**P11PreRequistie: Selenium IDE**

**Automation class**

**Day1:**

**Overview on Automation Testing -Regression Test cases**

**Automation Testing team Process**

**Selenium Automation Tool overview -web testing ,mobile testing**

**Different Types of Automation Tool overview -QTP ,testcomplete,Selenium**

**Framework ?**

**Code Repository(Source code management Tool : SVN,Git hub)**

**Understanding the requirement (userstory)**

**Develop automation script**

**Push the code in to code repository**

**Review the code -code review peer**

**Executing the automation script**

**Reporting bug if any .**

**Day2:**

**Selenium Components:**

**Selenium Components (IDE) -Firefox -Record and playback tool**

**Selenium RC -Selenium Remote control Just explore**

**Selenium WebDriver(interface)- fipola click (placeorder)**

**Selenium Basic Concepts:**

* By CSS ID: *find\_element\_by\_id*
* By CSS class name: *find\_element\_by\_class\_name*
* By name attribute: *find\_element\_by\_name*
* By DOM structure or xpath: *find\_element\_by\_xpath*
* By link text: *find\_element\_by\_link\_text*
* By partial link text: *find\_element\_by\_partial\_link\_text*
* By HTML tag name: *find\_element\_by\_tag\_name*

**Xpath Locator :**

**//tagname[@attribute =’value’]**

**Dom Structure :**

**<form id="loginForm">**

**<input name="name" type="text" value="First Name" />**

**<input name="name" type="text" value="Last Name" />**

**<input name="email" type="text" value="Business Email" />**

**<input name="password" type="password" />**

**XPATH Writing:**

**1.Basic Xpath :**

//tagname [@attribute=’value’]

**2.Contains**

//\*[contains(@type,'sub')]

Xpath=//\*[contains(text(),'here')]

Xpath=//\*[contains(@href,'guru99.com')]

**3.Using And /OR**

Xpath=//\*[@type='submit' or @name='btnReset']

**4.Xpath starts-with**

Id=” message8769″

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

**5.Text Function**

Xpath=//td[text()='UserID']

**6.Xpath Axes methods**

These XPath axes methods are used to find the complex or dynamic elements

**following**

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

**Ancestor:**

The ancestor axis selects all ancestors element (grandparent, parent, etc.) of the current node as shown in the below screen.

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

### Child:

Selects all children elements of the current node

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

**Preceding:**

Select all nodes that come before the current node

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

### Following-sibling:

Select the following siblings of the context node. Siblings are at the same level of the current node as shown in the below screen. It will find the element after the current node.

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

**Parent:**

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

**Decendents:**

In the below expression, it identifies all the element descendants to current element ( ‘Main body surround’ frame element) which means down under the node (child node , grandchild node, etc.).

**package** project;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** Login {

**static** WebDriver *driver*;

**public** **static** **void** launchChromeAndFipola() **throws** InterruptedException

{

System.*setProperty*("webdriver.chrome.driver",System.*getProperty*("user.dir")+"\\src\\test\\resources\\drivers\\chromedriver.exe");

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

//Maximize the browser

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

*driver*.get("https://www.fipola.in/");

Thread.*sleep*(1000);

//Thread.sleep(20000);

//driver.close();

//driver.quit();

//xpath Syntax

//tagname[@attribute='value']

**if**(*driver*.findElement(By.*xpath*("//h1[@id='modal-title-1']")).isDisplayed())

{

System.***out***.print("Choose Delivery location found");

}

*driver*.findElement(By.*id*("DelLocation")).sendKeys("123");

*driver*.findElement(By.*className*("top\_pincode\_select")).click();

*driver*.close();

}

**public** **static** **void** main(String args[])**throws** Exception

{

*launchChromeAndFipola*();

}

}

**Selenium WebDriver and RC Difference:**

## **1.Testing Mobile Applications:**Selenium WebDriver supports OS (Operating System) for mobile applications like iOS, windows mobile and android. On the other hand, Selenium RC doesn't support testing of mobile applications.

2. Selenium WebDriver is purely object oriented API, whereas Selenium RC is less object oriented API.

WebDriver is entirely based on object oriented programming languages like Java, C#, etc.

## 3.Speed

Selenium WebDriver performs faster than Selenium RC because it interacts directly with the browser without using any external proxy server. Selenium RC, on the other hand uses an intermediate RC Server to communicate with the browser.

