**Selenium Syllabus:**

1. Selenium architecture and theoretical concepts
2. Simple script to automate a scenario.
3. Locators - Deep dive. - SIbling functions.
4. Webdriver and Webelement methods.
5. Maven,TestNG
6. Automating different web elements - Actions ,drowdowns, Switch window, Frames.etc
7. JavaScriptExecutor - scroll,click
8. Waits in Selenium.
9. Page Object model framework
10. Hybrid Framework.

**How to install Selenium -**

1. Downloaded Selenium Webdriver from Official Selenium Website.
2. Extracted Selenium files
3. In Eclipse, Properties of Project → Java Build path → Libraries —> Add External JARS→ Selected and opened all the Selenium files that we extracted.--> Apply and close.
4. Download Chromedriver (Make sure to download the latest version of chromedriver by checking against your chrome version / the chrome version used in your project)
5. We include the chromedriver file path in your script.

Java Concepts to Understand Selenium -

1. Class and Interfaces.
2. OOPS - Inheritance, Abstraction, Overloading, Overriding.
3. Polymorphism, Encapsulation.
4. Upcasting, Downcasting

**First Selenium Script :**

public class FirstScript {

public static void main(String[] args) {

//System.setProperty("webdriver.chrome.driver", "G:\\chromedriver\_14Version\\chromedriver.exe");

System.setProperty("webdriver.chrome.driver", "G:\\chromedriver\_14Version\\chromedriver.exe");

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

driver.get("https://www.guru99.com/");

}

}

**System.setproperty** - It is used to enter the details of the driver(chromedriver,firefoxdriver,edgedriver) in a key value pair of Setproperty(key,value) key is the driver detail, value is the location of

the driver

**Webdriver** is an **Interface**, driver is the reference variable, new is the keyword, ChromeDriver is the Constructor, together new **ChromeDriver**() object is created.

and stored in reference variable **driver**.

Here,All the driver methods are just declared in **Webdriver** Interface, and all the driver methods(get,close,quit,findelement) are implemented in

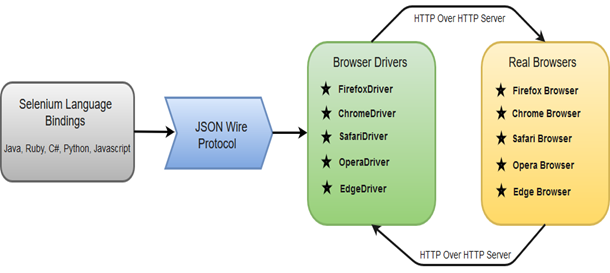
Implementation class called **ChromeDriver**();

Here **Upcasting** is used to call the Chrome Browser, when we upcast the main methods opens the chrome browser

**Selenium Architecture**

Selenium WebDriver API provides communication facility between languages and browsers.

The following image shows the architectural representation of Selenium WebDriver.

****

### Selenium Language Bindings / Selenium Client Libraries

Selenium developers have built language bindings/Selenium Client Libraries in order to support multiple languages. For instance, if you want to use the browser driver in java, use the java bindings.

### JSON Wire Protocol

JSON (JavaScript Object Notation) is an open standard for exchanging data on the web. It supports data structures like objects and arrays.

### 

### Task after today :

1. Update to Java 17.
2. Selenium version - 4

### Browser Drivers

Selenium uses drivers, specific to each browser in order to establish a secure connection with the browser without revealing the internal logic of browser's functionality. The browser driver is also specific to the language used for automation such as Java, C#, etc.

**We downloaded Chromedriver for Java.**

There will be different chromedriver files for different Language bindings.

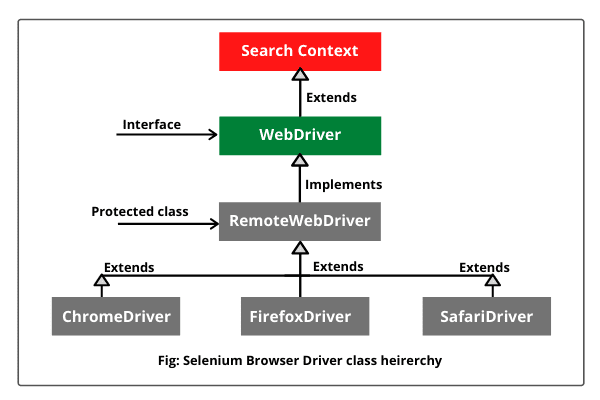
The communication between driver and Browser will happen in the form of HTTP Request and Response

To specify the path of drivers we use System.setProperty().

