1) Selenium First Lecture

Automation testing was started from 2000

Selenium is open source automated testing framework or tool used to validate web application across different platform & Browsers. Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite. It was originally developed by Jason Huggins in 2004

What is Selenium?

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization. Here is the list of tools.

Introduction of Selenium

1) Architecture of selenium

2) Structures of webdriver interface & implemented classes.

3) configration of program for selenium

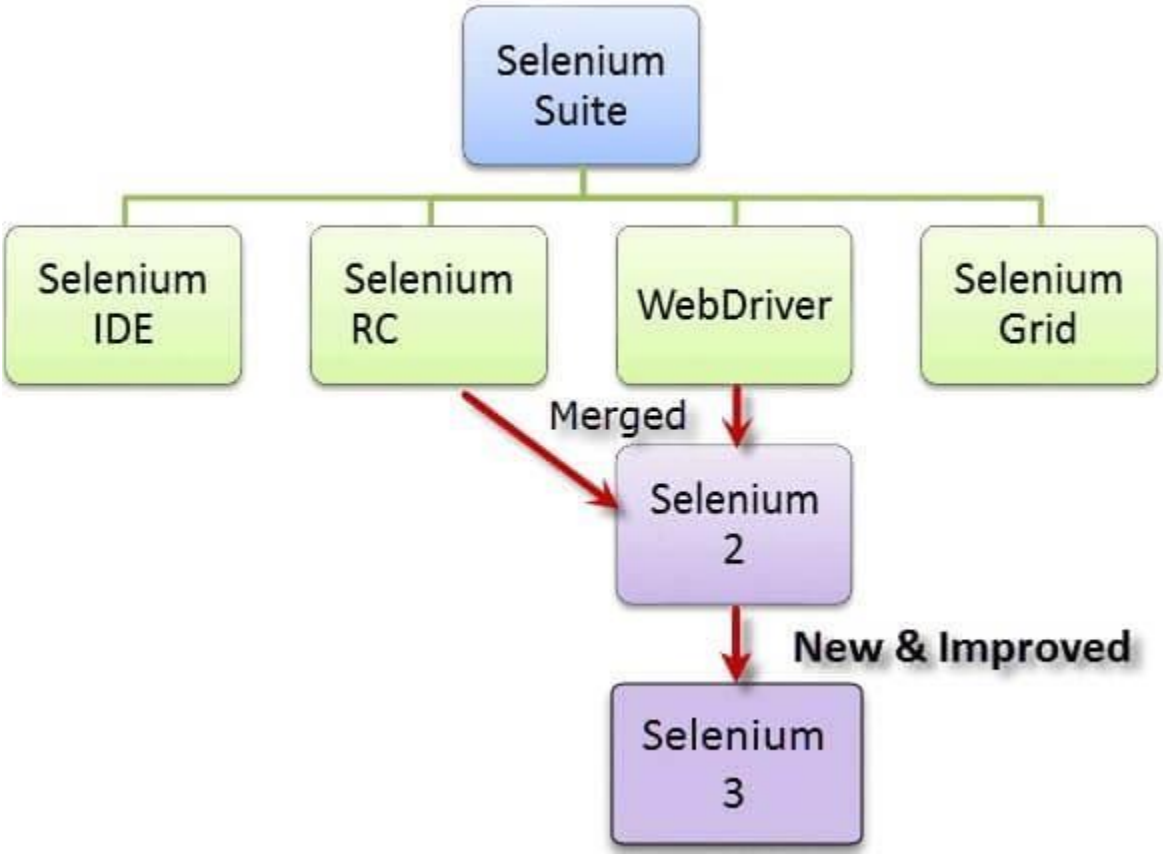
selenium webdriver download for OS 3.14.selenium webdriver download for OS 3.14.

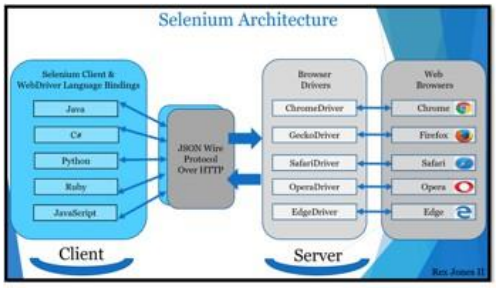
4) Fisrt script for browser.

Selenium Tool Suite

Selenium is not just a single tool but a suite of software, each with a different approach to support automation testing. It comprises of four major components which include:

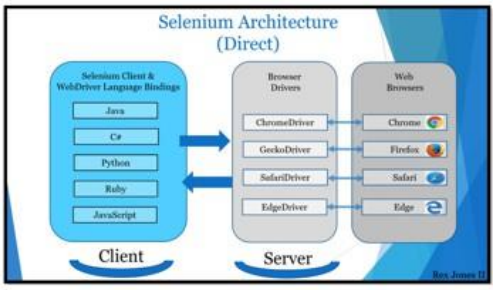
1. Selenium Integrated Development Environment (IDE)
2. Selenium Remote Control (Now Deprecated)
3. WebDriver
4. Selenium Grid





Selenium 3.0

Selenium Grid



Selenium 4.0

At the moment, Selenium RC and WebDriver are merged into a single framework to form **Selenium 2**. Selenium 1, by the way, refers to Selenium RC.

1. Selenium Integrated Development Environment (IDE)

Mozila firefox

Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG. You can use these exported script in Selenium RC or Webdriver.

>Selenium IDE has limited scope and the generated test scripts are not very robust and portable.

2. Selenium Remote Control

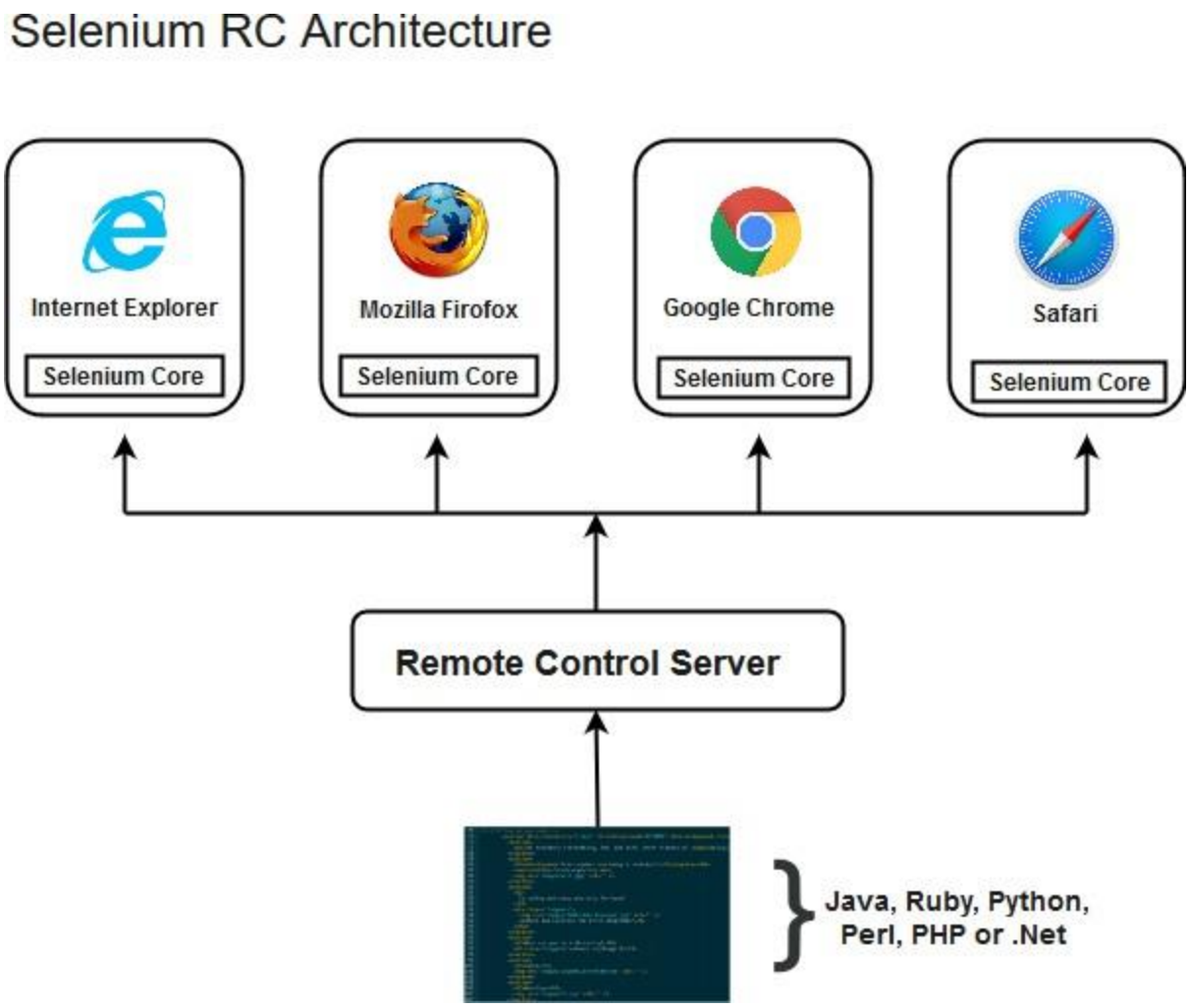
Selenium RC Need to download server locally.

Selenium RC (officially deprecated by selenium) allows testers to write automated web application UI test in any of the supported programming languages. It also involves an HTTP proxy server which enables the browser to believe that the web application being tested comes from the domain provided by proxy server.

Selenium RC comes with two components.

1. Selenium RC Server (acts as a HTTP proxy for web requests).
2. Selenium RC Client (library containing your programming language code).

The figure given below shows the architectural representation of Selenium RC.



3. Selenium WebDriver

Selenium WebDriver supports

i)cross-Platform-

ii) Cross browser-Chrome, edge, firefox, iE etc

Selenium WebDriver (Selenium 2) is the successor to Selenium RC and is by far the most important component of Selenium Suite. SeleniumWebDriver provides a programming interface to create and execute test cases. Test scripts are written in order to identify web elements on web pages and then desired actions are performed on those elements.

Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the web browser.

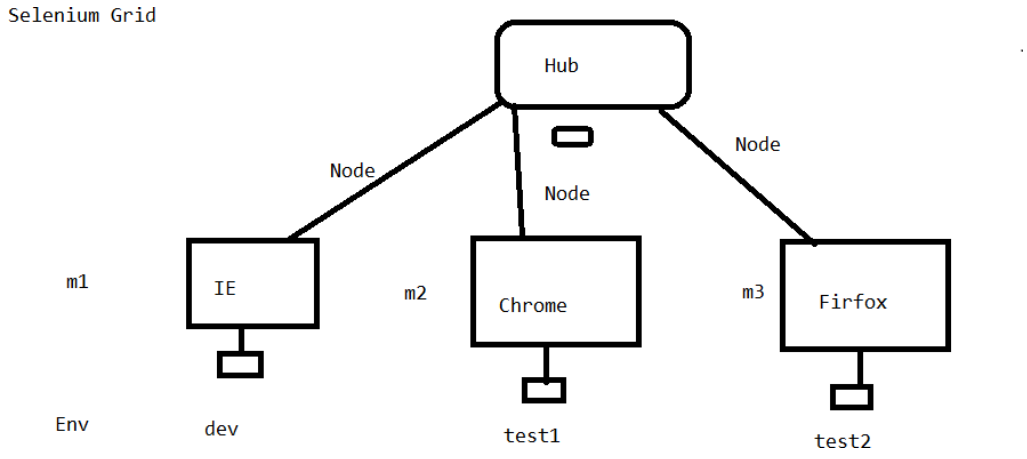
Since, WebDriver directly calls the methods of different browsers hence we have separate driver for each browser. Some of the most widely used web drivers include:

- Mozilla Firefox Driver (Gecko Driver)
- Google Chrome Driver
- Internet Explorer Driver
- Opera Driver
- Safari Driver
- HTML Unit Driver (a special headless driver)

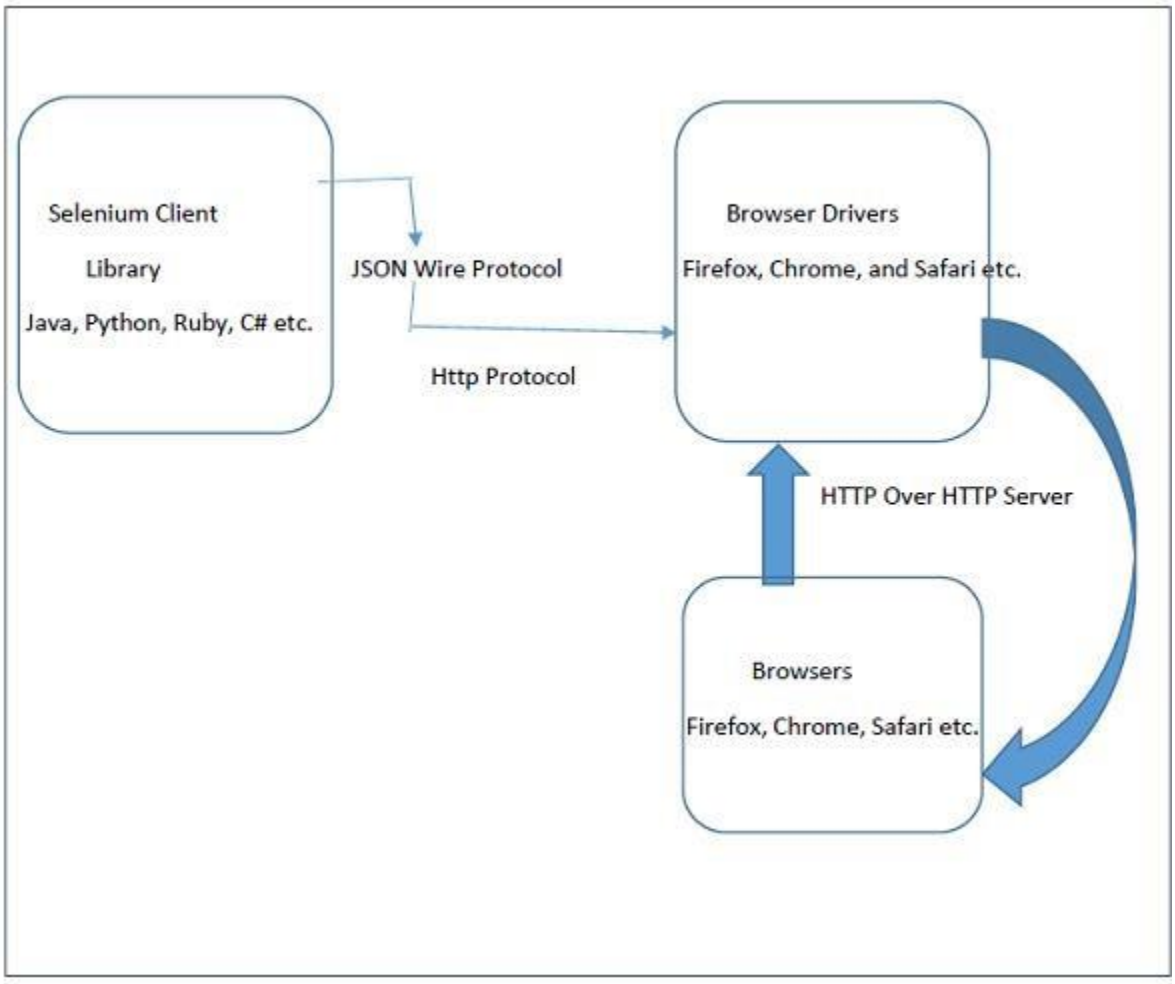
4. Selenium Grid

Selenium Grid is also an important component of Selenium Suite which allows us to run our tests on different machines against different browsers in parallel. In simple words, we can run our tests simultaneously on different machines running different browsers and operating systems.

Selenium Grid follows the **Hub-Node Architecture** to achieve parallel execution of test scripts. The Hub is considered as master of the network and the other will be the nodes. Hub controls the execution of test scripts on various nodes of the network.



Architecture of Selenium Web Driver



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.

Browser Driver: 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.

Let's consider the following block of code

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

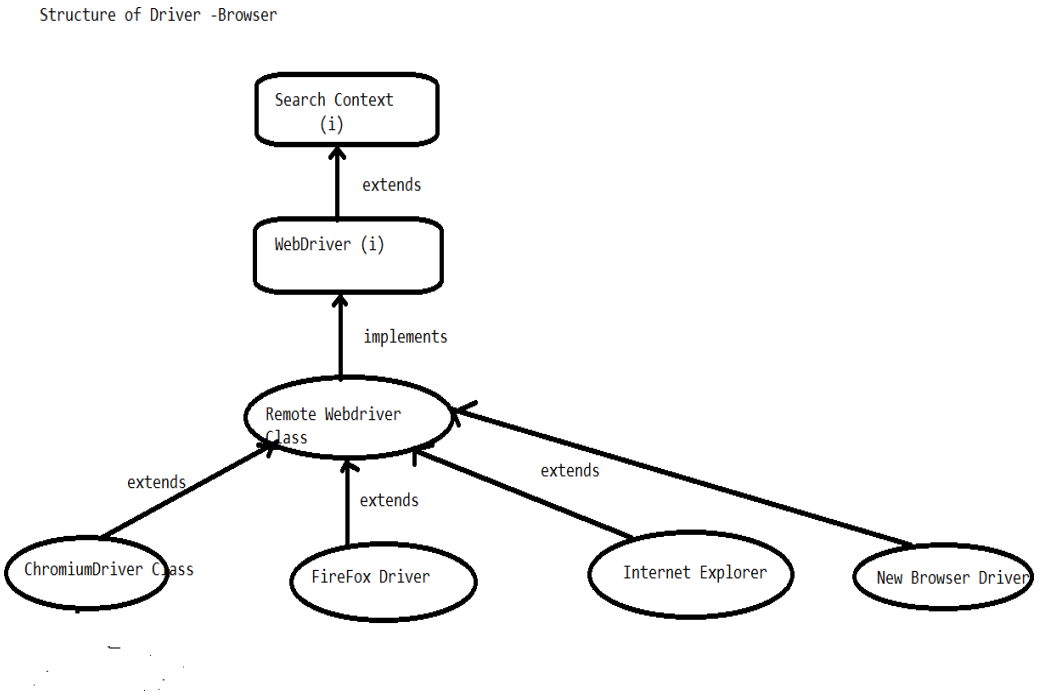
Once we run this block of code, the entire code will be converted with the help of JSON Wire Protocol over HTTP as a URL. The converted URL will be fed to the ChromeDriver.