Execution of test scripts takes more time in Selenium RC than WebDriver, since it uses JavaScript commands as instructions to the browser.

**Alert Handling in Selenium**

1.Dismiss the alert

driver.switchTo().alert().dismiss();

2.Accepts the alert

driver.switchTo().alert().accept()

3.Get text from alert

driver.switchTo().alert().getText();

4.Sendkeys to the alert box

driver.switchTo().alert().sendKeys("Text");

Example for student to practice:

<https://www.testandquiz.com/selenium/testing.html>

**Selenium WebBrowser Commands**

**1.get –** command to load webpage in browser

**2.getTitle() -** fetches the title of the current web page

**3. driver.getCurrentUrl();**

**4.driver.getPageSource()-** returns the source code of the current web page loaded on the current browser.

**5.driver.close()**

**DropDown handling in Selenium**

**Select class is used to select/deselect values from dropdown**

**Select sel=new Select(driver.findElement(By.xpath(“”));**

**Sel.selectByIndex(0)-Select the item based on index**

**Sel.selectByValue(“”)-based on option “value” attribute**

**Sel.SelectByVisibleText(“”)- based on text on the element**

**Actions Class in Selenium**

**Action-** build a sequence of composite events and then perform it using Action (an interface which represents a single user-interaction).

* **clickAndHold(WebElement element)** - Clicks a web element at the middle(without releasing).
* **moveToElement(WebElement element)** - Moves the mouse pointer to the middle of the web element without clicking.
* **release(WebElement element)** - Releases the left click (which is in pressed state).
* **build()** - Generates a composite action

//WebElement on which drag and drop operation needs to be performed

WebElement from = driver.findElement(By.id("src1"));

//WebElement to which the above object is dropped

WebElement to = driver.findElement(By.id("destination1");

//Creating object of Actions class to build composite actions

Actions act = **new** Actions(driver);

//Performing the drag and drop action

act.dragAndDrop(from,to).build().perform();

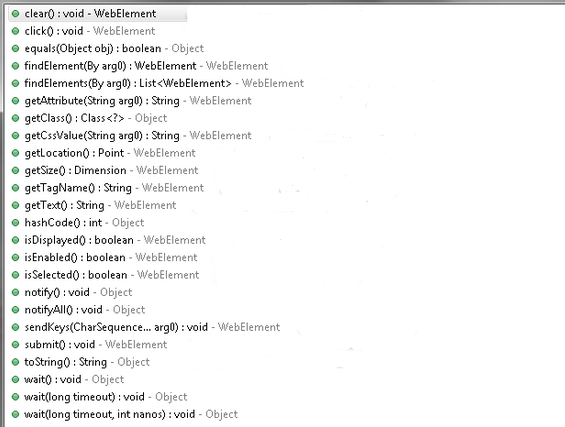
**Navigation Commands**

driver.navigate().to (“”)

driver.navigate().back()

driver.navigate().refresh()

**WebElement Methods:**



**Handling Radio buttons**

driver.findElement(By.xpath("//input[@value='jeck']")).click();

**Handling CheckBox**

We have used two methods in the above code:

**isSelected():** This method determines whether the checkbox is selected or not. If the checkbox is selected, then this method returns true otherwise false.

**click():** This method selects the locator.

        System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());

        driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).click();

        System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());

**Assertions in Selenium**

1.Hard Assertion

2.Soft Asssertion

**Hard Assertion**

**String s=Drievr.findElementBy.path(“//a[text()='Know more about Shopsy']”).getText();**

**Assert.assertEquals(s,”Know more about shopsy”)**

Assert.AssertFalse();

Assert.AssertTrue(condition);

Assert.AssertEquals(actual,expected);

Assert.AssertNotEquals(actual,expected,message);

Assert.AssertNull(Object)

Assert.AssertNotNull(10);

**SoftAsssertion:**

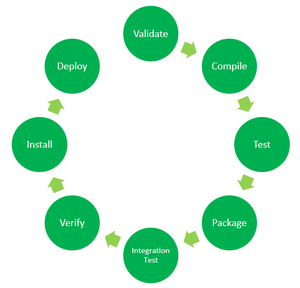
SoftAssertion assertion=new SoftAssertion();

assertion.assertTrue(false);0

assertion.assertAll();

**Maven LifeCyle**

**Maven Lifecycle:**Below is a representation of the default Maven lifecycle and its 8 steps: Validate, Compile, Test, Package, Integration test, Verify, Install and Deploy.