Exceptions :

1. **NoSuchDriverException** - This exception we get when we don't provide the correct address of the chromedriver.
2. **IllegalStateException** - We get this when we provide incorrect key value pair of System.Set.Property.
3. **IllegalArguement Exception** - when we provide empty key

**Hierarchy of Selenium Webdriver.**

****

**SearchContext -** is an Interface with 2 Abstract methods - FindElement() and FindElements().

**WebDriver -** is an interface which has 11+2 abstract methods.

**RemoteWebDriver -** is a protected class which is an implementation class for webdriver. It has 13 concrete methods.

Remote webdriver implements Webdriver.

**ChromeDriver - A** [**WebDriver**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebDriver.html) **implementation that controls a Chrome browser running on the local machine. It requires a chromedriver executable to be available in PATH.**

**ChromeDriver extends RemoteWebDriver**

We can also write **:**

**RemoteWebDriver driver = new ChromeDriver**

**instead of**

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

**Other statement to invoke browser -**

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

**Homework -**

1. Write explanation as much as possible for those 2 lines .

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

WebDriver driver = new ChromeDriver();

2) Inspect Webelements in Youtube,Amazon, <https://demoqa.com/>,

WebElement firstname =driver.findElement(By.xpath("//input[@id='userName']"));

**WAP to open a firefox browser and close the browser.**

**Locators**

**I**n Automation , before performing any action such as click, clear, pass the data (send keys) we need to find the address of the element.

We do that by inspecting the element,

To find the element, we use Locators

Locators are classified into 8 types.

1. ClassName.
2. CSS selector.
3. ID.
4. LinkText.
5. Name
6. PartialLink text.
7. TagName
8. X path

All of the Locators are static methods of By class.

By class is an abstract class.

Official info about Locators in Selenium website :

https://www.selenium.dev/documentation/webdriver/elements/locators/

**How to use By.id in selenium script?**

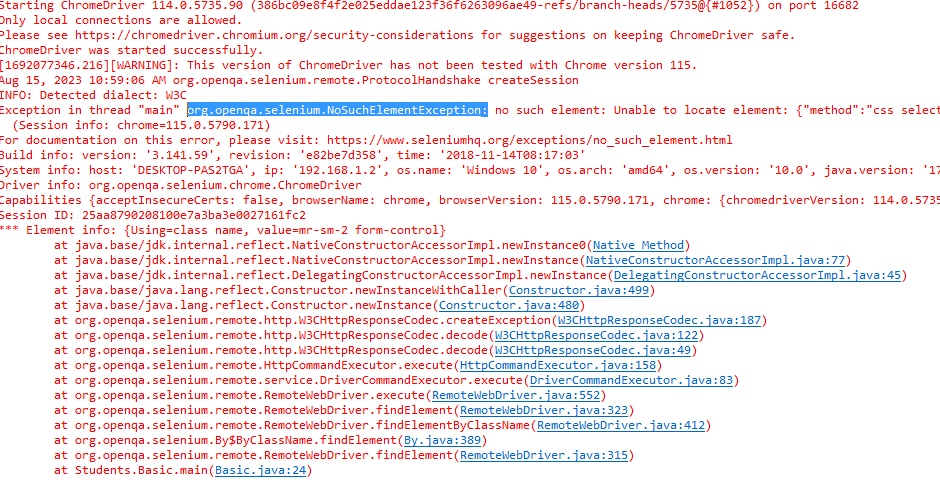
WebElement username = driver.findElement(By.id("userName"));

**How to use By.name in selenium script?**

driver.findElement(By.name("checkBoxOption1"));

**Note :**

Whenever we give a incorrect Locator, we get **NoSuchElement Exception.**

****

**HTML**

Hyper text markup Language.

HTML is used to design a webpage.

In HTML, Developers use predefined keywords within Angular braces<> to develop a webpage.

By using different tags Dev designs the Web Elements inside a Web Page.

<html>

<head>

<title> AUtomation website </title>

</head>

<body>

Username :<input id= "username"> <br><br>

Password : <input type ="password" <br><br>

Submit : <input type ="password" <br><br>

Forgot Pass: <input type ="password" <br><br>

</body>

</html>

**13 Methods used from Webdriver Instance :**

1. close()
2. findElement()
3. findElements()
4. get()
5. getCurrentUrl()
6. getPageSource()
7. getTitle()
8. getWindowHandle()
9. getWindowHandles()
10. manage()
11. navigate()
12. quit()
13. switchTo()

Relative Locators in Selenium Website :

<https://www.selenium.dev/documentation/webdriver/elements/locators/#above>

Tasks :

1. Explore 13 methods of webdriver -
2. Design a HTML webpage

**Xpath**

It is a path of the element in the HTML tree structure.