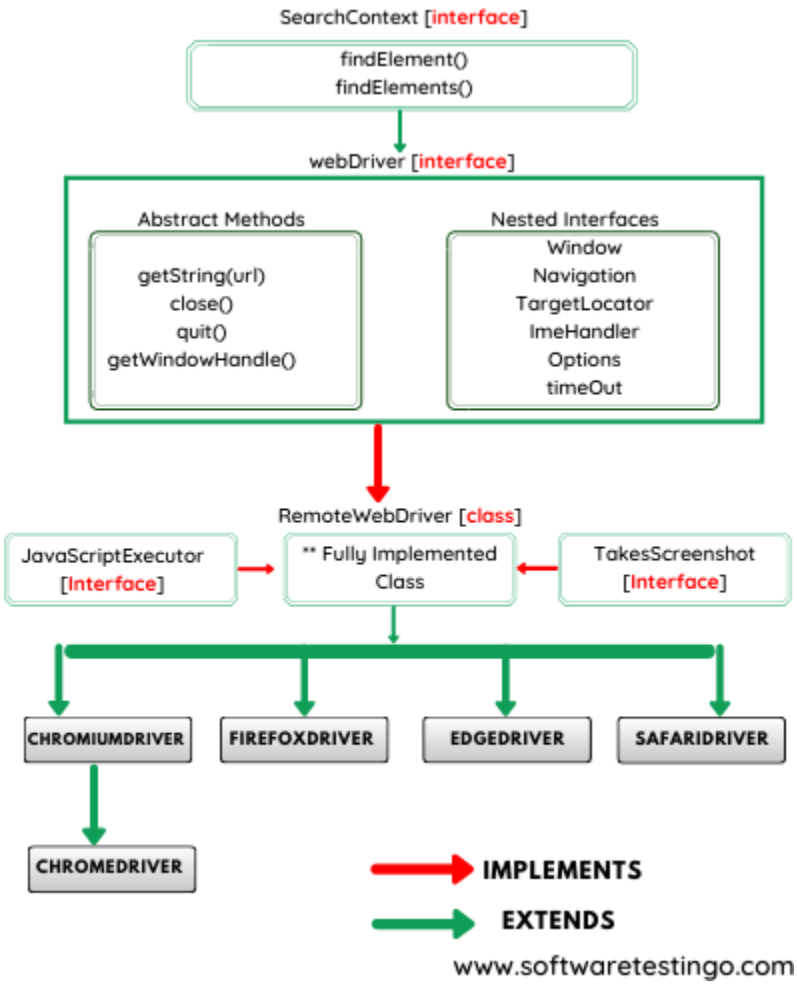
The browser driver utilizes HTTP server to get the request from HTTP. As the browser driver gets the URL, it passes the request to its browser via HTTP. It will trigger the event of executing the Selenium instructions on the browser.

Now if the request is that of POST, it will trigger an action on the browser. If it's a GET request, then the response will be produced at the browser end. Finally it will be passed over HTTP to the browser driver. The browser driver will in turn send it to the UI via JSON Wire Protocol.

2) Selenium 2 Lecture

Structure of Driver Browser / Selenium Hierarchy / Selenium Architecture

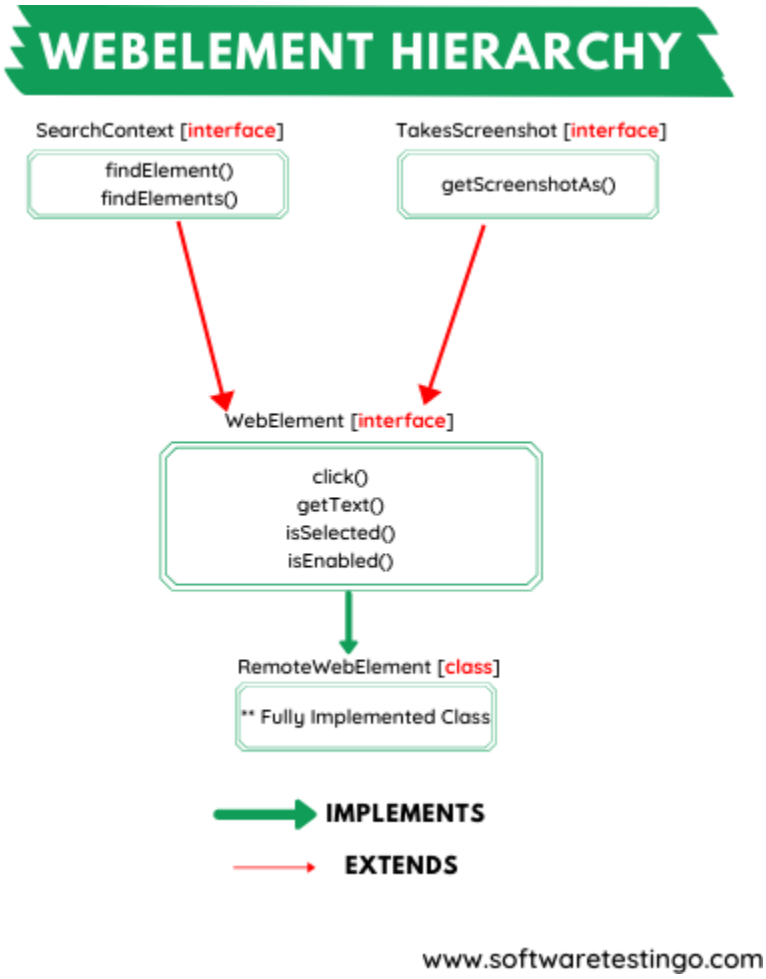




Let us explain the above hierarchy in details:

- **SearchContext** is the topmost interface of Webdriver which contains only two abstract method findElement() and findElements(). These two methods don't have a method body.
- **WebDriver** also is an interface which **extends SearchContext interface** which has also so many abstract methods like **close(), get(java.lang.String URL), quite(), navigate(), switchTo()** and other so many methods for more details you can visit this URL.
- The next one is **RemoteWebDriver**, which is a fully implemented class where all abstract methods of WebDriver and SearchContext interface implemented. Also, two other interfaces **JavascriptExecutor and TakesScreenshot** abstract methods are implemented in RemoteWebDriver class.
- And Finally, browser-specific driver classes available like FirefoxDriver, ChromeDriver, IEDriver, SafariDriver, etc.

WebElement Hierarchy:



- The **webElement interface extends** two other interfaces like **SearchContext** and **TakesScreenshot interfaces**. The webElement interface has so many useful methods that are frequently used during the automation. those methods like **clear(), click(), getText(), submit()** etc.
- RemoteWebELement is a class which implements all the abstract methods of the webElement interface
- When you try to search an element in DOM then you are going to use findElement() and findElements() then the remoteWebelement class is up-casting to webElement interface, and if you perform any operation like click() or submit(), then the overridden method of the remoteWebelement class is executed.

Selenium Webdriver Interface Interview Questions: Why we do Up-casting browser driver class object to Webdriver but why we are not up-casting into SearchContext or RemoteWebDriver?

Ans: When you are automating using selenium, then you may come across a frequently used statement where we are doing up-casting like:

- WebDriver driver= **new firefoxDriver()**;

Reasons:

1. If we write like above then by a single driver, we can able to launch any browser.
2. From the good programming, viewpoint its always a good practice to up-cast the object up to maximum level but keeping one thing in mind that we are not losing any important functionalities.
3. If we up-cast to SearchContext then as you saw in the above picture, it has only two methods that are findElement() and findElements(). So if we try to access other important methods, then we need to down-casting.
4. Because of that, we are Up-casting up to WebDriver only because if we up-cast to WebDriver then we are not facing any problems in access the important methods.

Selenium Features

- Selenium is an **open source and portable** Web testing Framework.

- Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
- It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a **reusable script with a simple-to-understand and easy-to-use interface.**
- Selenium supports various operating systems, browsers and programming languages. Following is the list:
    - Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
    - Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
    - Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It also supports **parallel test execution which reduces time and increases the efficiency of tests.**
- Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
- Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
- Selenium requires fewer resources as compared to other automation test tools.
- **Selenium web driver does not require server installation, test scripts interact directly with the browser.**

There are several features provided in the IDE under the toolbar, using which one can control the execution of test cases:

1. Speed Control – Helps control the speed of test cases
2. Run All – Allows execution of the entire Test Suite
3. Run – Runs the currently selected test
4. Pause/Resume – Allows a user to pause and resume a particular test case
5. Step – Helps step into each specific command in the test script

6. Rollup – Helps group all the Selenese Commands together and make them execute as a single operation.

Advantages and Disadvantages of Selenium-IDE

Advantages of Selenium IDE:

1. UI based record and playback tool.
2. Easy to understand & developed a test script.
3. Selenium-IDE test script export into web driver code & Selenium–RC
4. It is very easy to install and use.
5. No programming experience is required to use Selenium IDE.
6. Selenium-IDE has built-in help feature, and they show the documentation on the selected or entered command.
7. It helps us in debugging by displaying the information and error messages.
8. It provides excellent support for extensions.

Disadvantage of Selenium IDE:

1. It supports only Firefox browser because it implements as a plug-in for Firefox.
2. It cannot launch new browser and run the test script (it work with existing browser only)
3. It supports only the html language.
4. Parameterization is not possible.
5. Reading from the external file and run the test script is called parameterization.
6. Can be used only with a simple application, cannot be applied to develop Ajax application.
7. It is an asynchronous java framework which is used to create dynamic applications like-facebook.com, BBC.com
8. Looping and conditional statements are not allowed.
9. **Selenium IDE does not support error handling and database testing.**
10. Selenium IDE cannot be used in iPhone and Android applications testing.
11. Selenium IDE does not support iterations and conditional operations.

3) Selenium 3 Lecture

How to add Selenium jar file & run a sample program

1) Download Latest stable version 3.141.59 of Selenium from https://www.selenium.dev/downloads/

2) We have to Configure Selenium into Eclipse project

i) Open Eclipse

ii) Create one package name as Selenium in src

iii) Create a new class in Selenium package

iv) Now we have to add Selenium libraries(downloaded) into our project for that we have to

Then right click on Project then Select Build path then Select Configure path

v) Then click on Class path & Select Add External Jar option from right side & Add Stand alone Jar file & click on Apply & Close.

3) Then Check your Google chrome browser version & according to that download chrome driver.

4) Extract that zip file of chrome driver into Specific path

5) Then copy the path of chromedrive.exe file & paste it into System.setProperty(Program).

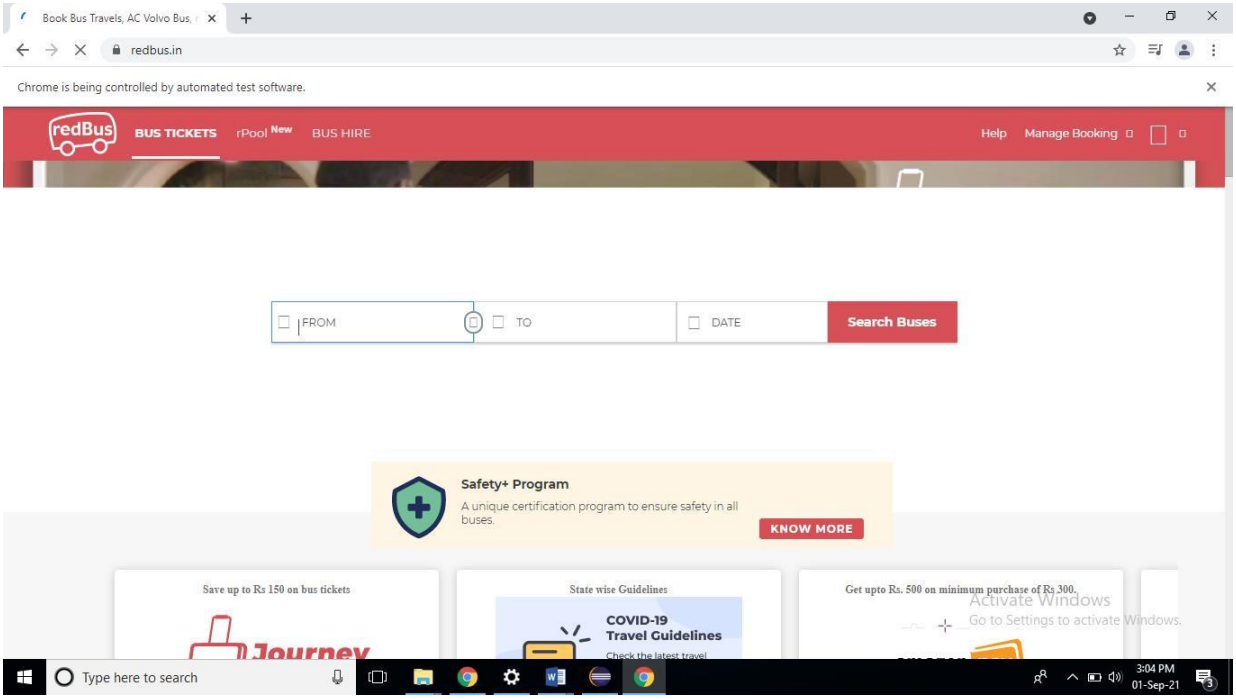
6) Run the program.**(If you get any error at import package statement then just comment all code in module-info.java file in Src of the package)**

package Selenium;

1) Program:

```
import org.openqa.selenium.WebDriver;

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

```
public class First {

	public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver= new ChromeDriver();