*8 Phases of the Default Maven Lifecycle*

1. **Validate:** This step validates if the project structure is correct. For example – It checks if all the dependencies have been downloaded and are available in the local repository.
2. **Compile:** It compiles the source code, converts the .java files to .class and stores the classes in target/classes folder.
3. **Test:** It runs unit tests for the project.
4. **Package:** This step packages the compiled code in distributable format like JAR or WAR.
5. **Integration test:** It runs the integration tests for the project.
6. **Verify:** This step runs checks to verify that the project is valid and meets the quality standards.
7. **Install:** This step installs the packaged code to the local Maven repository.
8. **Deploy:** It copies the packaged code to the remote repository for sharing it with other developers.

**Maven Commands:**

* **mvn clean:** Cleans the project and removes all files generated by the previous build.
* **mvn compile:** Compiles source code of the project.
* **mvn test-compile:** Compiles the test source code.
* **mvn test:** Runs tests for the project.
* **mvn package:** Creates JAR or WAR file for the project to convert it into a distributable format.
* **mvn install:** Deploys the packaged JAR/ WAR file to the local repository.
* **mvn deploy:** Copies the packaged JAR/ WAR file to the remote repository after compiling, running tests and building the project.

**Selenium Waits**

Most of the web applications are developed using [Ajax](https://www.guru99.com/php-ajax.html) and [Javascript](https://www.guru99.com/interactive-javascript-tutorials.html). When a page is loaded by the browser the elements which we want to interact with may load at different time intervals.

1.Implicit Wait

2.Explicit wait

3.Fluent Wait

**Implicit wait**

The **Implicit Wait in Selenium** is used to tell the web driver to wait for a certain amount of time before it throws a “No Such Element Exception”.

driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);//Synatx

driver.manage().timeouts.implicitlywait(20,TimeUnit.Seconds);

**Example**

System.setProperty ("webdriver.chrome.driver",".\\chromedriver.exe" );

driver = new ChromeDriver();

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

**Explicit Wait**

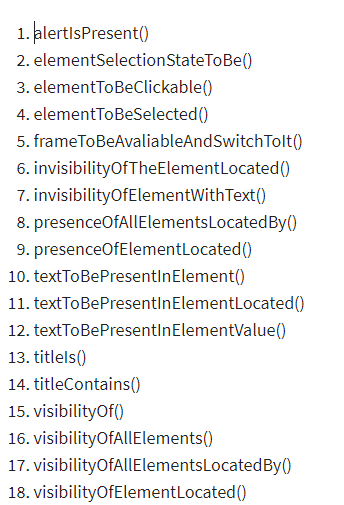
The **Explicit Wait in Selenium** is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or maximum time exceeded before throwing “ElementNotVisibleException” exception. It is an intelligent kind of wait, but it can be applied only for specified elements. It gives better options than implicit wait as it waits for dynamically loaded Ajax elements.

**WebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);**

**Example**

WebDriverWait wait=new WebDriverWait(driver, 20);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( "/html/body/div[1]/section/div[2]/div/div[1]/div/div[1]/div/div/div/div[2]/div[2]/div/div/div/div/div[1]/div/div/a/i")));



**Fluent Wait**

The **Fluent Wait in Selenium** is used to define maximum time for 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. It checks for the web element at regular intervals until the object is found or timeout happens.

Wait wait = new FluentWait(WebDriver reference) .withTimeout(timeout, SECONDS) .pollingEvery(timeout, SECONDS) .ignoring(Exception.class);

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

.withTimeout(30, TimeUnit.SECONDS)

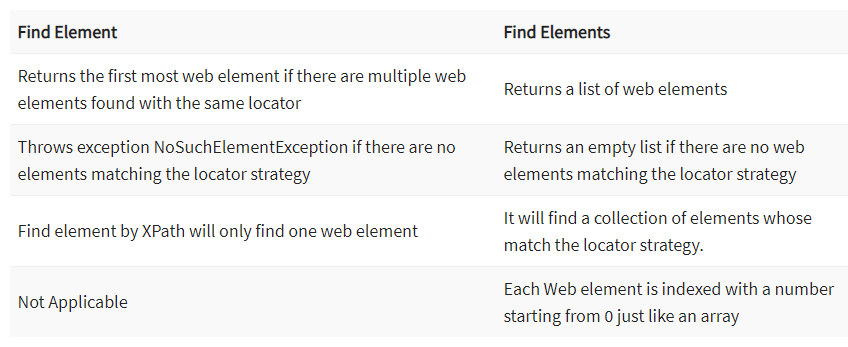
.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);