Xpath Syntax - //Tagname[@key = ‘Value’];

**Xpath example** - **//input[@id='userName']**

**Absolute Xpath - only Forward slash** - Used to navigate from parent to immediate child tag.

**Relative Xpath -**

**Double forward slash** - Used to travel directly to the specified tag.

We can also use combination of **Relative** and **Absolute** Xpaths.

**How to write Xpath with Multiple attributes.**

USing ‘**and’** operator

**Example** - //input[@class='nav-input nav-progressive-attribute' and @type="text"]

**How to use Text Function in Xpath -**

**Syntax :**

//tagname[text()=’Text Value’]

Example in Gmail SIgninPage :

//span[text()='Create account']

**How to write Xpath using Indexing -**

Syntax - **(//tag[@attribute=’Value’])[index]**

**Contains Function**

**Syntax -**

//Tag[contains(text(),’textvalue’)]

Example -

(//div[@class="\_fluid-quad-image-label-v2\_style\_fluidQuadImageLabelBody\_\_3tld0"]//img)[3]

//a[contains(text(),'Grocery ')]

//\*[contains(text(),'Forgot')]

**Combination of Contains as well as Indexing in Xpath** (//\*[contains(text(),'akshay')])[2]

Assignment -

<div

<a1 id=”A1” name = “N1” class = “C1” href= “[www.g](http://www.google.com/)mail.com” > Gmail</a1>

<a id=”a1” name = “n1” class = c1 href= “[www.google.com/](http://www.google.com/)” > Google </a>

</div>

Write As much as possible Xpath both for Gmail and Google link

**Note :**

**ElementClickInterceptedException** - WE get this Exception when the Selenium click action is Intercepted by other Elements in the Webpage like Live ads or other elements in the webpage.

**Creating a maven project.**

Instead of every time providing the jar files in the build path of the project, we can use a maven project and provide dependencies in pom.xml from maven repositories

Example of Dependency tag in pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>orange</groupId>

<artifactId>orange</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.3.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.8.1</version>

</dependency>

</dependencies>

</project>

**Features of Maven :**

1. **Pom.xms – >** Directly downloads the dependencies mentioned in Pom.xml in dependencies tag.
2. It provides folder structures to your project. src/main/java and src/test/java. To write your Automation framework -

**src/main/java** - We write our POM(Page Object Model) classes and Utility classes.

**src/test/java** - We write our Test scripts.

1. We can also integrate Maven with the Jenkins pipeline.
2. We can also Run Test scripts with Maven.

**TestNG**

Installation of TestNG :

**1)Download and install testNg from Eclipse marketplace.**

**2) Also add TestNG dependency from MVN repo.**

**Demonstrate how to write tests using TestNG annotations.**

public class TestNgDemoTest {

WebDriver driver;

@BeforeTest

public void setup() throws InterruptedException {

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

driver = new ChromeDriver();

Thread.sleep(2000);

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

}

@Test

public void teststep() {

WebElement username = driver.findElement(By.name("username"));

WebElement password = driver.findElement(By.name("password"));

WebElement Loginbtn = driver.findElement(By.xpath("//button"));

username.sendKeys("Admin");

password.sendKeys("admin123");

Loginbtn.click();

}

@AfterTest

public void laststep() {

driver.quit();

}

}

**­­­­­­TestNG - Unit Testing Framework Tool**

TestNG is an **open source automated testing framework**, where NG stands for Next Generation.

Unit testing framework tools available are:

TestNG ------------> Java, .Net

Junit -------> Java

Nunit --------->.Net

Py dev -------->Python

All unit testing framework tool is implemented as plugin for eclipse IDE, but Junit is a default plugin for eclipse IDE.