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

		driver.get("https://www.redbus.in");
		}
}
```



4) Selenium 4 Lecture

We have to perform some operation on Facebook web site

First we Navigate to given URL means we access that URL through Selenium script which will open in specific browser (Google Chrome)

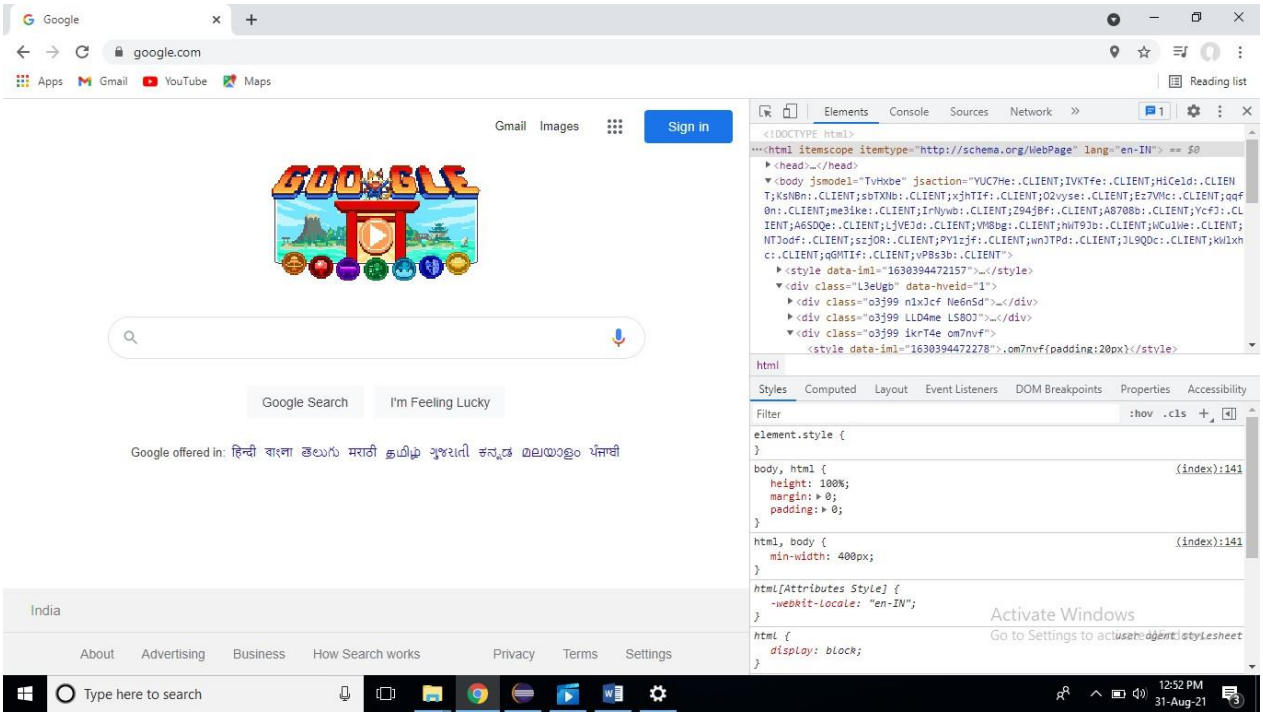
Then We will perform specific operation on login page

We access web page through the browser. There is interaction between browser & HTML pages.

HTML pages designed & located on that web address.

Whenever this type of design happens at that time some data is loaded in this browser & that data is presented in the form of document. That document is known **as DOM (Document object model)** structure.

If we right click on web page then we get Inspect Element option.



Then we are able to see HTML.

In HTML it is given that what has designed in this web page.

When developer develops UI then there are Elements in background like in google text box is one element

Element: Whenever webpage loads into browser then all the information is stored into object. That object is called as Element (Web Element).

DOM object is also called as Element. DOM follows HTML

In above diagram Gmail, Images, Google search, I'm feeling lucky all these are elements.

On the basis of HTML documents we locate Elements & we perform operations on Elements

By Clicking on any element we get some response. In this we do Element Identification for identifying what response we are getting by clicking on any element.

Locator: It is used to locate the web element on the webpage.

Locator is needed for Element Identification.

Example: For identifying Text box Element we use Locator.

Requirement:

Example: Facebook login web page

In above example Email id is a Text box element which takes input from user

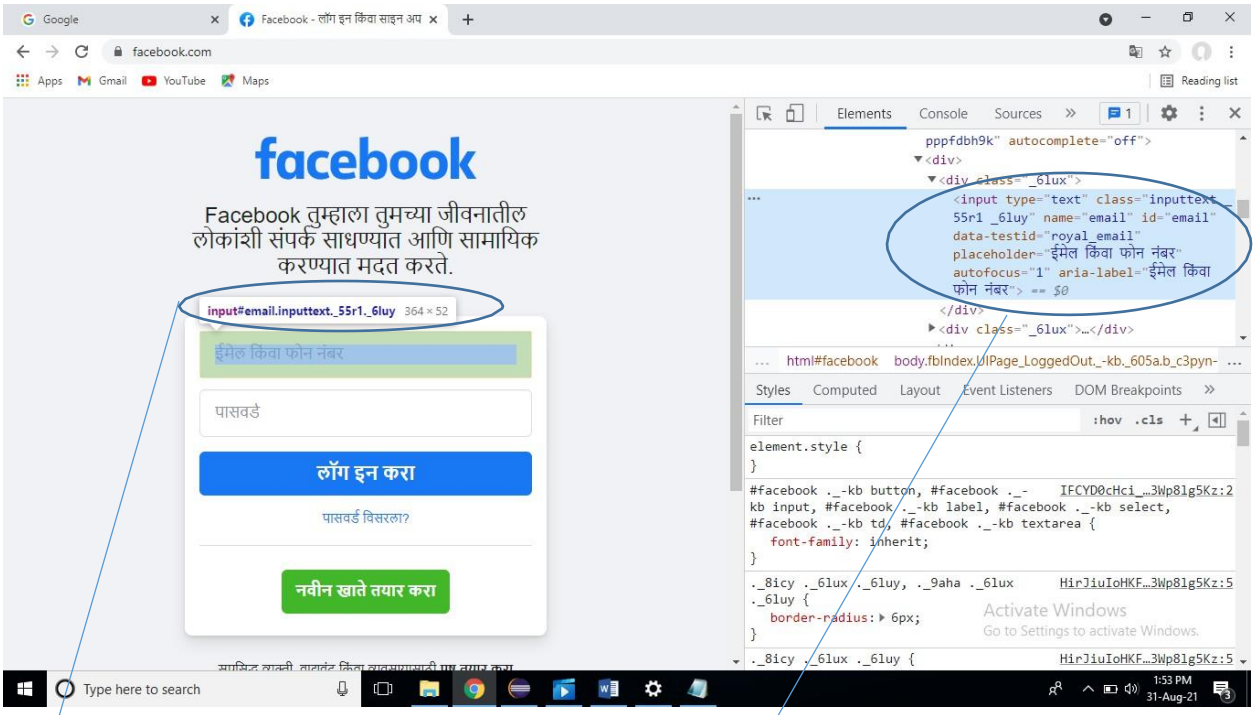
We have to give input to Email id text box element through Selenium script for that Locator tells the location of Email id text box element(exactly where it is located on facebook webpage)

i.e. **We use Locator to locate the Element on web page.**

After locating it will perform the operation i.e. Enter the value in Email id text box element.

When developer creates layout of web page at the same time developer enters the details about the identification of Element (developer enter details about which things are places at which place) in the HTML

We can **inspect specific element** also by right click on that element(Eg: Email id field on facebook)



It is known as Tag Name. **It is known as Attributes of Element**

In above example when we click on Element then we can see one identification bar on Email id field

There are two types of Locators

1)By Id 2)By Name

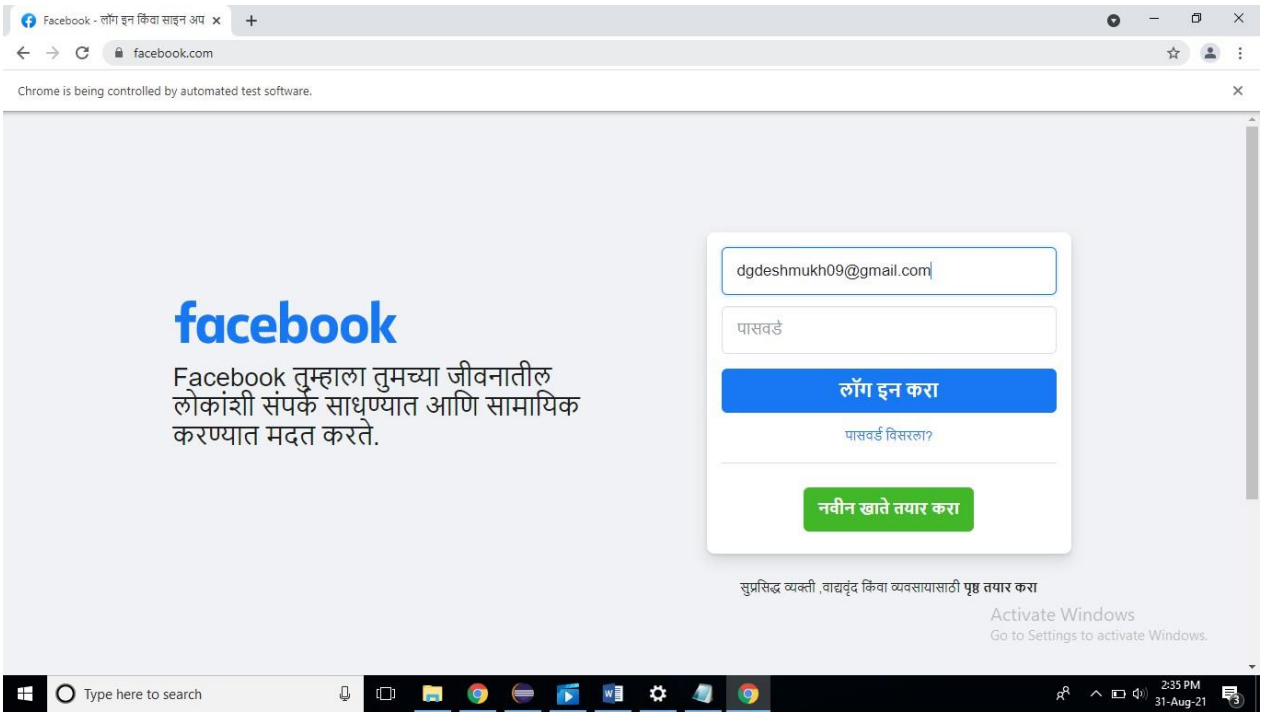
1) By Id Locator

Requirement: We have to locate Emaild Id in facebook webpageElement through Selenium script

Program:

```
package Selenium;
/*Date:31/08/2021
Program of Locator*/

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

public class ElementLocator {
```

```
	public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver= new ChromeDriver();

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

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

	driver.findElement(By.id("email")).sendKeys("dgdeshmukh09@gmail.c
om");
	}

}
```



5) Selenium 5 Lecture (Date: 1 Sep 2021)

Methods: Basic Methods in Web Driver

1) get ();

It use to open the web page by taking string argument

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

2) close ();

It will close the current window

Example: driver.close();

3) manage().windows();

It help to manage windows

Example: driver.manage.windows.maximize();

Why Maximize a Browser in Selenium Automation?

Elements on the web application may not be recognized by the selenium if the browser is not maximized and thereby making framework fail. Hence, Maximize the browser is very important part of selenium framework. It is good practice to maximize the browser while automating any web application. When the user executes the selenium framework or any script, browser may not be in the full screen state and you need to maximize the browser using window maximize in Selenium to view all the elements of the web application. It is good to maximize the browser at the start of the script, so that the script gets executed successfully without any error.

4) findElement ();(It is a unimplemented method of Search contect interface)

It returns a single element

To find the element on the webpage we need to use findElement() method

Example: driver.findElement(By arugment);

findElement method will be identifies element with the help of By class which contains Static method.

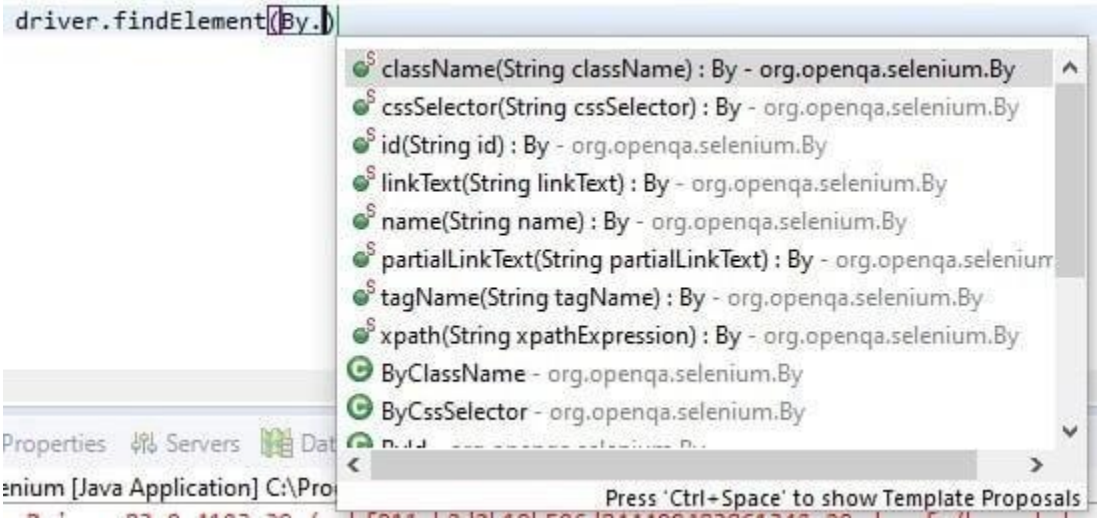
All the static methods present in By class are known as Locators types.

Types of Locators in Selenium

There are various types of locators, using which we can identify a web element uniquely on the Webpage.

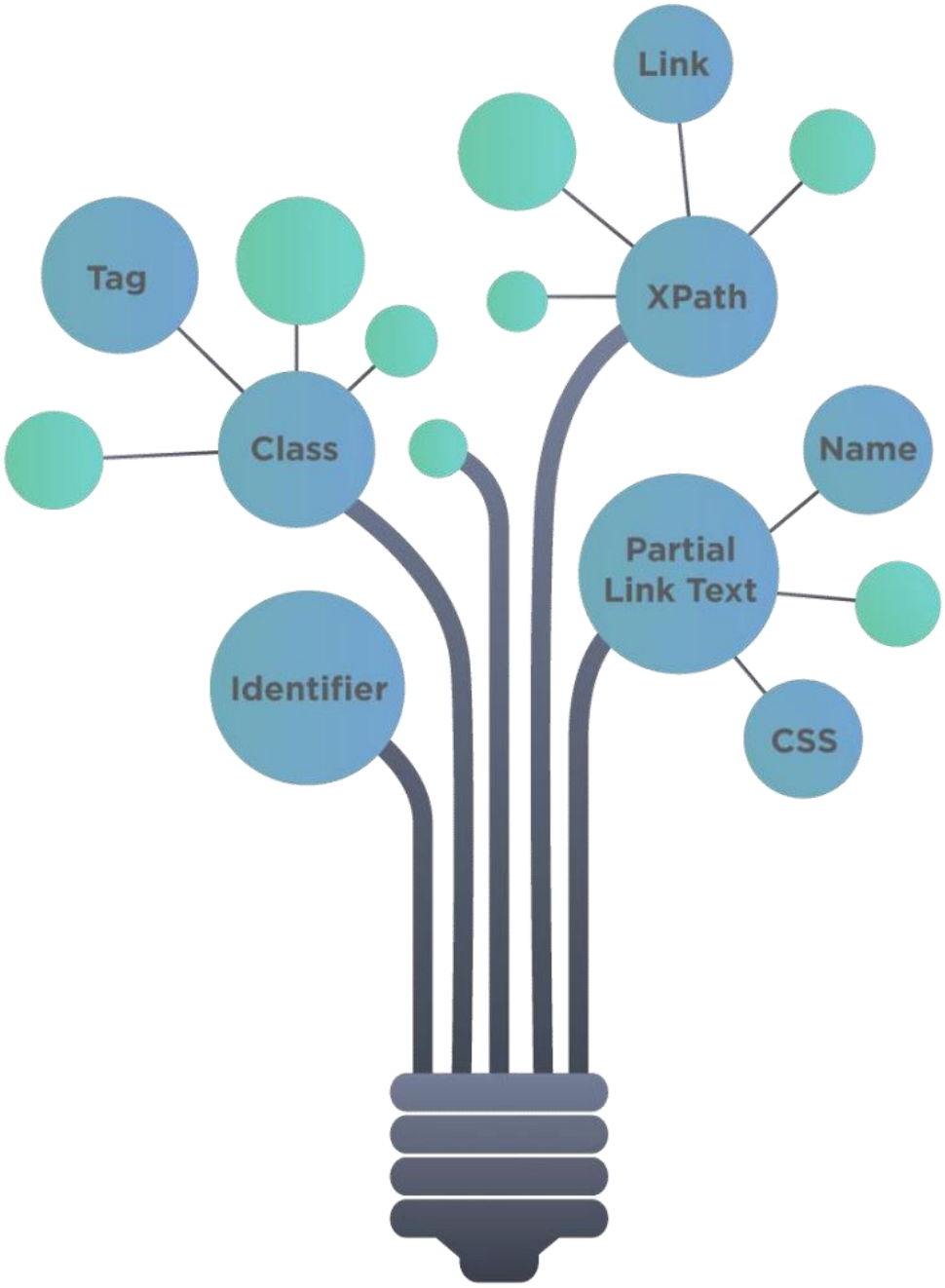
To access all these locators, Selenium provides the "***By***" class, which helps in locating elements within the DOM. It offers several different methods (*some of which are in the image below*) like ***className, cssSelector, id, linkText, name, partialLinkText, taName, and xpath, etc***., which can identify the web elements based on their corresponding locator strategies.

You can quickly identify all the supported locators by Selenium by browsing all the visible methods on the "***By***" class, as shown below:



In above diagram first 8 are Locator elements

The following figure shows a good depiction of several types of locators that Selenium supports.



Selenium Locators

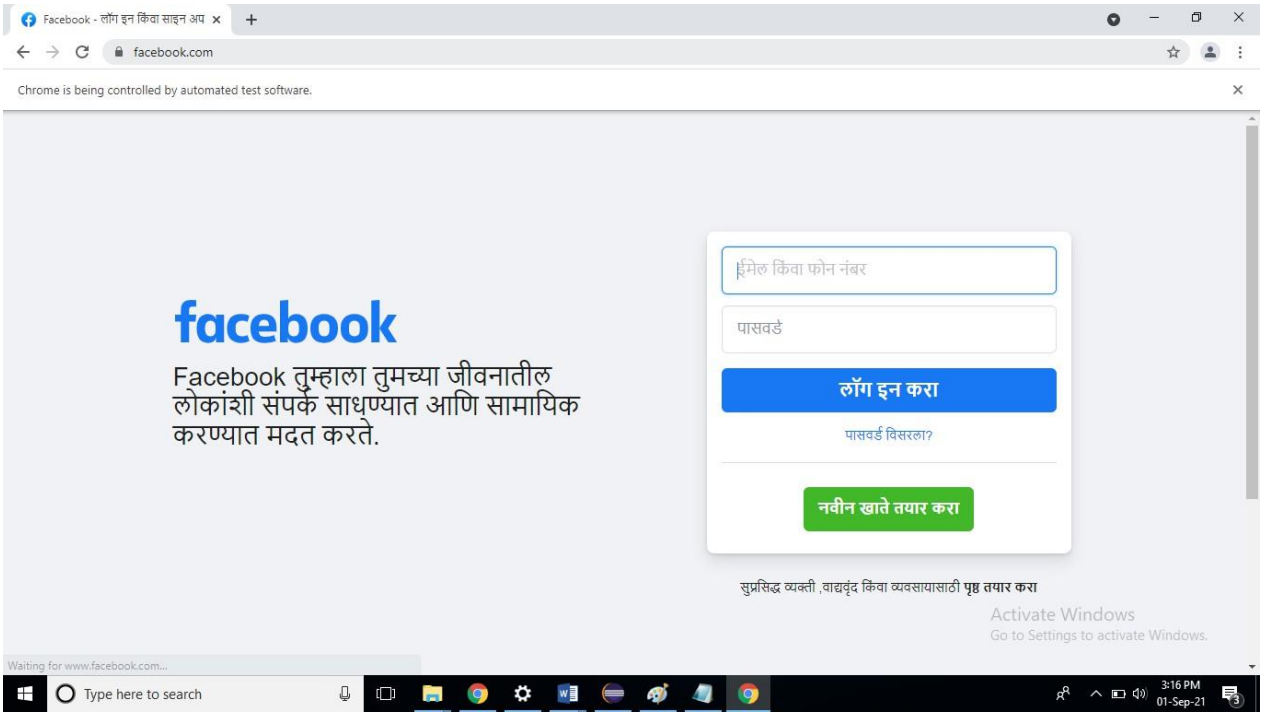
As we can see, Selenium supports the following locators:

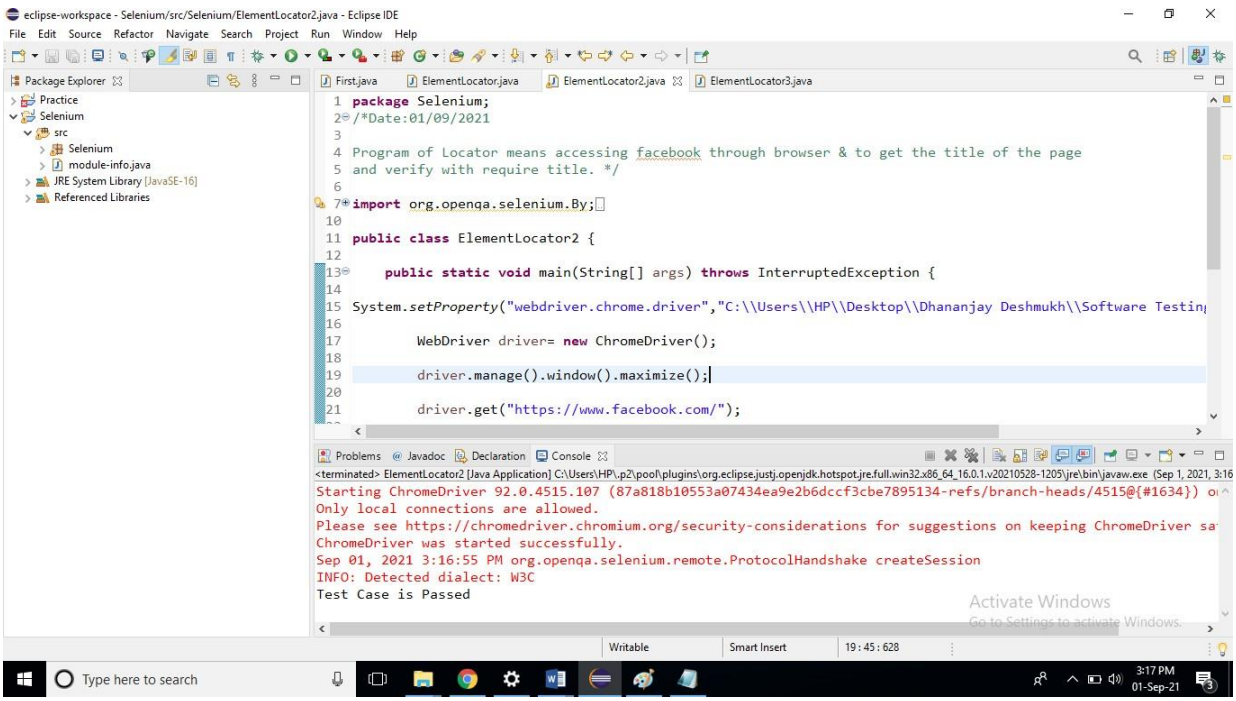
- **ClassName** – A ClassName operator uses a class attribute to identify an object.
- **cssSelector** – CSS is used to create style rules for webpages and can be used to identify any web element.
- **Id** – Similar to class, we can also identify elements by using the 'id' attribute.
- **linkText** – Text used in hyperlinks can also locate element
- **name** – Name attribute can also identify an element
- **partialLinkText** – Part of the text in the link can also identify an element
- **tagName** – We can also use a tag to locate elements
- **xpath** – Xpath is the language used to query the XML document. The same can uniquely identify the web element on any page.

Program1: Program of Locator means accessing facebook through browser & to get the title of the page and verify with require title.