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

**FindElement &FindElements :**

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



**Relative and Absolute Xpath**

The relative xpath starts by referring to the element that we want to identify and not from the root node

Absolute xpath refer the element from the root node

**Date Picker in Selenium**

Date d = new Date(1);

SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy");

String date = formatter.format(d);

String splitter[] = date.split("-");

String month\_year = splitter[1];

String day = splitter[0];

System.out.println(month\_year);

System.out.println(day);

**Auto Suggestion Dropdown Selenium**

Actions act = new Actions(driver);

act.sendKeys(Keys.DOWN).perform();

act.sendKeys(Keys.ENTER).perform();

**Broken Links in Selenium**

List<WebElement> links = driver.findElements(By.tagName("a"));

For(int i=0;i<links.size;i++)

Iterator<WebElement> it = links.iterator();

while(it.hasNext()){

String url=links.get(i).getAttribute(“href”)

url = it.next().getAttribute("href");

System.out.println(url);

if(url == null || url.isEmpty()){

System.out.println("URL is either not configured for anchor tag or it is empty");

continue;

}

**SelectortsHub Installation**

**Chrome addon which help us to inspect elements ,take xpath and css.**

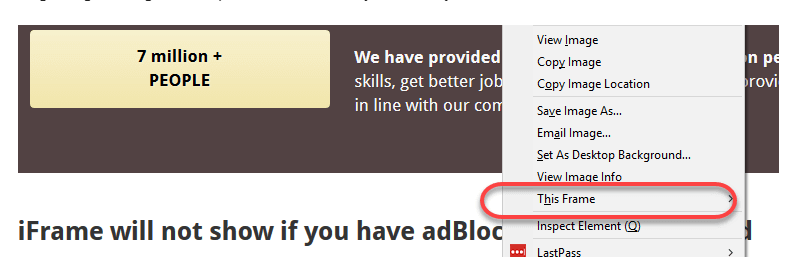
**Find Element based on Text**

**Text()**

**getText()**

**Working with frames in Selenium**

**iFrame in Selenium Webdriver** is a web page or an inline frame which is embedded in another web page or an HTML document embedded inside another HTML document. The iframe is often used to add content from other sources like an advertisement into a web page. The iframe is defined with the <**iframe**> tag.



* 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

**Int size = driver.findElements(By.tagName(“iframe”)).size();**

**Switch over the elements in frame**

**Switch to the frame by index: starts from 0 ->driver.switchTo() .frame(0); -switch to frame**

**Switch over the elements by name:**

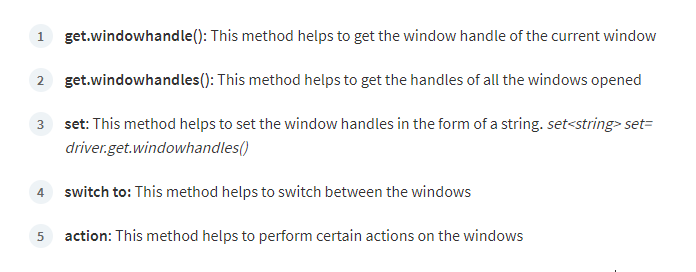
**Driver.switchTo.frame(“name”)**

**Switch over the elements by id**

**Driver.switchTo.frame(“id”)**

**Driver.switchto.deafultContent(); back to old html**

**Window Handling in Selenium**



// 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(); //parent,child

// 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);

**JavaScript Executor in Selenium**

JavaScriptExecutor is an Interface that helps to execute[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)through Selenium Webdriver.