**Installation steps of TestNG:**

Goto Eclipse window

Click on help option-> Eclipse Marketplace

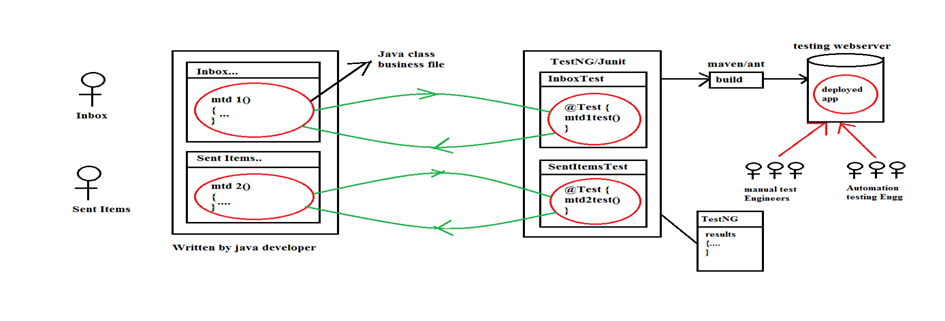
1. Write TestNG in find edit box and click on go button.
2. Find TestNG for eclipse division and click on install button
3. Click on confirm button and I accept the terms and conditions and click on finish

In order to verify the TestNG installation->Go to windows

-> Show view-> others -> expand java folder, TestNG symbol will be present

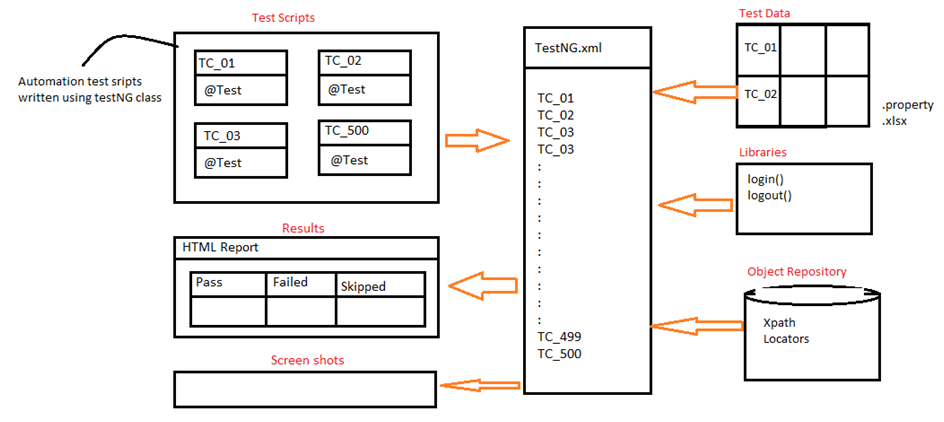
**TestNG for development:**

TestNG is used in development to write white box testing test cases/ unit test case and each unit test case will be used to test the source code of the application

****

In case of development, testNG will used to develop; unit test cases and each unit test case build the business of the source code.

**TestNGfor selenium automation**

****

In case of automation, testNG will be used to develop all the scripts using testNG annotations and achieve batch execution without any manual interaction.

TestNG will be used to handle framework components. TestNG is inspired from Junit and Nunit, but introducing few new functionality makes testNG become more powerful.

**New functionalities are:**

->Html report

->Parallel execution - 1) Executing the separate scripts at the same time.

2) Executing the same scripts with different browsers.- Compatibility

->Grouping execution - We can group certain scripts like smoke, regression.

->Additional annotations

->batch execution is easier

-> iTest Listeners - Interfaces which are defined, to implement Screenshot and Extent reports.

->Retry Analyser - is also a Interface to make our scripts run again. - Retrying.

**TestNG Annotations:**

Annotations - Its Java block , which is used to provide metadata(information/instruction ) to the JVM , at the time of execution in RUN-Time.

**Why can’t we use Main methods for execution? Why do we need TestNG annotations?**

Execution - we cant execute multiple main methods at once. We can only execute one main method.

And we cant write all out test scripts in only main method.