```
package Selenium;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;


public class ElementLocator2 {


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


System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");


		WebDriver driver= new ChromeDriver();
```

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

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

        String title=driver.getTitle();

        if(title.contains("Facebook"))
        {
            System.out.println("Test Case is Passed");
        }
        else
        {
            System.out.println("Test Case is Failed");
        }
        Thread.sleep(5000);//It Executes next statement after 5 seconds
    driver.close();//It close the browser immediately
    }

}
```

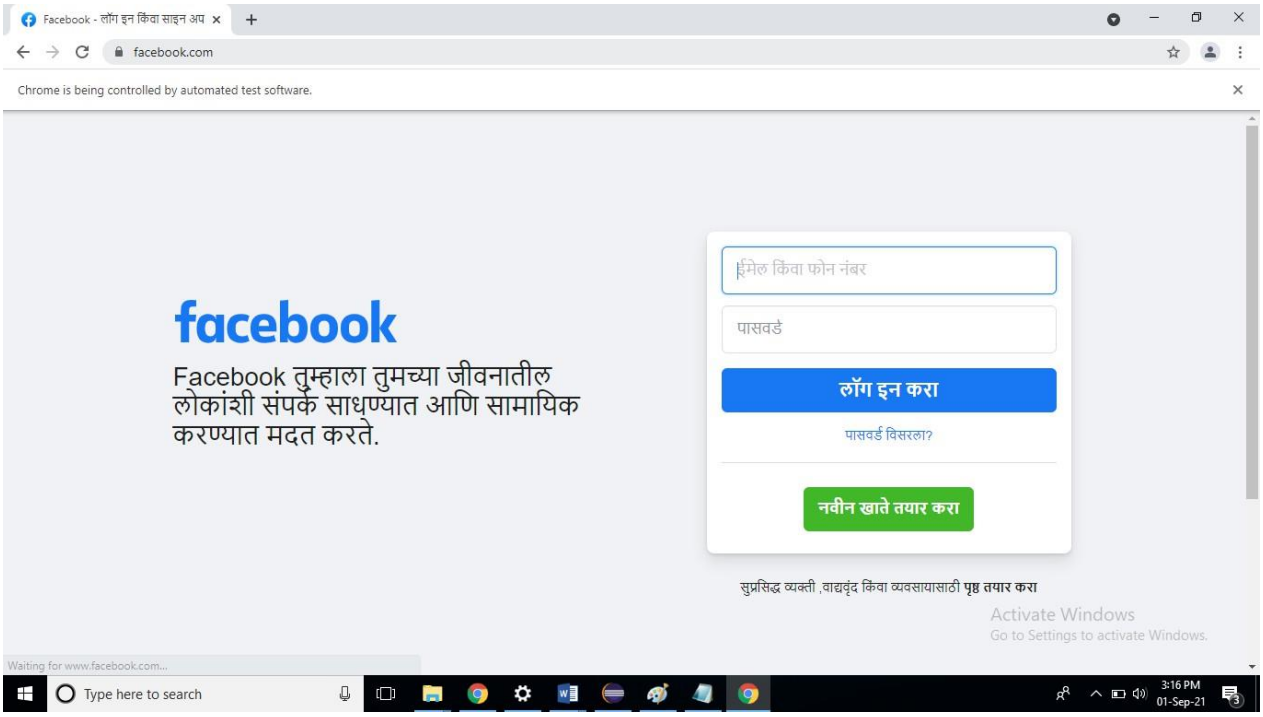

Facebook - लॉग इन किंवा साइन अप
facebook.com
Chrome is being controlled by automated test software.
facebook
Facebook तुम्हाला तुमच्या जीवनातील लोकांशी संपर्क साधण्यात आणि सामायिक करण्यात मदत करते.
ईमेल किंवा फोन नंबर
पासवर्ड
लॉग इन करा
पासवर्ड विसरला?
नवीन खाते तयार करा
सुप्रसिद्ध व्यक्ती ,वाद्यवृंद किंवा व्यवसायासाठी पृष्ठ तयार करा
Activate Windows
Go to Settings to activate Windows.
Waiting for www.facebook.com...
Type here to search
3:16 PM
01-Sep-21

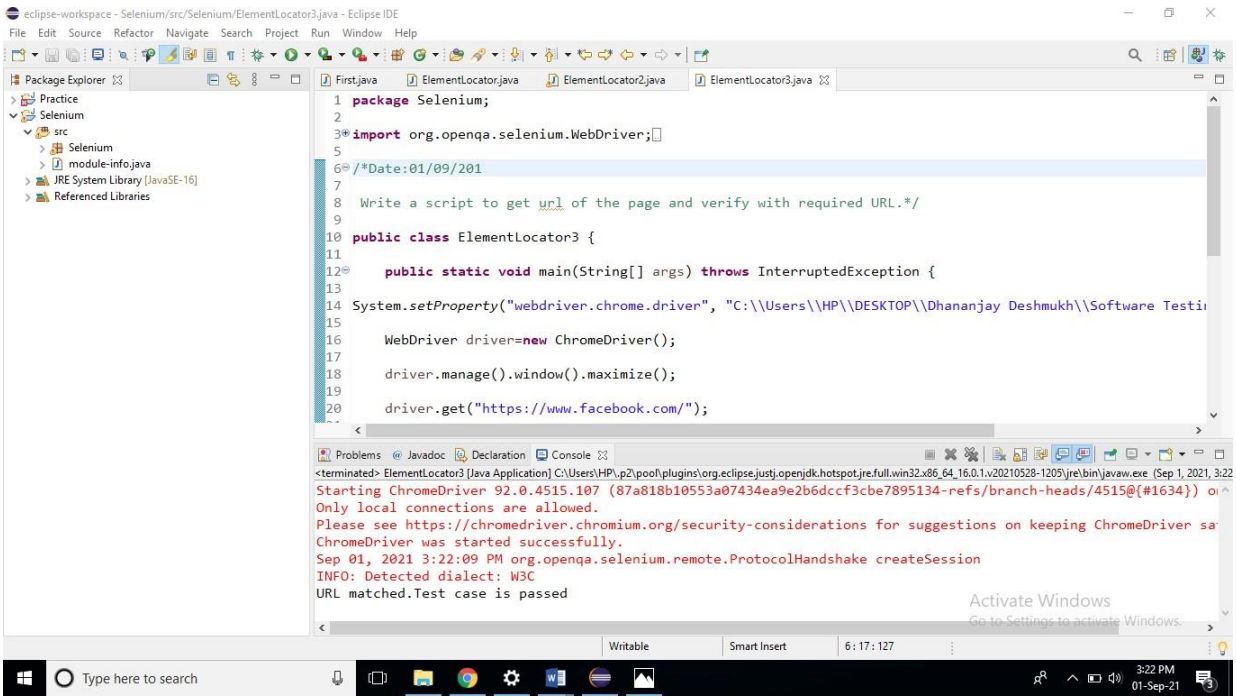

eclipse-workspace - Selenium/src/Selenium/ElementLocator2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Practice
Selenium
src
Selenium
module-info.java
JRE System Library [JavaSE-16]
Referenced Libraries
First.java
ElementLocator.java
ElementLocator2.java
ElementLocator3.java
package Selenium;
/*Date:01/09/2021
Program of Locator means accessing facebook through browser & to get the title of the page
and verify with require title. */
import org.openqa.selenium.By;
public class ElementLocator2 {
public static void main(String[] args) throws InterruptedException {
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testin
WebDriver driver= new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://www.facebook.com/");
Problems Javadoc Declaration Console
<terminated> ElementLocator2 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 1, 2021, 3:16
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634})
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver
ChromeDriver was started successfully.
Sep 01, 2021 3:16:55 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Test Case is Passed
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
19 : 45 : 628
Type here to search
3:17 PM
01-Sep-21

Program2: Write a script to get URL of the page and verify with required URL

```
package Selenium;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class ElementLocator3 {

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

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\DESKTOP\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

   WebDriver driver=new ChromeDriver();

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

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

   String title=driver. getCurrentUrl() ();

   if(title.contains("https://www.facebook.com/"))
   {
	System.out.println("URL matched.Test case is passed");

   }
   else
   {
	System.out.println("URL does not match.Test case is Fail ");
```

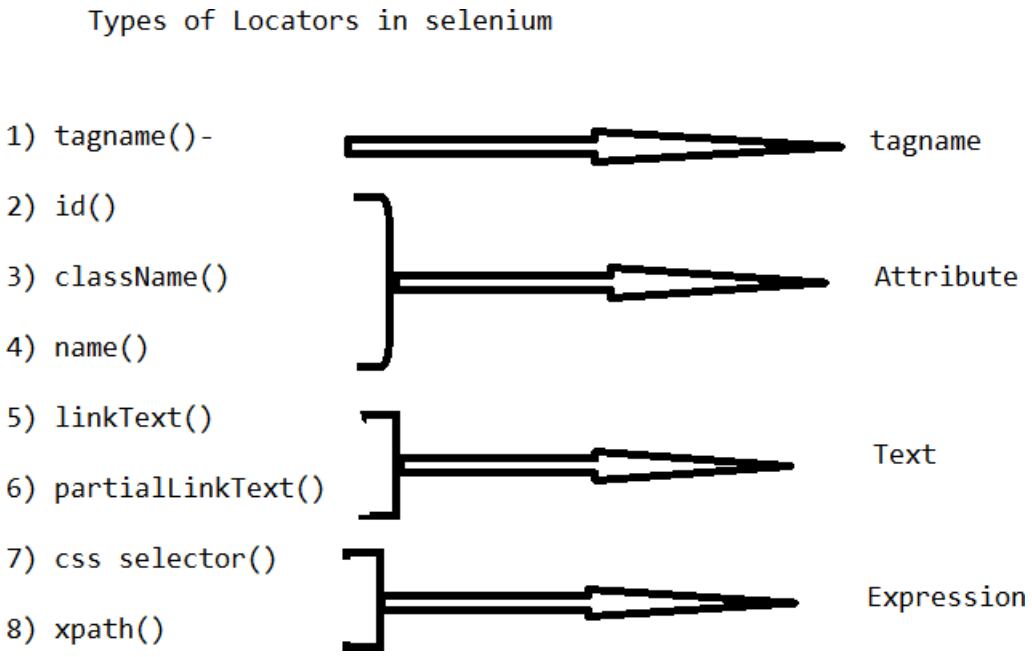
```
    }

    Thread.sleep(5000);//It Executes next statement after 5 seconds
    driver.close();//It close the browser immediately

        }

}
```


Facebook - लॉग इन किंवा साइन अप
facebook.com
Chrome is being controlled by automated test software.
facebook
Facebook तुम्हाला तुमच्या जीवनातील लोकांशी संपर्क साधण्यात आणि सामायिक करण्यात मदत करते.
ईमेल किंवा फोन नंबर
पासवर्ड
लॉग इन करा
पासवर्ड विसरला?
नवीन खाते तयार करा
सुप्रसिद्ध व्यक्ती ,वाद्यवृंद किंवा व्यवसायासाठी पृष्ठ तयार करा
Activate Windows
Go to Settings to activate Windows.
Waiting for www.facebook.com...
Type here to search
3:16 PM
01-Sep-21


eclipse-workspace - Selenium/src/Selenium/ElementLocator3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Practice
Selenium
src
Selenium
module-info.java
JRE System Library [JavaSE-16]
Referenced Libraries
First.java
ElementLocator.java
ElementLocator2.java
ElementLocator3.java
1 package Selenium;
2
3 import org.openqa.selenium.WebDriver;
5
6 /*Date:01/09/201
7
8 Write a script to get url of the page and verify with required URL.*/
9
10 public class ElementLocator3 {
11
12 public static void main(String[] args) throws InterruptedException {
13
14 System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\DESKTOP\\Dhananjay Deshmukh\\Software Testi
15
16 WebDriver driver=new ChromeDriver();
17
18 driver.manage().window().maximize();
19
20 driver.get("https://www.facebook.com/");
Problems Javadoc Declaration Console
<terminated> ElementLocator3 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 1, 2021, 3:22
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634})
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver
ChromeDriver was started successfully.
Sep 01, 2021 3:22:09 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
URL matched.Test case is passed
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
6 : 17 : 127
Type here to search
3:22 PM
01-Sep-21

6) Selenium 6 Lecture (Date: 2 Sep 2021)

Types of Locators



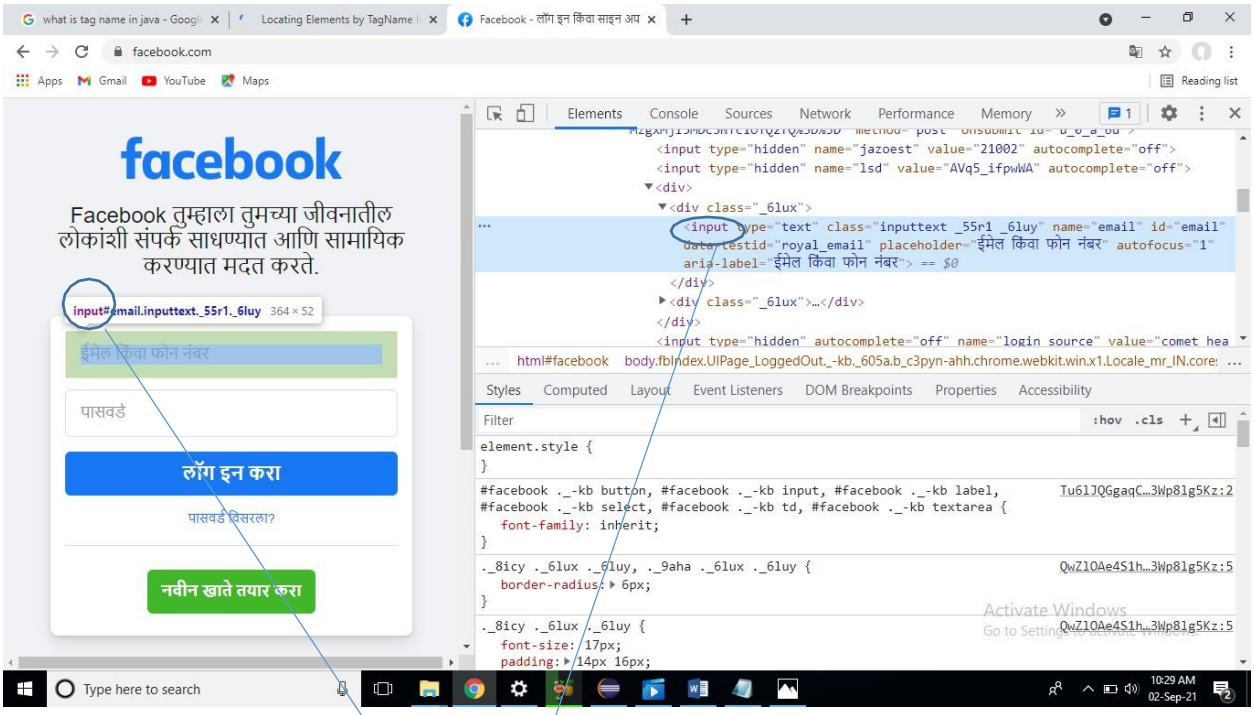
1) tag name():

A tagName is **a part of a DOM structure** where every element on a page is been defined via tag like input tag, button tag or anchor tag etc. Each tag has multiple attributes like ID, name, value class etc.

Below is the DOM structure of LambdaTest login page where I have highlighted the tag names:
Email Field: `< input type="email" name="email" value="" placeholder="Email" required="required" autofocus="autofocus" class="form-control mt-3 form-control-lg">`

Password Field: `< input type="password" name="password" placeholder="Password" class="form-control mt-3 form-control-lg" >`

Login Button: `< button type="submit" class="btn btn-primary btn-lg btn-block mt-3">LOGIN< /button >`
Forgot Password Link: `< button type="submit" class="btn btn-primary btn-lg btn-block mt-3">LOGIN< /button >`

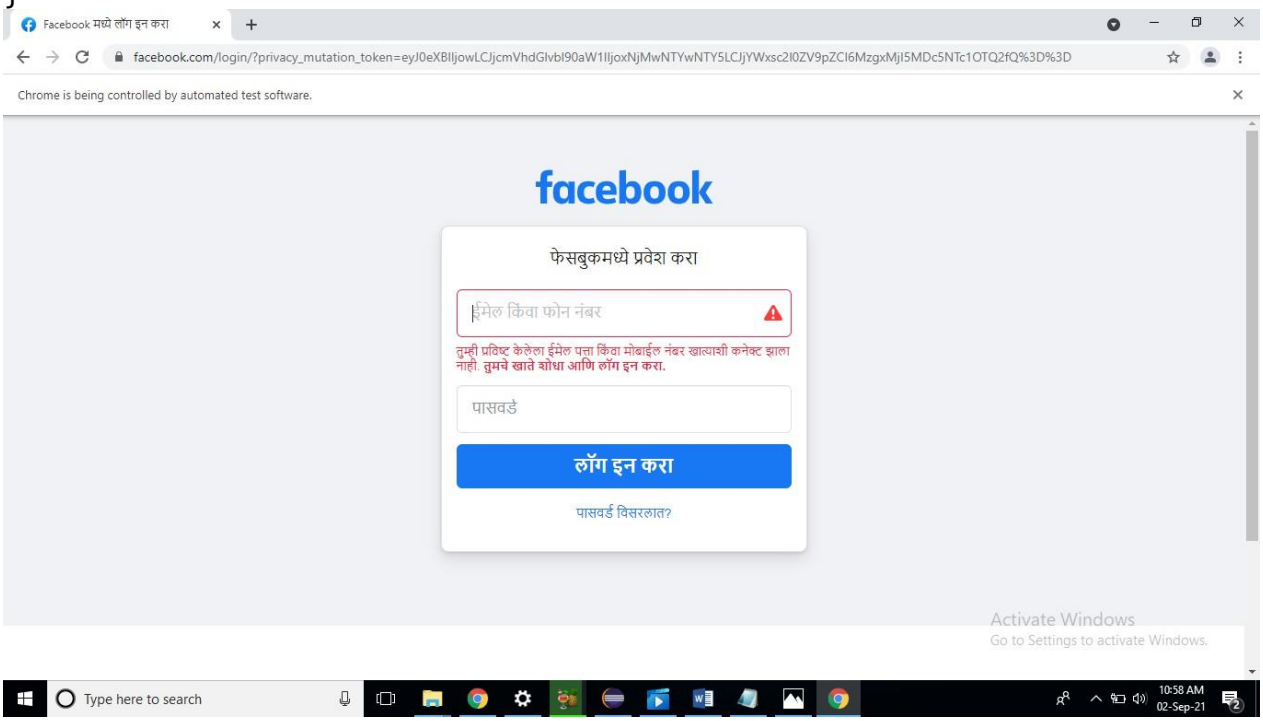


In above diagram **input** is a tag name.