//Creating the JavascriptExecutor interface object by Type casting

WebElement button=driver.findElement(By.xpath(“”));

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

**js.executeScript("arguments[0].click();", button);**

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

**Js.executeScript(window.scrollBy(0,600); -Vertical scroll**

**Screenshots in Selenium**

TakesScreenshot scrShot =((TakesScreenshot)webdriver);

File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);

//Move image file to new destination

File DestFile=new File(fileWithPath); C:\\output.png

//Copy file at destination

FileUtils.copyFile(SrcFile, DestFile);

**Log4J in Selenium**

Log4j is a logging framework written in Java that provides an easy way for logging in Selenium

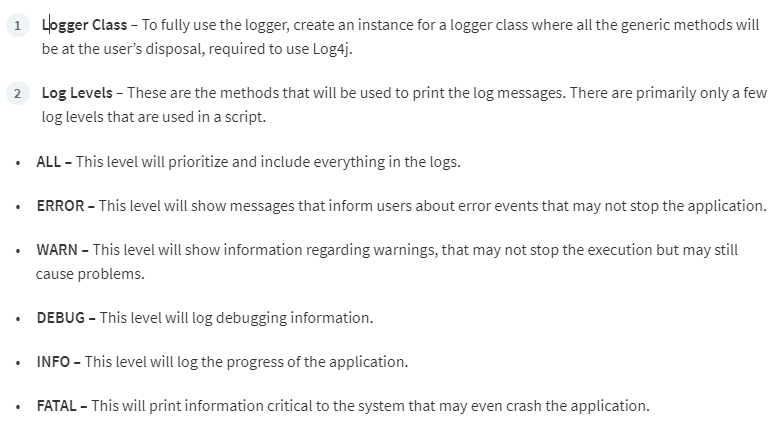
**Components of Log4j**

**Logger**

**Appender**

**Layout**

**Logger:**



public class Myclass {

static Logger log = Logger.getLogger(Myclass.class);

public static void main(String[] args) {

PropertyConfigurator.configure("path\\to\\log4j.properties");

System.setProperty("webdriver.chrome.driver", "Path\\to\\chromedriver.exe");

ChromeOptions options = new ChromeOptions();

options.setUnhandledPromptBehaviour(UnexpectedAlertBehaviour.IGNORE);

WebDriver driver = new ChromeDriver(options);

JavascriptExecutor js = (JavascriptExecutor)driver;

driver.get("https://www.browserstack.com/users/sign\_in");

log.info("Open browserstack");

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

log.info("Maximize window size");

js.executeScript("document.getElementById('user\_email\_login').value='rbc@xyz.com';");

log.info("enter username");

js.executeScript("document.getElementById('user\_password').value='password';");

log.info("enter password");

js.executeScript("document.getElementById('user\_submit').click();");

log.info("click on login");

driver.close();

**Automation Framework:**

Test automation frameworks are **a set of rules and corresponding tools that are used for building test cases**.

Unit testing framework:

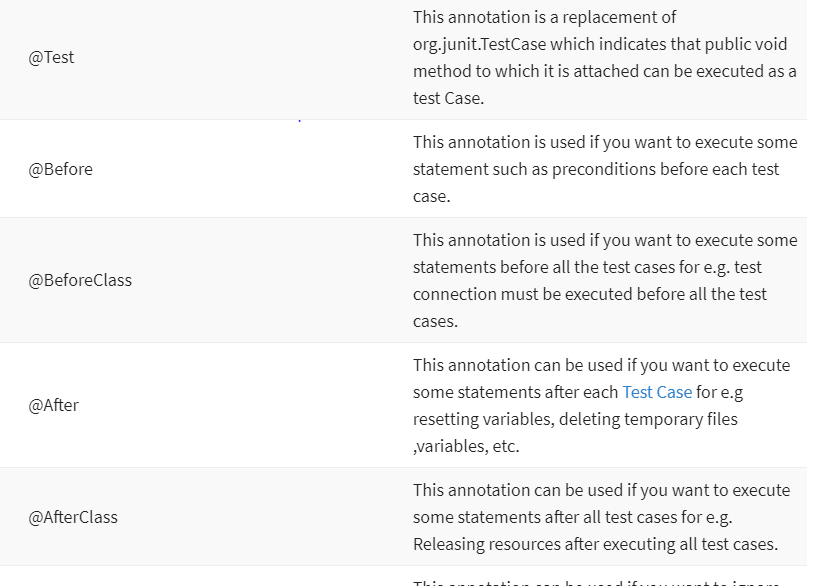
1.Junit

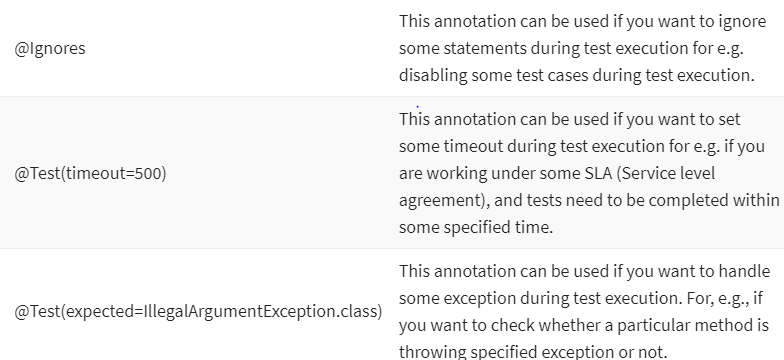
2.TestNG

3.cucumber

**Junit and annotation**

**JUnit Annotations** is a special form of syntactic meta-data that can be added to Java source code for better code readability and structure. Variables, parameters, packages, methods and classes can be annotated.





**Avenstack Extent Report**

**package** test;

**import** java.util.List;

**import** org.junit.Assert;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.interactions.Actions;

**public** **class** BigBasket {

**public** **static** **void** main(String args[]) **throws** InterruptedException

{

//Acrions example

System.***out***.print(System.*getProperty*("user.dir"));

System.*setProperty*("webdriver.chrome.driver",System.*getProperty*("user.dir")+"\\src\\main\\resources\\drivers\\chromedriver.exe");

// trying to avoid notifications - 11/20

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

String expected="box";

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

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

Thread.*sleep*(10000);

List<WebElement> list=driver.findElements(By.*xpath*("//li[@class='hl-cat-nav\_\_js-tab']/a"));

**for**(**int** i=0;i<list.size();i++)

{

System.***out***.print(list.get(i).getText());

list.get(i).click();

driver.findElement(By.*xpath*("//a[text()='"+list.get(i).getText()+"']")).click();

}

Assert.*assertEquals*(expected, "ball");

driver.findElement(By.*xpath*("//input[@placeholder='"+expected+"']")).sendKeys("Chennai");

Thread.*sleep*(4000);

Actions action=**new** Actions(driver);

action.sendKeys(Keys.***DOWN***).sendKeys(Keys.***ENTER***).build().perform();

//new Actions(driver).sendKeys(Keys.DOWN).sendKeys(Keys.ENTER).build().perform();

//svg xpath

///\*[local-name()='svg' and @data-icon='home']/\*[local-name()='path']

//Example- scenario

}

}

**package** test;

**import** java.awt.Window;

**import** java.util.List;

**import** org.junit.Assert;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.interactions.Actions;

**public** **class** BigBasket {

**public** **static** **void** main(String args[]) **throws** InterruptedException

{

//Acrions example

System.***out***.print(System.*getProperty*("user.dir"));

System.*setProperty*("webdriver.chrome.driver",System.*getProperty*("user.dir")+"\\src\\main\\resources\\drivers\\chromedriver.exe");

// trying to avoid notifications - 11/20

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

String expected="box";

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

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

Thread.*sleep*(10000);

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("window.scrollBy(0,850)");

List<WebElement> list=driver.findElements(By.*xpath*("//li[@class='hl-cat-nav\_\_js-tab']/a"));

**for**(**int** i=0;i<list.size();i++)

{

System.***out***.print(list.get(i).getText());

list.get(i).click();

driver.findElement(By.*xpath*("//a[text()='"+list.get(i).getText()+"']")).click();

}

Assert.*assertEquals*(expected, "ball");

driver.findElement(By.*xpath*("//input[@placeholder='"+expected+"']")).sendKeys("Chennai");

Thread.*sleep*(4000);

Actions action=**new** Actions(driver);

action.sendKeys(Keys.***DOWN***).sendKeys(Keys.***ENTER***).build().perform();

//new Actions(driver).sendKeys(Keys.DOWN).sendKeys(Keys.ENTER).build().perform();

//svg xpath

///\*[local-name()='svg' and @data-icon='home']/\*[local-name()='path']

//Example- scenario

}

}

**ExcelReading:**

**package** test;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.xssf.usermodel.XSSFSheet;

**import** org.apache.poi.xssf.usermodel.XSSFWorkbook;

**import** org.junit.Test;

**public** **class** ExcelReading {

@Test

**public** **void** excel() **throws** Exception {

FileInputStream fis=**new** FileInputStream(**new** File(System.*getProperty*("user.dir")+"//TestData.xlsx"));