@Test

@BeforeMethod

@AfterMethod

@BeforeClass

@AfterClass

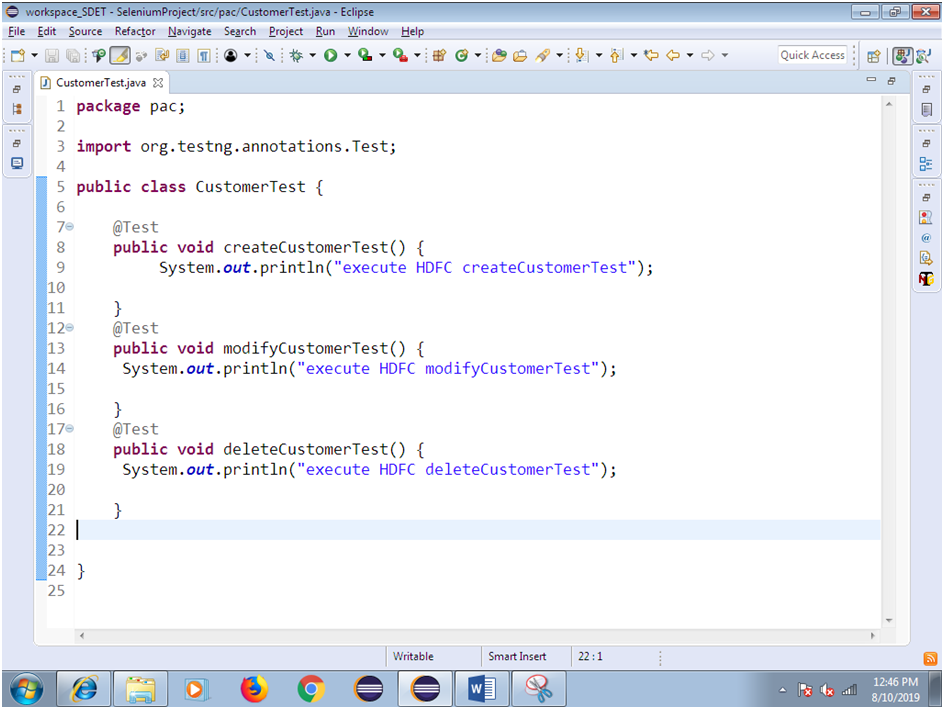
@BeforeTest

@AfterTest

@paramaters

@dataProvider

@Listner



**TestNG important pointers**

1. Whenever we execute a testng class, compiler always looks for @test Annotation method to start the execution.

**2.** **Without @Test , testNG class will not be executed, @test annotation method act like main method in testing.**

**Rules for writing TestNG scripts in fraemework:**

1. In one testng class we can have multiple test methods, but each test method should have @Test annotation before method signature.

2. Annotation method return type should be “void” and access specifier should be public., but method name can be anything.

3. As per the Rule of the Automation , TESTNG class Name should be ModuleNAme , @test method name should be manual testCase Name

4. As per the Rule TestNG class Name & testNG method Name should end With “Test”

TestNg will always look for this string whenever it is executing.

7. Verify html report

**How to access html report file in TestNG?**

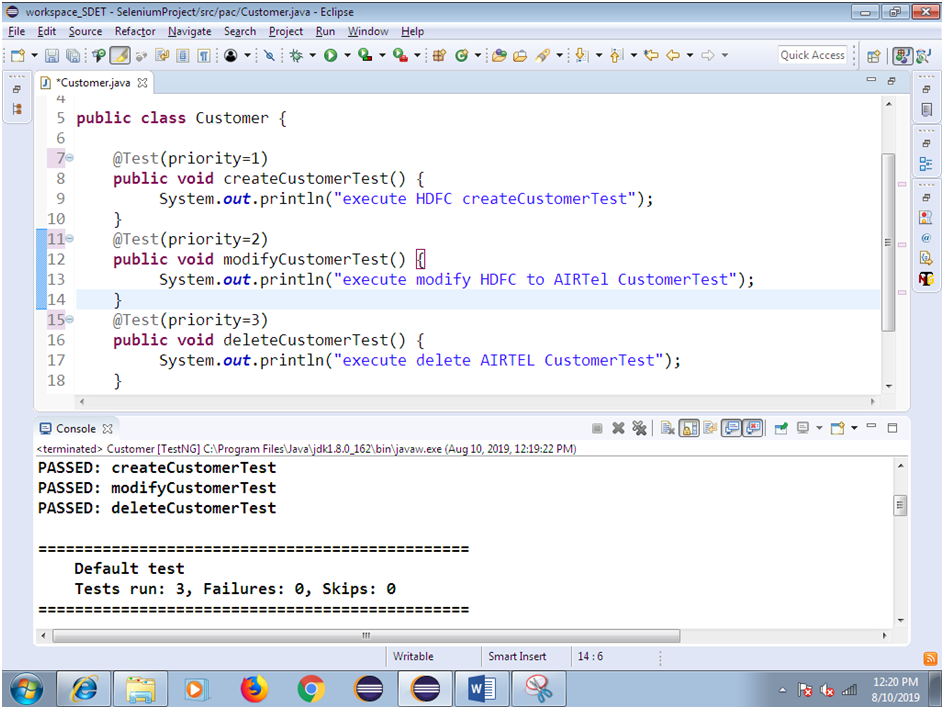
Refresh the project after execution—select project right click and click on refresh

Automatically we get **test-Output** folder within the same project.