Program: Write a script to click on button by using tag name locator

```
package Selenium;

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

public class ElementLocatorTagName {

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

	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

    WebDriver driver=new ChromeDriver();

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

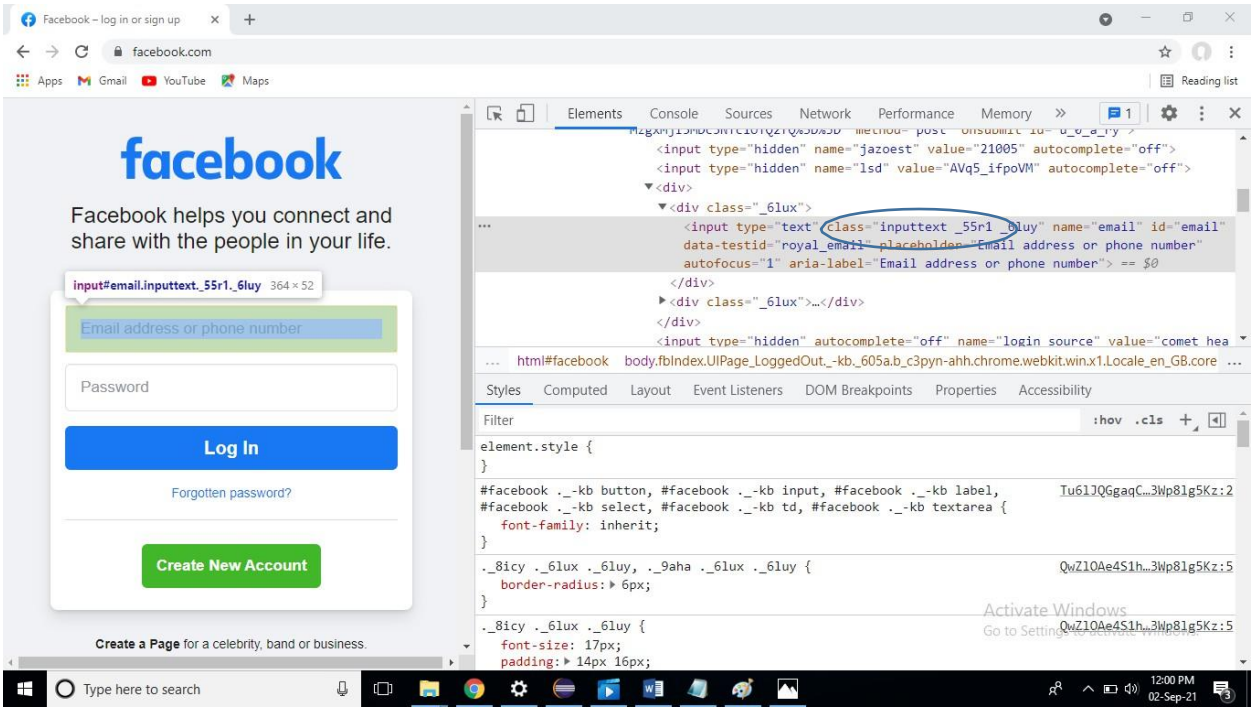
```
    driver.get("https://www.facebook.com/");

    driver.findElement(By.tagName("button")).click();

    Thread.sleep(5000);
    driver.close();
      }
}
```



2) class name():

The class name is stored in the class attribute of an HTML tag. By design, a CSS class applies to a group of DOM elements. The *.find_element_by_class_name()* method only returns the first element with the matching class. It raises a *NoSuchElementException* if there is no element with the given class name.

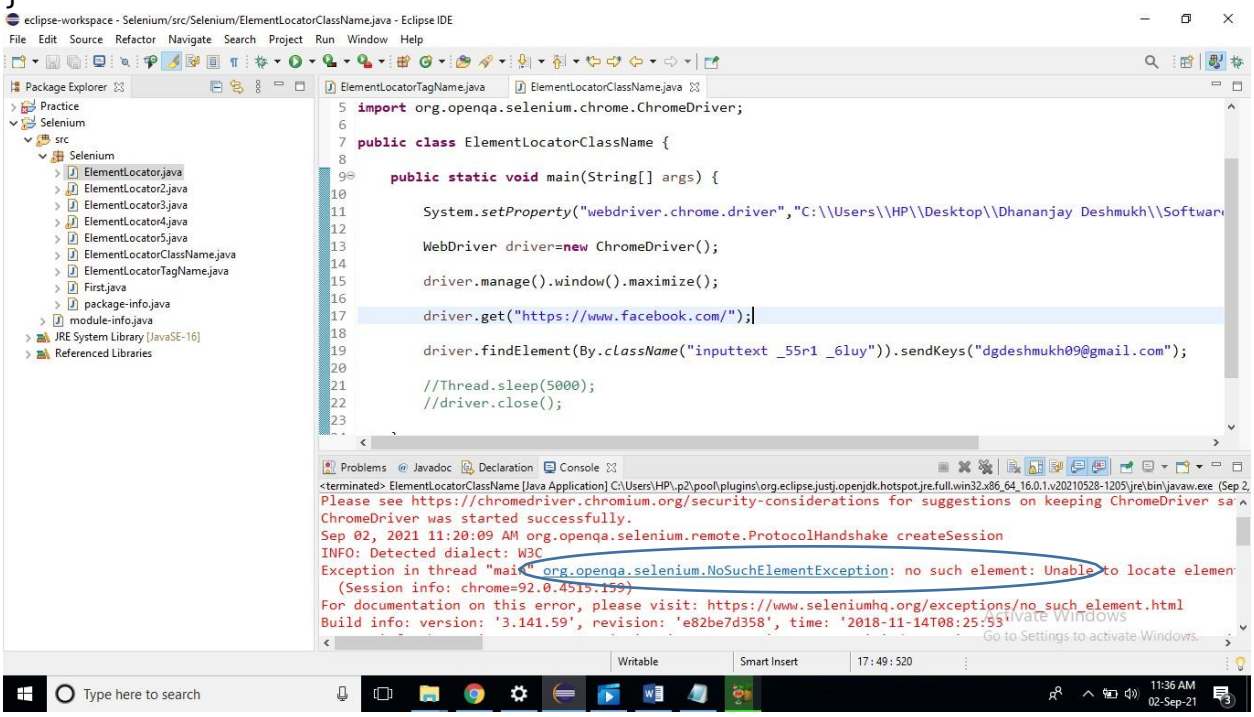
In this case, you can opt to locate element by CSS selector in Selenium or by XPath in Selenium using the class name attribute.



Program: Write a script to insert emaild id by using Class name locator

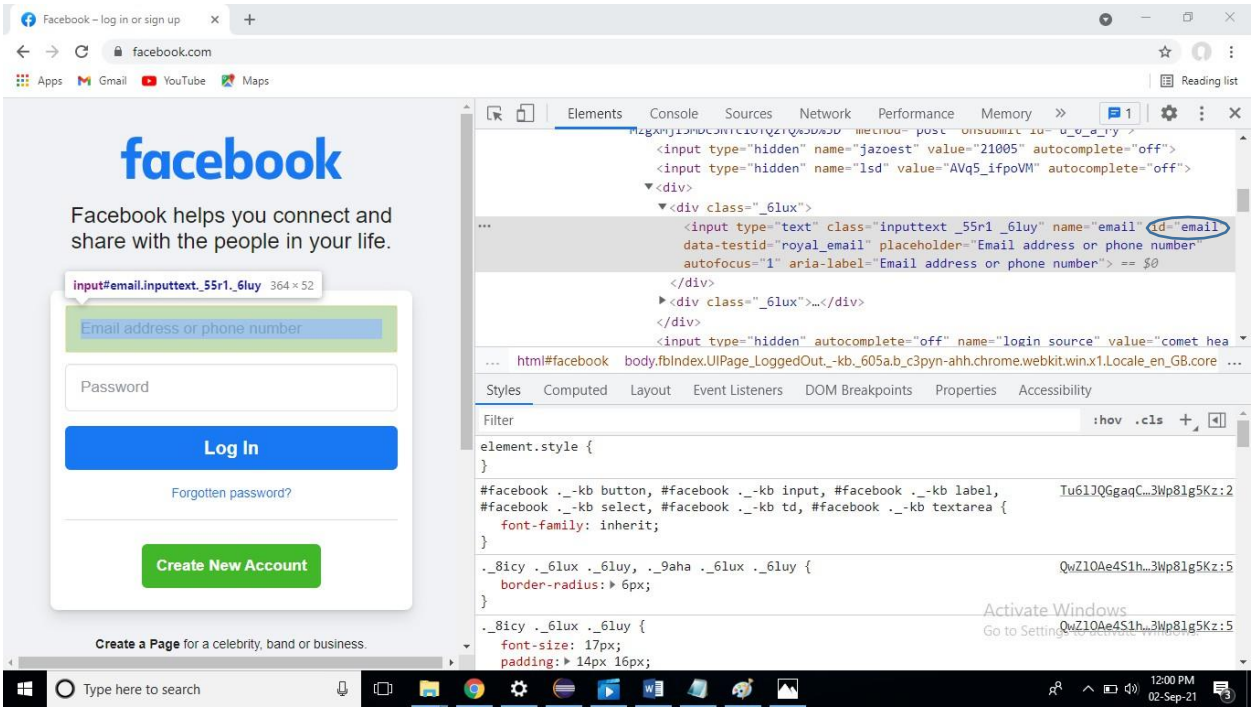
```
package Selenium;

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

public class ElementLocatorClassName {

	public static void main(String[] args) {


	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\SoftwareTesting\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

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

		driver.findElement(By.className("inputtext _55r1 _6luy")).sendKeys("dgdeshmukh09@gmail.com");
```

```
        //Thread.sleep(5000);
        //driver.close();

    }
}
```



3) Id():

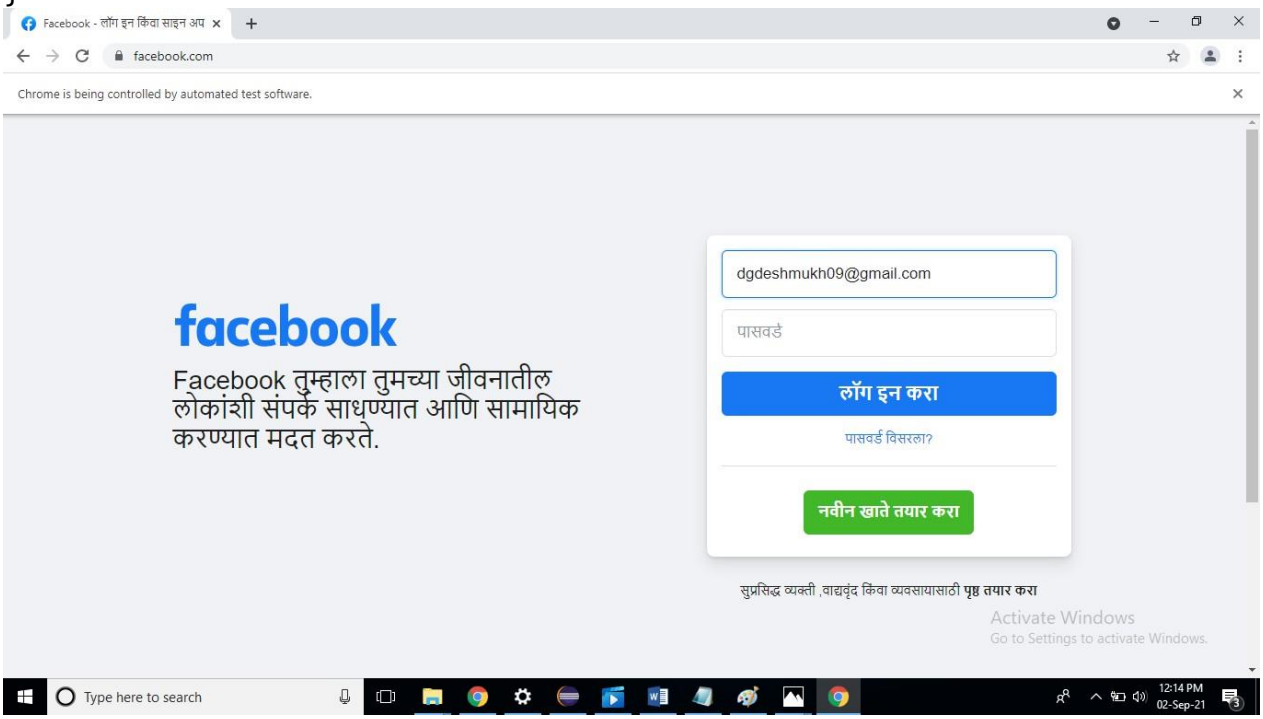
ID's are unique for each element so it is common way to locate elements using ID Locator. As per W3C, ID's are supposed to be unique on a page and it makes ID's are the most reliable locator. ID locators are the fastest and safest locators out of all locators.



Program: Write a script to insert emaild id by using Id locator

```
package Selenium;
/*Date:02/09/2021
Write a script to insert emaild id by using Class name locator*/

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

public class ElementLocatorId {

	public static void main(String[] args) {


	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

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

```
driver.findElement(By.id("email")).sendKeys("dgdeshmukh09@gmail.com");

          //Thread.sleep(5000);
          //driver.close();

      }

}
```



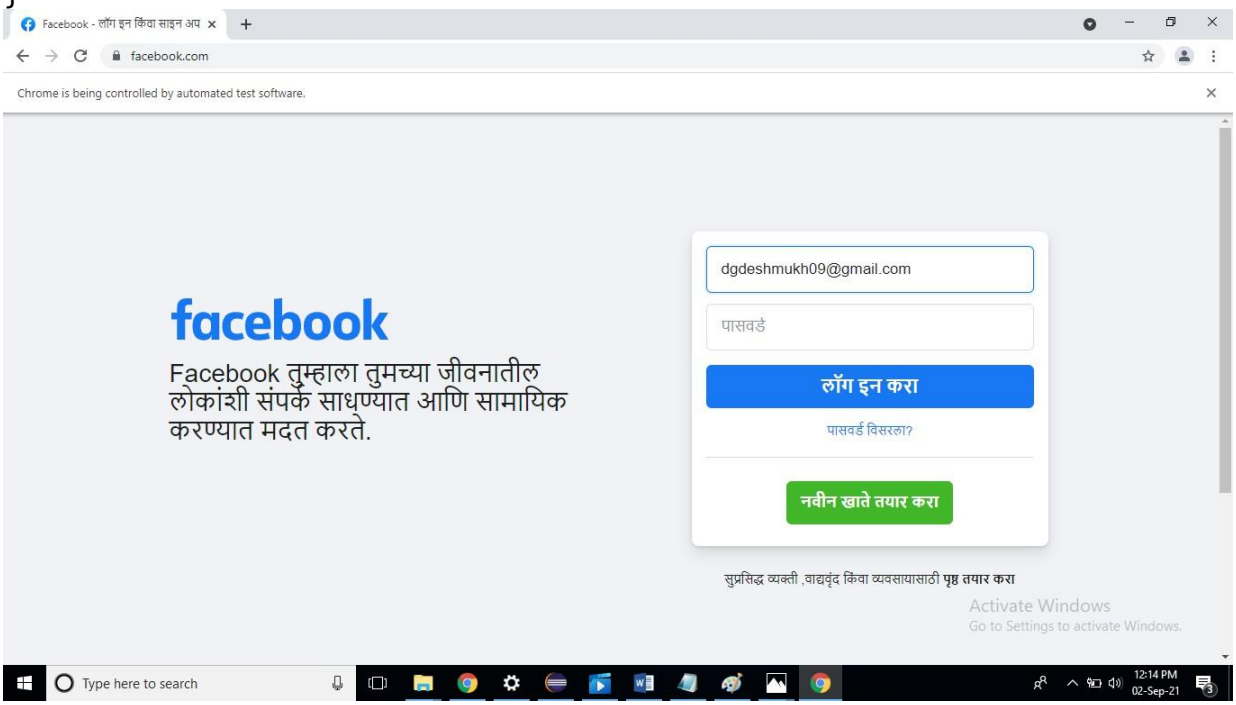
4) name():

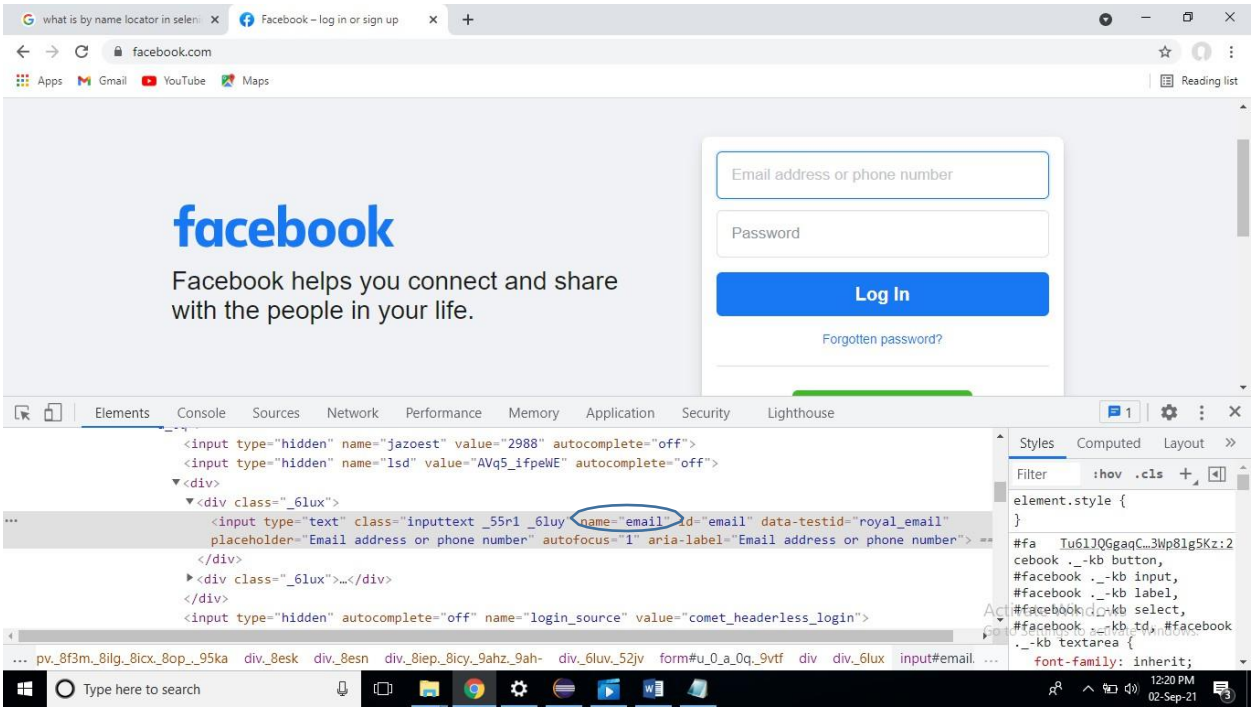
By.name Selenium locator is **used to identify the elements of a webpage**. This attribute can be mentioned as part of multiple tags like < input >, < button >, < select > etc. Unlike ID, this may or may not be unique to a page. A webpage may contain multiple tags with the same By.name attribute value.

