum(); j++) { Cell cl = sh.getRow(i).getCell(j);

System.out.println(cl);

}

}

**Write Data in to xml.**

To write the data from xml file we have to make use of poi-jar’s .we have add those jar to build path. After that we have make the object the fileInputStream and pass the path of xml file, after that we have to call a static method create() which is present in WorkBookFactory class and it will take fileInputStream object as an argument and it will return Workbook interface object.after that we have to call a non static method getSheet which will take String sheet name as an argument and it will return sheet interface object. In Sheet interface we have on method called creatRow which will take int as argument from which row we want to store data and we have to call createcell which will take int as an argument and we have to call setCellValue which will take String as an argument fileOutputStream and we have pass the xml file location ,then we have to call write method which is present in workbook interface and it will take fileOutputStream as an argument .At last we have to close the Workbook using close method.

FileInputStream fis = new FileInputStream("D:\\Urban.xlsx");

Workbook wb = WorkbookFactory.create(fis); wb.getSheet(sheet).

createRow(row).

createCell(col)

.setCellValue(data);

FileOutputStream fio = new FileOutputStream("D:\\Urban.xlsx"); wb.write(fio); wb.close();

**POM (PAGE OBJECT MODEL)**

***Stale Element Reference Exception*** to avoid this exception we go for pom class.

Stale element reference exception means the reference of the element is old.

After finding the element & before performing any action on that element if page is refreshed we get **Stale element reference Exception.**

Pom class is use to test the web pages.in POM class we have @FindBy annotation to find Webelement.

@FindBy(name=”Username”) we can go for any locators inside findBy annotation. In order to initialize all the elements present in specified POM class we go for pageFactory.initElements(webdriver,POMObject); pageFactory.initElements(webdriver,this);

In pom we have to define methods of every webelement which we are performing on that webelement.

public class SeleniumHome {

@FindBy(id = "q") private WebElement search; @FindBy(id = "submit") private WebElement go; public SeleniumHome(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void sendText(String str) { search.sendKeys(str);

}

public void Click() { go.click();

}

public void clear() { search.clear();

}

}

In order to use we have to call every method of pom class.all method are non-static so we have to make object and we have to call it.

System.setProperty("webdriver.chrome.driver", "./Softwares/chromedriver.exe"); WebDriver driver = new ChromeDriver(); driver.manage().window().maximize(); driver.get("https://www.seleniumhq.org/");

Thread.sleep(2000);

SeleniumHome sh = new SeleniumHome(driver); sh.sendText("java");

sh.Click(); Thread.sleep(2000); driver.navigate().back(); Thread.sleep(2000); sh.clear(); Thread.sleep(2000); sh.sendText("selenium");

sh.Click();