Expand test-output folder à select emailable.html and right click à open withà Open with browser

8. Priority

Whenever we execute testNG class , by default all the test method will be executed based on Alphabetical Order , in order the change the Order of Execution , we go for priority

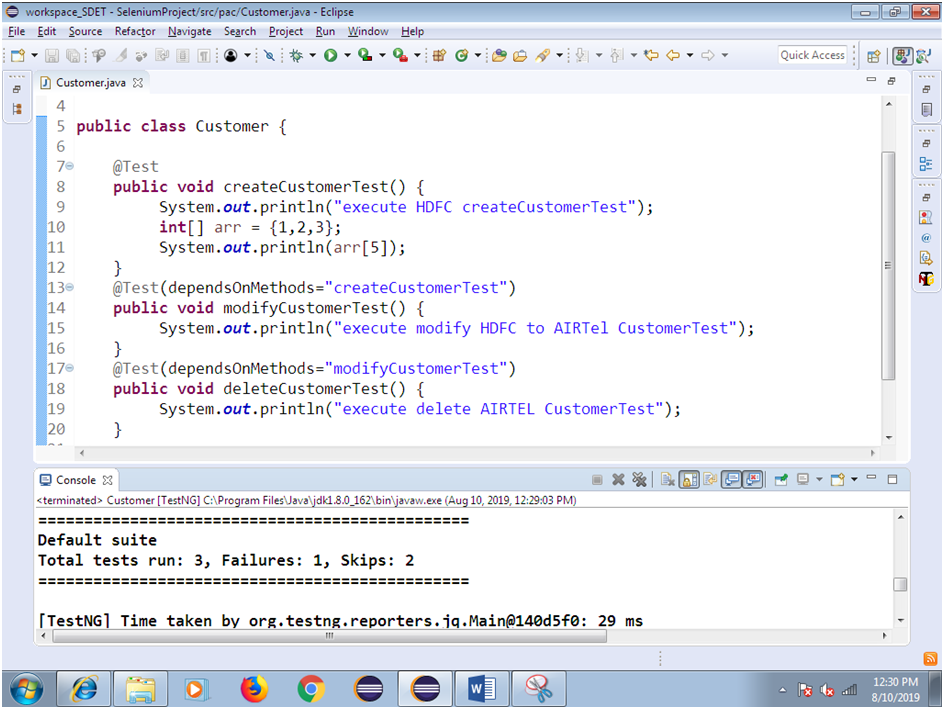


**9.** **DependsOnMethod :**

Its help us to check the dependent test case is pass or fail,

If dependent test-script get pass, execution will continue

If dependent test-script get fail, skip the test execution



**Robot Class implementation for Keyboard actions.**

@Test **public** **void** test2() {

Robot rt= **new** Robot();

Actions act= **new** Actions(wd); WebElement we1=wd.findElement(By.*xpath*("(//\*[@class='gb\_y'])[1]")); act.contextClick(we1).build().perform();

rt.keyPress(MenuKeyEvent.***VK\_DOWN***);

rt.keyRelease(MenuKeyEvent.***VK\_DOWN***);

rt.keyPress(MenuKeyEvent.***VK\_DOWN***);

rt.keyRelease(MenuKeyEvent.***VK\_DOWN***);

rt.keyPress(KeyEvent.***VK\_ENTER***);

//rt.keyPress(MenuKeyEvent.VK\_ENTER);

//rt.keyRelease(MenuKeyEvent.VK\_ENTER); } }

**How to Handle Dropdown in Selenium For Select tag :**

**Step 1 :**

Whenever we encounter a Dropdown, inspect and check which HTML tag is the dropdown developed with.

**Step 2:**

Scripting -

Create an object for Select class and pass the Webelement in the constructor argument :

WebElement **dropdown** = driver.findElement(By.xpath("//select[@id='dropdown-class-example']"));

Select sel = new Select(dropdown);

Using following methods select the options from the dropdown :

// To select Option1 from dropdown

**sel**.**selectByValue**("option1");

// To select option 2

sel.selectByIndex(2);

// To select option 3

sel.selectByVisibleText("Option3");

**For Other tags :**

Drop downs in a website could be created in several different ways. Some dropdowns are created using **<select>** HTML tag and some others are created using <**ul> ,<li>, <button>** and **<div>** tags.

Selenium WebDriver provides Select class which can be used only for drop down created using **<select>** HTML tag. Select class has methods such as **selectByVisibleText()**, **selectByIndex()** and **selectByValue()** to select the desired option from dropdown.

However, for **NonSelect** dropdowns, **Select** class cannot be used. There should be a common way to handle different types of dropdown through Selenium Automation.