Program: Write a script to insert emaild id by using name locator

```
package Selenium;
/*Date:02/09/2021
Write a script to insert emaild id by using name locator*/

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

```java
public class ElementLocatorName {

	public static void main(String[] args) {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

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


driver.findElement(By.name("email")).sendKeys("dgdeshmukh09@gmail.com");

		//Thread.sleep(5000);
		//driver.close();

	}

}
```



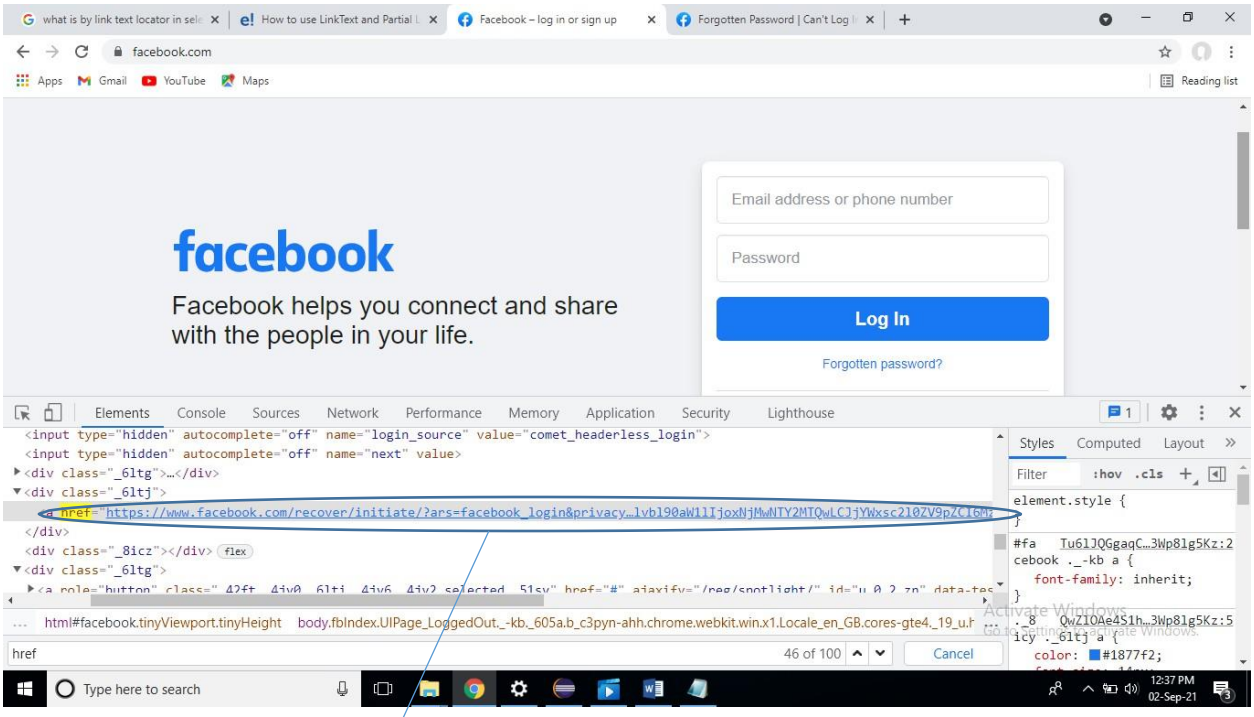


5) LinkText():

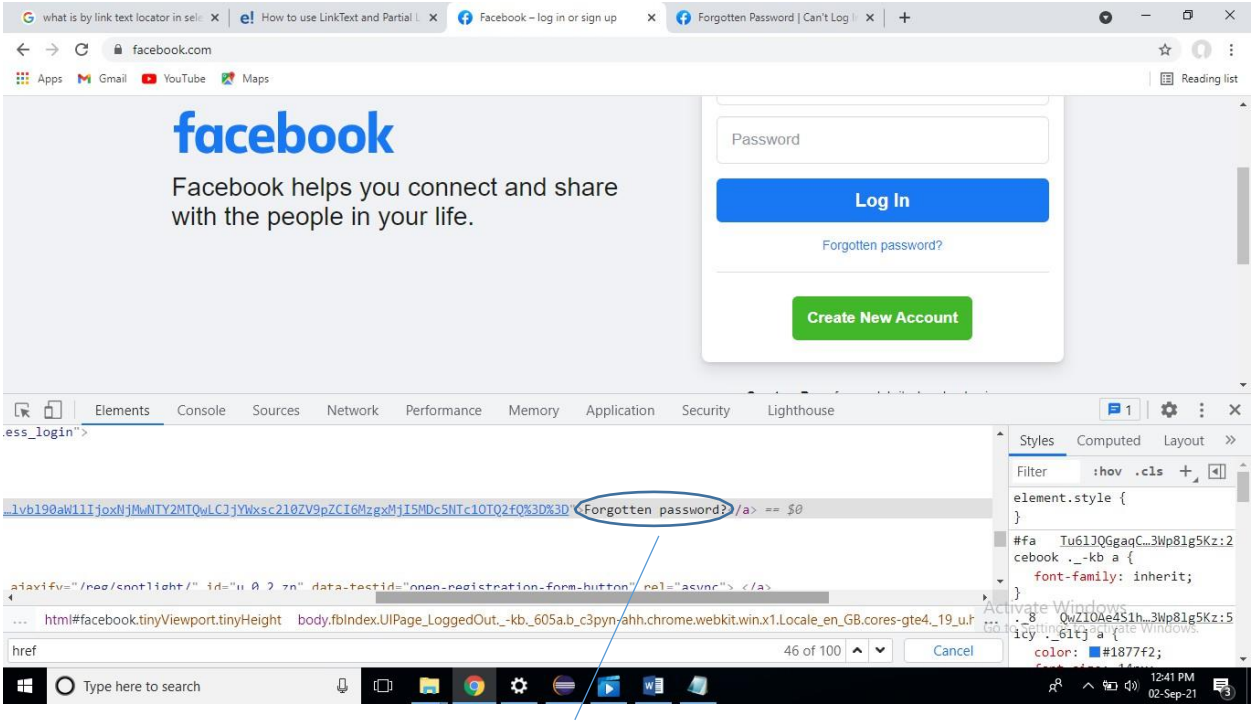
A **linkText** is used to identify the hyperlinks on a web page.

We can handle the link in web page by using linkText.

We are identifying & we are locating the elements based on it’s link text.



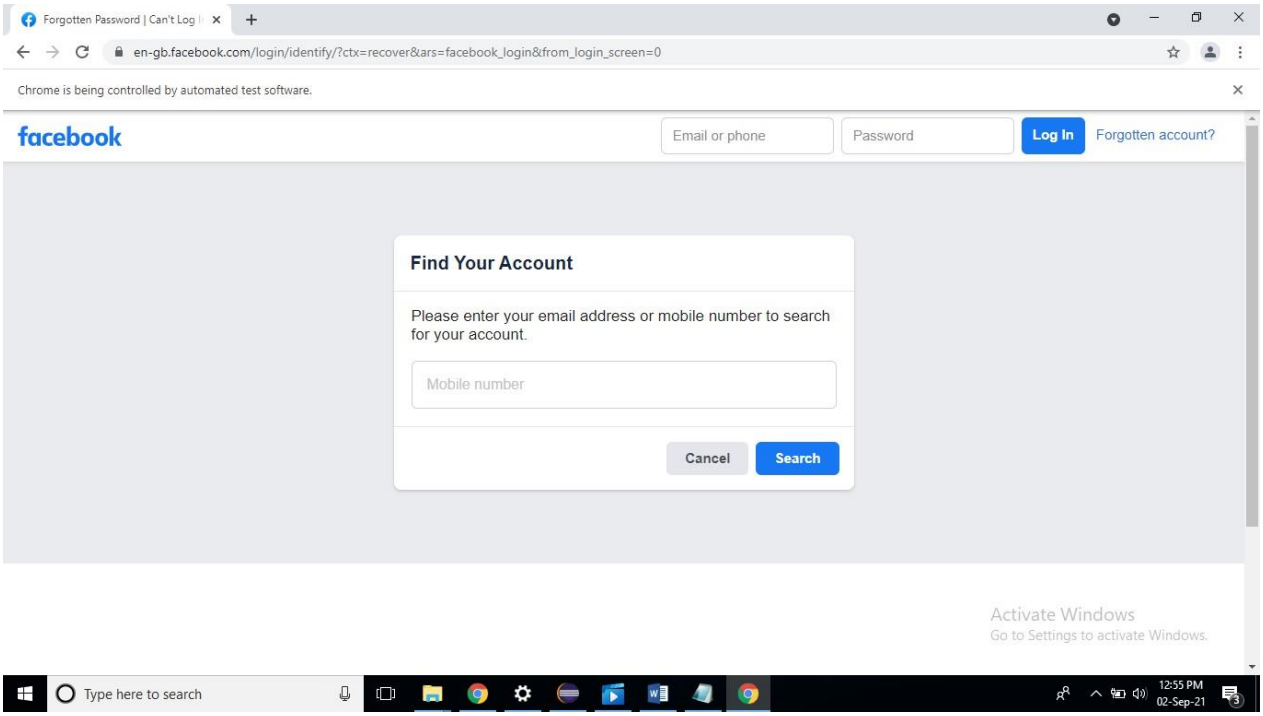
Link



Link Text

Program: Write a script to locate the elements by using it's link text.

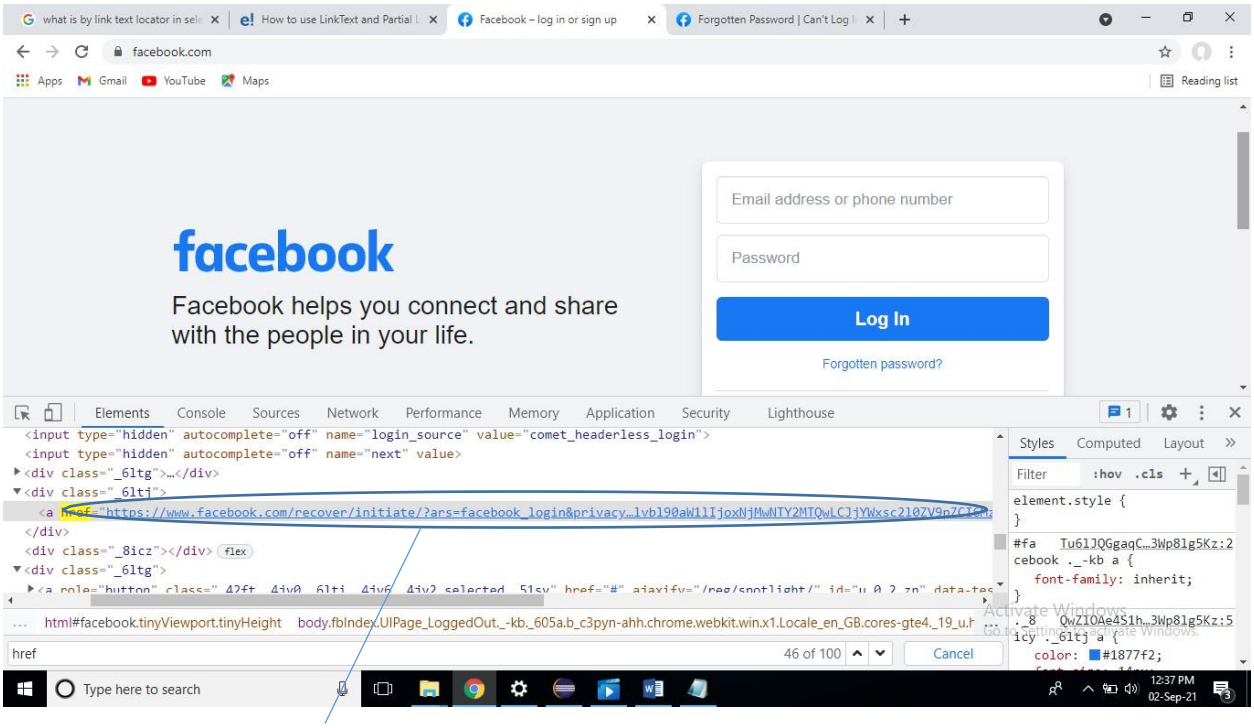
```
package Selenium;

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

/* Date:02/09/2021
 Program: Write a script to locate the elements based on it's link
text.*/

public class ElementLocatorLinkText {

	public static void main(String[] args) {
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://en-gb.facebook.com/");

		driver.findElement(By.linkText("Forgotten
password?")).click();



	}

}
```



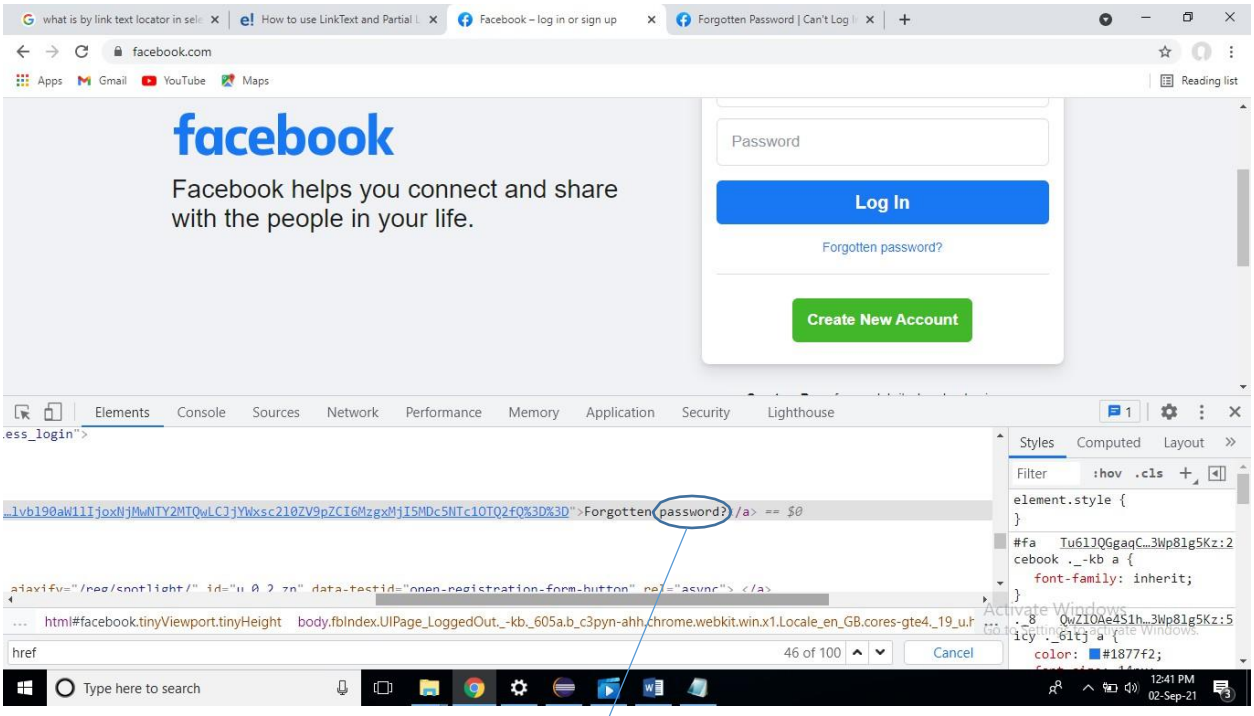
6) PartialLinkText():

Partial link text in Selenium is **another way of locating an element via a link**. The only difference from the link text in Selenium to partial link text is that it does not look into the exact match of the string value but works on a partial match.

In above example if we pass argument as only **Password** then also it will give same result as LinkText().



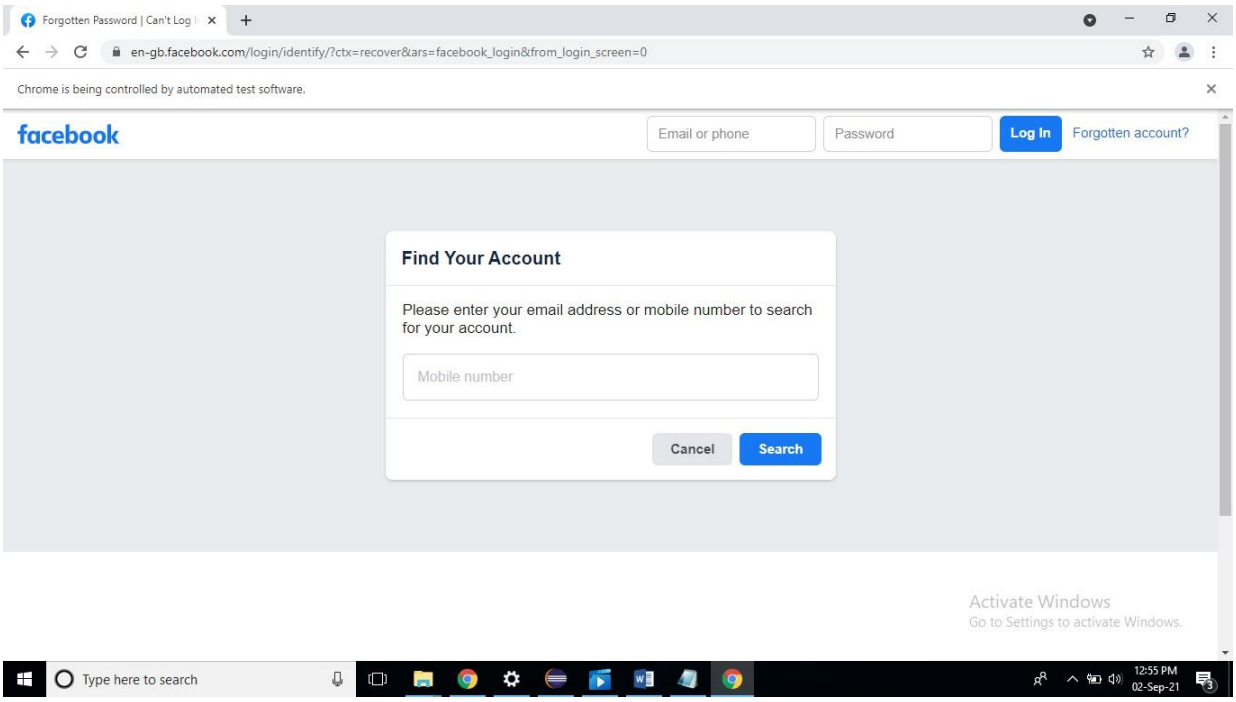
Link



Partial Link Text

Program: Write a script to locate the elements by using it's Partial link text