XSSFWorkbook wb=**new** XSSFWorkbook(fis);

XSSFSheet sheet=wb.getSheetAt(0);

**int** row = sheet.getLastRowNum();

System.***out***.print(row+"Row Count\n");

String s=sheet.getRow(1).getCell(0).getStringCellValue();

System.***out***.print(s);

**int** rowsCount = sheet.getLastRowNum();

System.***out***.println("Total Number of Rows: " + (rowsCount + 1));

**for** (**int** i = 0; i <= rowsCount; i++) {

Row row1 = sheet.getRow(i);

**int** colCounts = row1.getLastCellNum();

System.***out***.println("Total Number of Cols: " + colCounts);

**for** (**int** j = 0; j < colCounts; j++) {

Cell cell = row1.getCell(j);

System.***out***.println("[" + i + "," + j + "]=" + cell.getStringCellValue());

}

}

}

}

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>5.2.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.2.2</version>

</dependency>

**AvenStack ExtentReport**

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.0.3</version>

</dependency>

**package** test;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** com.aventstack.extentreports.ExtentReports;

**import** com.aventstack.extentreports.ExtentTest;

**import** com.aventstack.extentreports.Status;

**import** com.aventstack.extentreports.markuputils.ExtentColor;

**import** com.aventstack.extentreports.markuputils.MarkupHelper;

**import** com.aventstack.extentreports.reporter.ExtentHtmlReporter;

**import** com.aventstack.extentreports.reporter.configuration.ChartLocation;

**import** com.aventstack.extentreports.reporter.configuration.Theme;

**import** junit.framework.~~Assert~~;

**public** **class** Reporting {

**static** ExtentHtmlReporter *htmlReporter*;

**static** ExtentReports *extent*;

**static** ExtentTest *logger*;

@Test

**public** **void** startReport(){

*htmlReporter* = **new** ExtentHtmlReporter(System.*getProperty*("user.dir") +"//STMExtentReport.html");

*extent* = **new** ExtentReports ();

*extent*.attachReporter(*htmlReporter*);

*extent*.setSystemInfo("Host Name", "SoftwareTestingMaterial");

*extent*.setSystemInfo("Environment", "QA");

*htmlReporter*.config().setDocumentTitle("Automation Report");

*htmlReporter*.config().setReportName("placeorder");

*htmlReporter*.config().setTestViewChartLocation(ChartLocation.***TOP***);

*htmlReporter*.config().setTheme(Theme.***STANDARD***);

}

@Test

**public** **void** passTest(){

*logger* = *extent*.createTest("Login \_TC\_001");

*logger*.log(Status.***PASS***, "Entered Username");

//logger.log(Status.PASS, MarkupHelper.createLabel("Entered Username", ExtentColor.GREEN));

*logger*.log(Status.***PASS***, "Entered Passed");

}

@After

**public** **void** endReport(){

*extent*.flush();

}

}

**package** test;

**import** java.util.Set;

**import** org.apache.log4j.Logger;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**import** org.testng.IClass;

**import** org.testng.IReporter;

**import** org.testng.ITestContext;

**import** org.testng.ITestListener;

**import** org.testng.ITestNGMethod;

**import** org.testng.ITestResult;

**import** org.testng.Reporter;

**import** org.testng.annotations.Listeners;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

@Listeners(test.BigBasket.**class**)//package.classname.name

**public** **class** BigBasket **implements** ITestListener{

**final** **static** Logger ***log*** =Logger.*getLogger*(BigBasket.**class**);

**public** **void** onTestStart(ITestResult result) {

System.***out***.println("New Test Started" );

}

**public** **void** onTestSuccess(ITestResult result) {

System.***out***.println("Test Successfully Finished" +result.getName());

}

**public** **void** onTestFailure(ITestResult result) {

System.***out***.println("Test Failed" +result.getName());

}

@Test

@Parameters({"a"})

**public** **void** display(String c) **throws** Exception

{

System.***out***.print(c+"Parameter value");

System.*setProperty*("webdriver.chrome.driver",System.*getProperty*("user.dir")+"\\src\\main\\resources\\drivers\\chromedriver.exe");

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

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

Reporter.*log*("flipkart launched");

}

}