String option="Highest to lowest";

for(int i=0; i<allOptions.size(); i++) {

if(allOptions.get(i).getText().contains(option)) {

allOptions.get(i).click();

System.out.println("clicked");

break;

}

}

Using For Each Loop :

for (WebElement we : Options) {

//String opt= we.getText();

//System.out.println(opt);

if(we.getText().contains("APP")) {

we.click();

System.out.println("clicked");

break;

}

**Web tables**

1.What is Web Table.

2. Inspect webtable from money control website

3. Explain Hierarchy of head, tbody, tr, and td.

4. Demonstrate how to write xpath in nth row and nth column from the webtable using traversing from table to thead to tr to td.

How to scroll the webpage using JavaScriptExecutor.

**Switch to Windows and Tabs.**

### **Get window handle**

WebDriver does not make the distinction between windows and tabs. If your site opens a new tab or window, Selenium will let you work with it using a window handle.

Each window has a **unique identifier** which remains persistent in a single session. You can get the window handle of the current window by using:

**driver.getWindowHandle()**

We can also say that **getWindowHandle()** will return the **window Handle of Parent Window.**

**Return type of getWindowHandle() is String**

If we print this string then we get a unique alphanumeric string.

### **Switching windows or tabs**

* Clicking a link which opens in a new window will focus the new window or tab on screen, but WebDriver will not know which window the Operating System considers active.
* To work with the new window you will need to switch to it. If you have only two tabs or windows open, and you know which window you start with, by the process of elimination you can loop over both windows or tabs that WebDriver can see, and switch to the one which is not the original.

**getWindowHandles()** - It returns the window handles of both parent and child windows.

Approach to switch Window -

**Step 1** : First we should get the window handle of Parent Window.

**Step 2:** Then, after multiple window openings, we get the window handle of all the opened windows

Through **getWindowHandles().**

**Step 3** : Then, we Iterate over each individual window id’s returned from **getWindowHandles().**

**Step 4** : Using if condition we compare the individual window id to the parent id.

If the individual Id is not equal to the parent ID, it means that we have the child window.

**Step 5** :Then we switch to the specified window ID of the child using **driver.switchto().window(child\_id);**

**Java Code**

public void switchwindowTest() throws InterruptedException {

WebElement switchbutton = driver.findElement(By.xpath("//button[@id='openwindow']"));

switchbutton.click();

// now the window is switched

String parentwindow = driver.getWindowHandle();

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

for (String individual : windows) {

if(!individual.equalsIgnoreCase(parentwindow)) {

driver.switchTo().window(individual);

Thread.sleep(3000);

break;

}

}

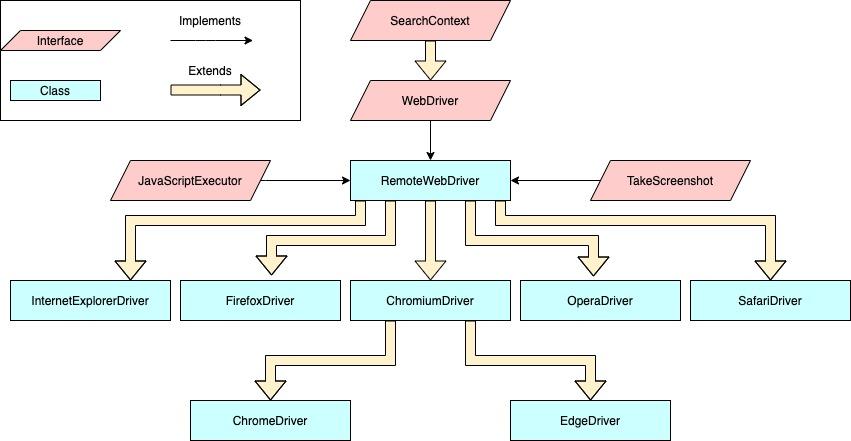
// To close the child window

driver.close()

}

}

**JavaScriptExecutor in Selenium**

****

As we can see the hierarchy of webdriver wrt JavaScriptExecutor Interface.

Here we type cast the driver instance of webdriver to javascript executor.(Explicit typecasting)

JavaScriptExecutor js = (JavaScriptExecutor) driver;

With the help of Java script executor :

We can automate scroll action.