```
package Selenium;

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

/* Date:02/09/2021
Program: Write a script to locate the elements based on it's link
text.*/

public class ElementLocatorPartialLinkText {

	public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://en-gb.facebook.com/");

		driver.findElement(By.partialLinkText("password")).click();


	}

}
```


Forgotten Password | Can't Log
en-gb.facebook.com/login/identify/?ctx=recover&ars=facebook_login&from_login_screen=0
Chrome is being controlled by automated test software.
facebook
Email or phone
Password
Log In
Forgotten account?
Find Your Account
Please enter your email address or mobile number to search for your account.
Mobile number
Cancel
Search
Activate Windows
Go to Settings to activate Windows.
Type here to search
12:55 PM
02-Sep-21

7) Selenium 7 Lecture

6)CSS.Selctor():

It is the kind of Expression(combination of different attributes). Expression can be statements.

CSS Selector combines an element selector and a selector value that can identify particular elements on a web page. Like XPath, CSS selector can be used to locate web elements without ID, class, or Name.

CSS Selectors in Selenium are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selectors in Selenium is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.

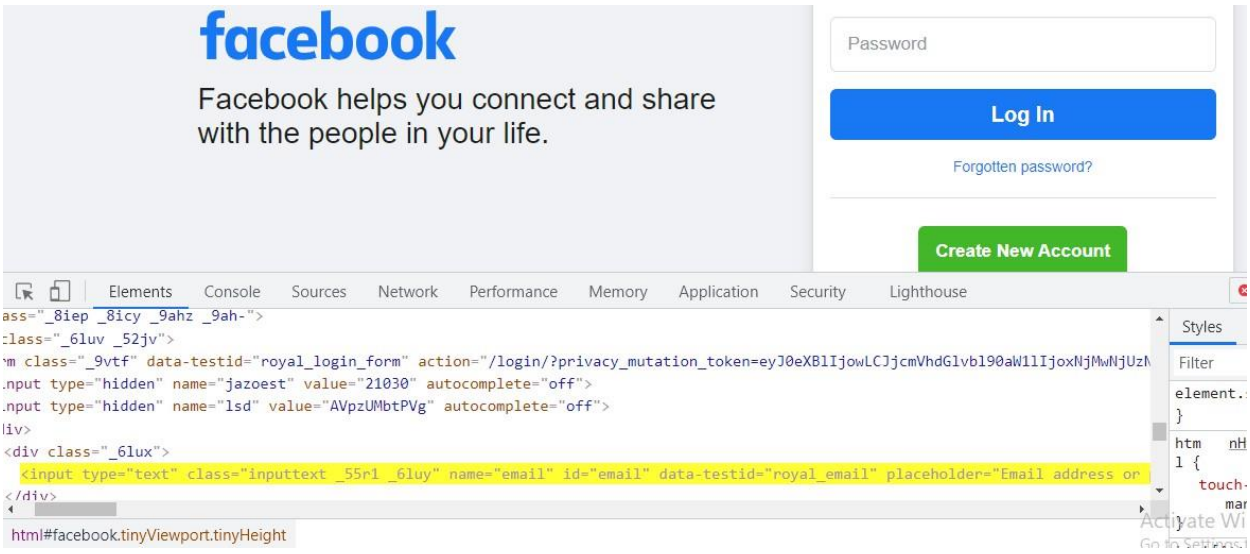
CSS Selectors in Selenium have many formats, but we will only focus on the most common ones.

- Tag and ID
- Tag and class
- Tag and attribute
- Tag, class, and attribute
- Inner text

When using this strategy, we always prefix the Target box with “css=” as will be shown in the following examples.

Example: When we want to append tag name with id then we search in Facebook email id inspect element like

Input#email then enter then we will get unique result.



Locating by CSS Selector – Tag name and ID

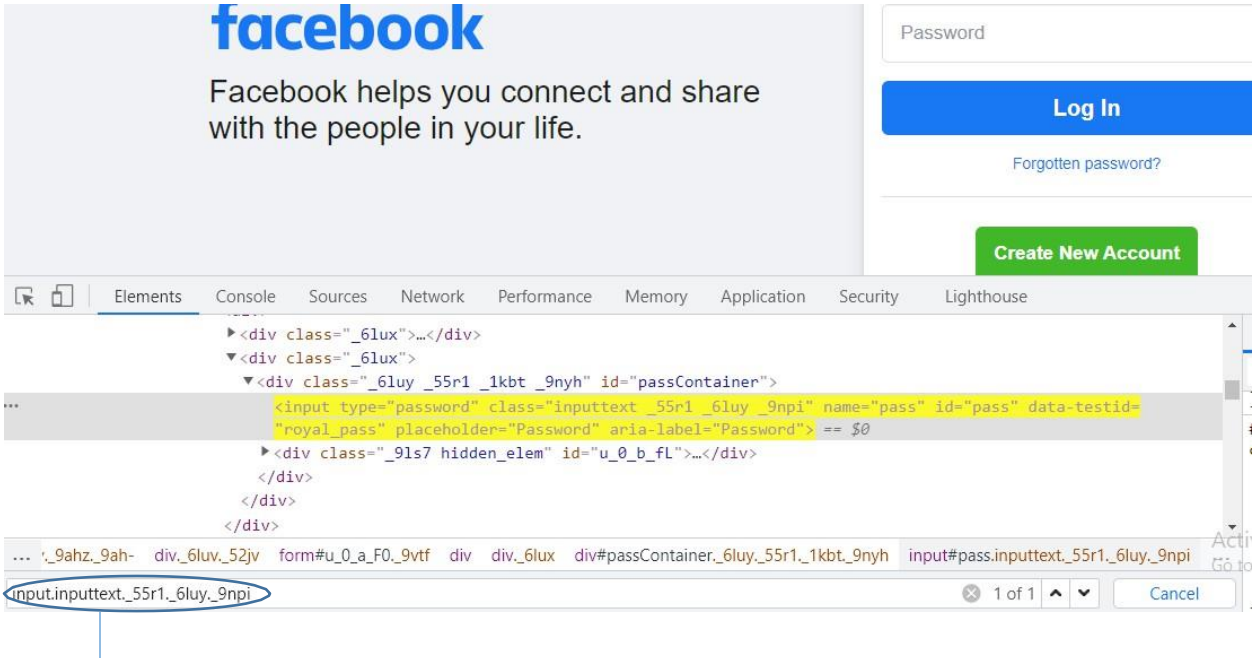
ID is represented as # in Css Selector.

Program: Write a script to locate the elements by using Css selector(Tag Name & ID)

```
package Selenium;

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

/* Date:03/09/2021
Program: Write a script to locate the elements based on it's Css
selector.*/


public class ElementLocatorCSSSelector {

	public static void main(String[] args) {


	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://en-gb.facebook.com/");


driver.findElement(By.cssSelector("input#email")).sendKeys("dgdeshmukh
09@gmail.com");

	}

}
```

Locating by CSS Selector – Tag name and Class Name

Class Name is represented as . (dot) in Css Selector.

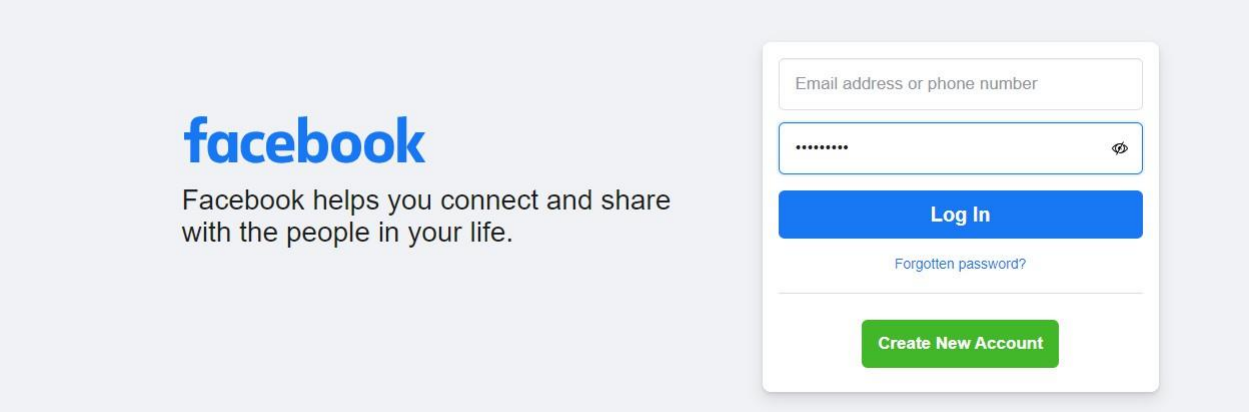


We have to add . (dot) before space in class name while searching & Script also.

Program: Write a script to locate the elements by using Css selector(Tag Name & Class Name)

```
package Selenium;

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

/* Date:03/09/2021
Program: Write a script to locate the elements based on it's Css
selector (Tag Name & Class name).
We have to add . (dot) before space in class name*/


public class ElementLocatorCSSSelector2 {

	public static void main(String[] args) {
```

```
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

          WebDriver driver=new ChromeDriver();

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

          driver.get("https://en-gb.facebook.com/");


driver.findElement(By.cssSelector("input.inputtext._55r1._6luy._9npi")
).sendKeys("dhanu9999");



     }

}
```



Locating by CSS Selector – Tag name and attributes

Write Attribute in [] after tag name. like **input[type="text"]** in searching & in program also.

Program: Write a script to locate the elements by using Css selector(Tag Name Attribute)

```
package Selenium;

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

```
/* Date:03/09/2021
Program: Write a script to locate the elements based on it’s Css
selector (Tag Name & Attribute).*/

public class ElementLocatorCSSSelector3 {

	public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://en-gb.facebook.com/");


driver.findElement(By.cssSelector("input[type=\"text\"]")).sendKeys("d
gdeshmukh09@gmail.com");



	}

}
```

Locating by CSS Selector – Tag name and class name and two attributes

Attributes:1)type 2)name

Write Attribute in [] after tag name.

input.inputtext._55r1._6luy[name="email"][type="text"] in searching & in program also.

Program: Write a script to locate the elements by using Css selector(Tag Name & Class Name & Atrribute)

```
package Selenium;

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

/* Date:03/09/2021
Write a script to locate the elements based on it's Css selector (Tag
Name & Class name& Attribute).*/

public class ElementLocatorCSSSelector4 {

	public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://en-gb.facebook.com/");


driver.findElement(By.cssSelector("input.inputtext._55r1._6luy[name=\"
email\"][type=\"text\"]")).sendKeys("dgdeshmukh09@gmail.com");



	}

}
```

8) xpath():

Xpath is a Expression which is like query language & it is used to travel in the HTML page through the nodes.

XPath is a technique in Selenium **to navigate through the HTML structure of a page**. XPath enables testers to navigate through the XML structure of any document, and this can be used on both HTML and XML documents.

There are two types of Xpath

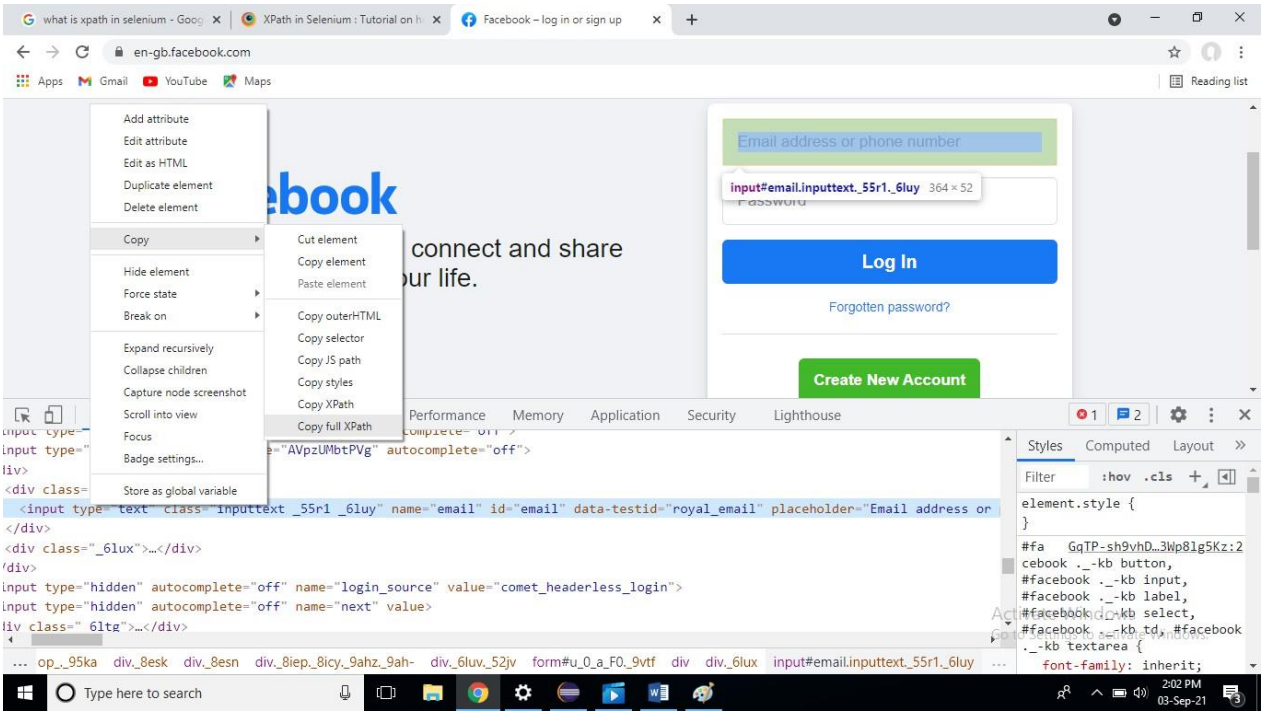
i) Absolute xpath

ii) Relative xpath

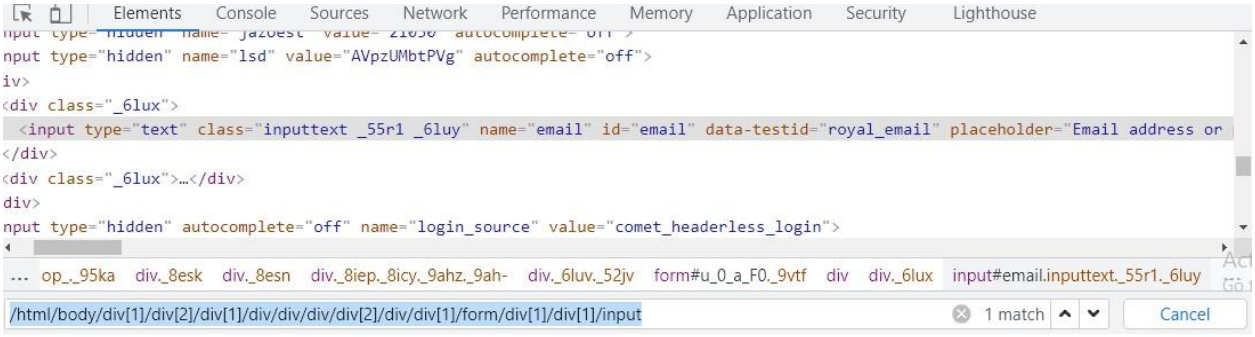
i) Absolute xpath:

Absolute xpath starts with single forward slash (/)

Syntax:/html/body/div[1]/div[2]/div[1]/div/div/div/div[2]/div/div[1]/form/div[1]/div[1]/input



Paste copied full xpath in search



As shown in above diagram we are traveling from starting node (html) to current node

The problem with Absolute xpath is:

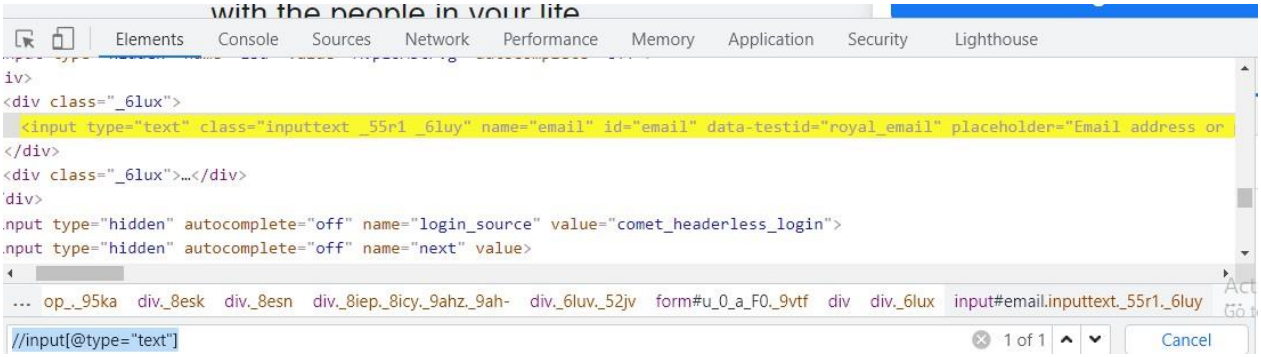
We have added full xpth in our code but developer remove some portion(div) in that html then selenium script will fail therefore Absolute xpath is not recommended to use in Automation Testing.

ii) Relative xpath:

Relative xpath overcomes the problem of Absolute xpath.