**To scroll down or up we use the following method :**

js.executeScript("window.scrollBy(0,250)", "");

* **JavascriptExecutor in Selenium to click a button**

js.executeScript(“document.getElementByID(‘element id ’).click();”);

* **JavascriptExecutor in Selenium to send text**

js.executeScript(“document.getElementByID(‘element id ’).value = ‘xyz’;”);

* **JavascriptExecutor in Selenium to interact with checkbox**

js.executeScript(“document.getElementByID(‘element id ’).checked=false;”);

* **JavascriptExecutor in Selenium to refresh the browser window**

js.executeScript(“location.reload()”);

**Waits**

**Implicit Wait** : Implicit Wait directs the Selenium WebDriver to wait for a certain measure of time before throwing an exception.

Once this time is set, WebDriver will wait for the element before the exception occurs.

Once the command is run, Implicit Wait remains for the entire duration for which the browser is open. It’s default setting is 0, and the specific wait time needs to be set by the following protocol.

Syntax :

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

### **Explicit Wait in Selenium**

By using the Explicit Wait command, the WebDriver is directed to wait until a certain condition occurs before proceeding with executing the code.

Setting Explicit Wait is important in cases where there are certain elements that naturally take more time to load. If one sets an implicit wait command, then the browser will wait for the same time frame before loading every web element. This causes an unnecessary delay in executing the test script.

Explicit wait is more intelligent, but can only be applied for specified elements. However, it is an improvement on implicit wait since it allows the program to pause for dynamically loaded Ajax elements.

In order to declare explicit wait, one has to use ExpectedConditions. The following Expected Conditions can be used in Explicit Wait.

* alertIsPresent()
* elementSelectionStateToBe()
* elementToBeClickable()
* elementToBeSelected()
* frameToBeAvaliableAndSwitchToIt()
* invisibilityOfTheElementLocated()
* invisibilityOfElementWithText()
* presenceOfAllElementsLocatedBy()
* presenceOfElementLocated()
* textToBePresentInElement()
* textToBePresentInElementLocated()
* textToBePresentInElementValue()
* titleIs()
* titleContains()
* visibilityOf()
* visibilityOfAllElements()
* visibilityOfAllElementsLocatedBy()
* visibilityOfElementLocated()

Difference between

**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 helps reduce code duplication and improves test case maintenance.

In POM design pattern, we store the objects of a Webpage in one particular class.

**Design Pattern :**

1. **For each Webpage we have a unique class design and store the webelements in that webpage into that class only.**
2. **We use PageFactory class for initialisation in constructor.**
3. **We use @FindBy annotation to store the webelements.**
4. **We can also make the webelements encapsulated by declaring them private and providing access through getters method.**
5. **Business methods are included in the class to reduce lines of code**

**In the Test script method.**

For Example - Facebook

In Login Page, we store username, password, submit, forgot password.

In Home Page - We create a class separately for Homepage and we store only the Web Elements present in Homepage like - Notifications, Messages, Profile, Settings button.

In POM ,we make use of constructor, to Initialise the driver instance.

Assignment : https://demo.guru99.com/test/newtours/index.php

1. Write a script to login, print title before login and after login and check if they are equal or not and print the result.
2. Book a flight, by filing all the details.
3. Register as a user , by filing all the details.
4. Print all the links present in the homepage.
5. Include Extent report and screenshot in all the scripts.

**Actions class :**

**Actions class is**

**Data providers**

public class DataProviderExample {

WebDriver driver;

@BeforeTest

public void setup() throws InterruptedException {

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

driver = new ChromeDriver();

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

Thread.sleep(2000);

}

@DataProvider

public Object[][] provider(){

return new Object[][]{

new Object[]{"Admin","admin123"},

new Object[]{"Admin","admin123"},

new Object[]{"Admin","Admin123"},

};

}

@Test(dataProvider="provider")

public void Logintest(String username,String password) throws InterruptedException {

driver.findElement(By.xpath("//\*[@name='username']")).sendKeys(username);

driver.findElement(By.xpath("//\*[@type='password']")).sendKeys(password);

driver.findElement(By.xpath("//\*[@type='submit']")).click();

Thread.sleep(2000);

driver.findElement(By.xpath("//\*[@class='oxd-userdropdown-img']")).click();

driver.findElement(By.xpath("(//\*[@class='oxd-userdropdown-link'])[4]")).click();

Thread.sleep(2000);

}

}

**Data Driven Testing**

**Maven Dependencies for Data driven Testing for Excel files -**

<!-- 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/net.sourceforge.jexcelapi/jxl -->

<dependency>

<groupId>net.sourceforge.jexcelapi</groupId>

<artifactId>jxl</artifactId>

<version>2.6.12</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>

Step 1 : Create data in an Excel file in rows and columns(.xlsx file)

**Cucumber**

1. **Install Cucumber in Eclipse Marketplace.**