Relative xpath starts with double forward slash (//)

Syntax:// tagname[@attribute='value']



Program: Write a script to lcate the elemnts by using Xpath(Relative xpath)

```
package Selenium;

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

/* Date:03/09/2021
Write a script to locate the elements using Xpath(Relative Xpath)*/
```

```java
public class ElementLocatorXpath {

	public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://en-gb.facebook.com/");


driver.findElement(By.xpath("//input[@type=\"text\"]")).sendKeys("dgde
shmukh09@gmail.com");



	}

}
```

8)Selenium 8 Lecture

Selenium Web Element

1)Scrolling operation on Window by using **Java Script Executor Interface**

- JavascriptExecutor is an interface that is used to execute JavaScript with Selenium. To simplify the usage of JavascriptExecutor in Selenium, think of it as a medium that enables the WebDriver to interact with HTML elements within the browser. JavaScript is a programming language that interacts with HTML in a browser, and to use this function in Selenium, JavascriptExecutor is required.

JavaScriptExecutor Methods

1. **executeAsyncScript**

With Asynchronous script, your page renders more quickly. Instead of forcing users to wait for a script to download before the page renders. This function will execute an asynchronous piece of JavaScript in the context of the currently selected frame or window in Selenium. The JS so executed is single-threaded with a various callback function which runs synchronously.

2. **executeScript**

This method executes JavaScript in the context of the currently selected frame or window in Selenium. The script used in this method runs in the body of an anonymous function (a function without a name). We can also pass complicated arguments to it.

The script can return values. Data types returned are

- Boolean
- Long
- String
- List
- WebElement.

The basic syntax for JavascriptExecutor is given below:

Syntax:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript(Script,Arguments);
```

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

```
[java]

js.executeScript("document.getElementByID('element id ').click();");

[/java]
```

- **JavascriptExecutor in Selenium to send text**

```
[java]

js.executeScript("document.getElementByID('element id ').value = 'xyz';");

[/java]
```

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

```
[java]

js.executeScript("document.getElementByID('element id ').checked=false;");

[/java]
```

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

```
[java]

js.executeScript("location.reload()");

[/java]
```

Question: Where Selenium all classes are available in which package?

Answer: Selenium openqa package (`import org.openqa.selenium.By;`)

For using JavaScriptExecutor Interface in Selenium Script

We can't create object of Interface therefore

We have to upcast reference of JavaScriptExecutor in the reference of Webdriver as follows:

JavascriptExecutor js=(JavascriptExecutor) driver;//Step5

js.executeScript("window.scrollBy(0,300"): //Step 6

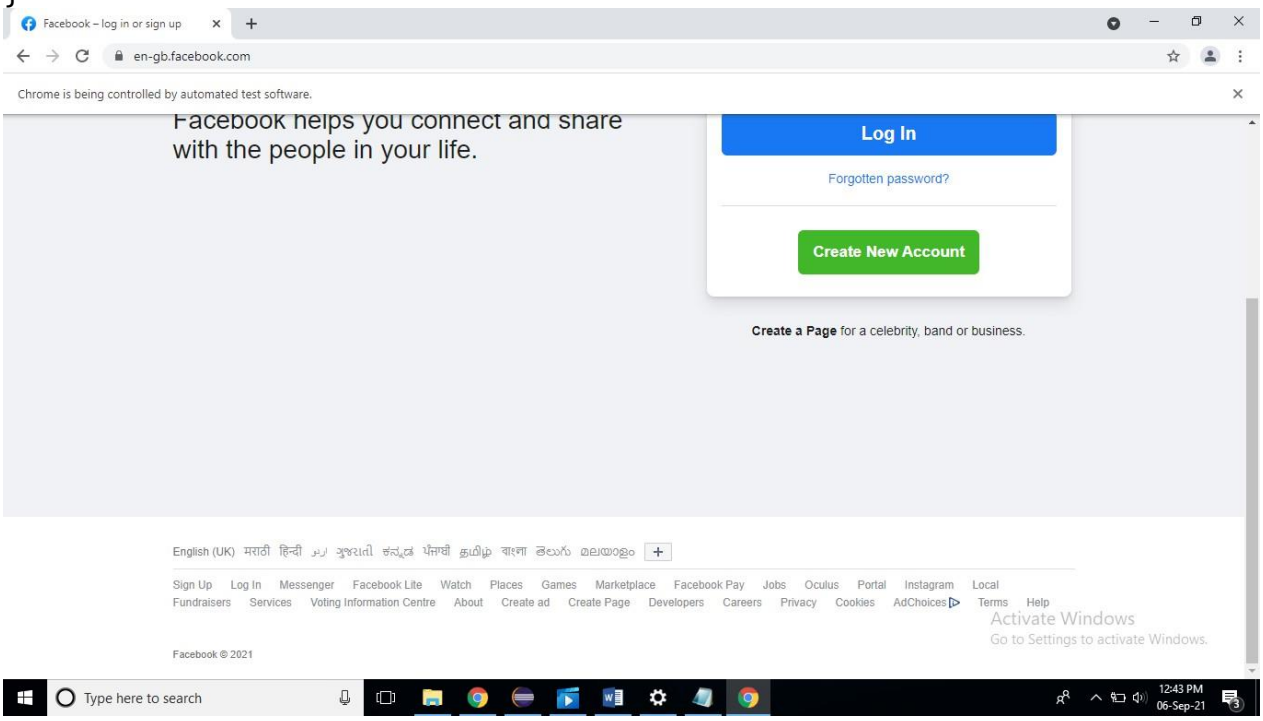
Web page is always designed in the form of X & Y Co-ordinates(Pixels).

X Co-ordinates are Horizontal & Y Co-ordinates are Vertical.

Program: Selenium Script to scroll down a web page using Java Script Executor.

```
package SeleniumWebElement;

/*Date: 06-09-2021
Program to Scroll down a web page using Java Script Executor.*/

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

public class WebElementScrollDown {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

          //Step-1
          WebDriver driver =new ChromeDriver();

          //Step-2
          driver.manage().window().maximize();

          //Step-3
          driver.get("https://en-gb.facebook.com/");
          Thread.sleep(4000);
```

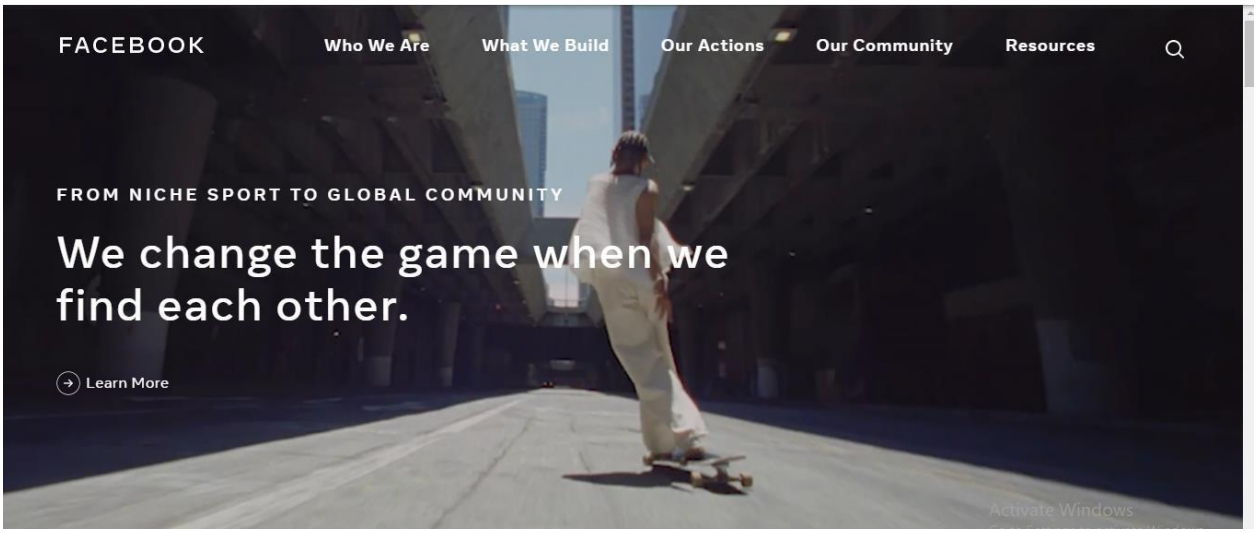
```
        //Step-4 Scroll down window
        JavascriptExecutor js=(JavascriptExecutor) driver;
        js.executeScript("window.scrollBy(0,300)");

        Thread.sleep(4000);
        driver.close();
         }
}
```



Program 2: Selenium Script to scroll down a facebook web page & click on About link on facebook web page using Java Script Executor.

```
package SeleniumWebElement;

import org.openqa.selenium.By;

/*Date: 06-09-2021
Selenium Script to scroll down a facebook web page
& click on About link on facebook web page using Java Script
Executor.*/

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

```
public class WebElementScrollDown2 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		//Step-1
		WebDriver driver =new ChromeDriver();

		//Step-2
		driver.manage().window().maximize();

		//Step-3
		driver.get("https://en-gb.facebook.com/");
		Thread.sleep(4000);


		//Step-4 Scroll down window
		JavascriptExecutor js=(JavascriptExecutor) driver;
		js.executeScript("window.scrollBy(0,300)");
		Thread.sleep(4000);

		//Step-5 Click on About link of fb
	driver.findElement(By.linkText("About")).click();

	Thread.sleep(4000);
		driver.close();
		 }
}
```



Dropdown Handling & Select option from Dropdown using Java Script Executor

There are two approaches for Dropdown Handling.

Select Class in Selenium

The **Select Class in Selenium** is a method used to implement the HTML SELECT tag. The html select tag provides helper methods to select and deselect the elements. The Select class is an ordinary class so New keyword is used to create its object and it specifies the web element location.

Select Option from Drop-Down Box

Following is a step by step process on how to select value from dropdown in Selenium:

Before handling dropdown in Selenium and controlling drop-down boxes, we must do following two things:

1. Import the package **org.openqa.selenium.support.ui.Select**
2. Instantiate the drop-down box as an object, Select in Selenium WebDriver

a) Locate the web element
WebElement dropdown =driver.findElement(By.id("dropdown-class-example"));

b) Create the object of select class -(Parameter-webelement)
Select sel=new Select(dropdown);

Here sel is reference variable for Dropdown Webelement.

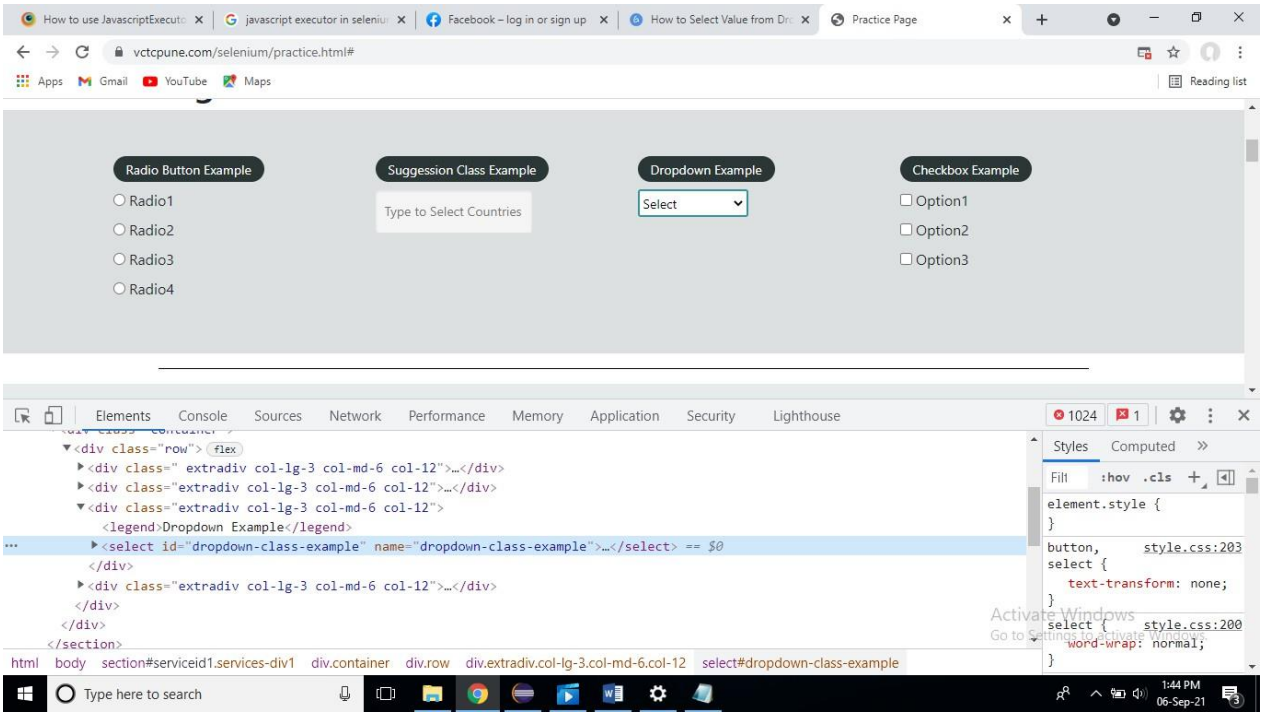
c) Select the required value(Select option from Drop down)

There are 3 approaches

i) sel.selectByIndex(2);

ii)sel.selectByVisibleText("Option2");

iii)sel.selectByValue



a) Select By Index:

selectByIndex:

This method selects the *dropdown* option by its *index number.* We provide an integer value as the index number as an argument. It possesses the following syntax:

```
selectByIndex(int arg0) : void
```

i.e., it accepts the index of the *dropdown* value, which needs to be selected. The index starts at 0.

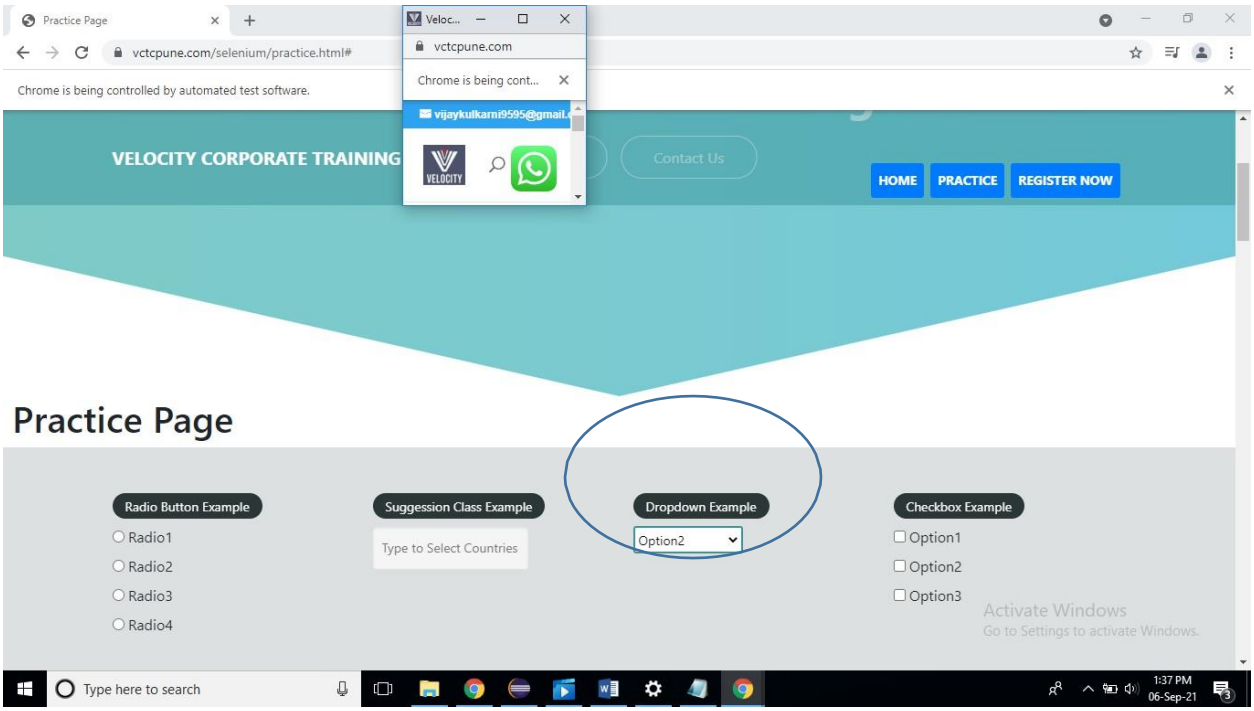
Program1: Selenium Script to drop down a following web page & Select option 2 from that drop down list by using Select class(By using Index method)

https://vctcpune.com/selenium/practice.html

Index in Drop Down starts from 0.

```
package SeleniumWebElement;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
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 WebElementDropDown {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		//Step-1
		WebDriver driver =new ChromeDriver();

		//Step-2
		driver.manage().window().maximize();

		//Step-3
		driver.get("https://vctcpune.com/selenium/practice.html#");
		Thread.sleep(4000);

	//Step-4 Scroll down window
		JavascriptExecutor js=(JavascriptExecutor) driver;
		js.executeScript("window.scrollBy(0,300)");
		Thread.sleep(4000);

		//Handling Dropdown
		//5) By using the select class

		//a) Locate the web element
		WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));

		 //b) Create the object of select class -(Parameter-
webelement)
		 Select sel=new Select(dropdown);
```

```
            //c) Select the require value
            sel.selectByIndex(2);

            Thread.sleep(4000);
            driver.close();



        }
}
```

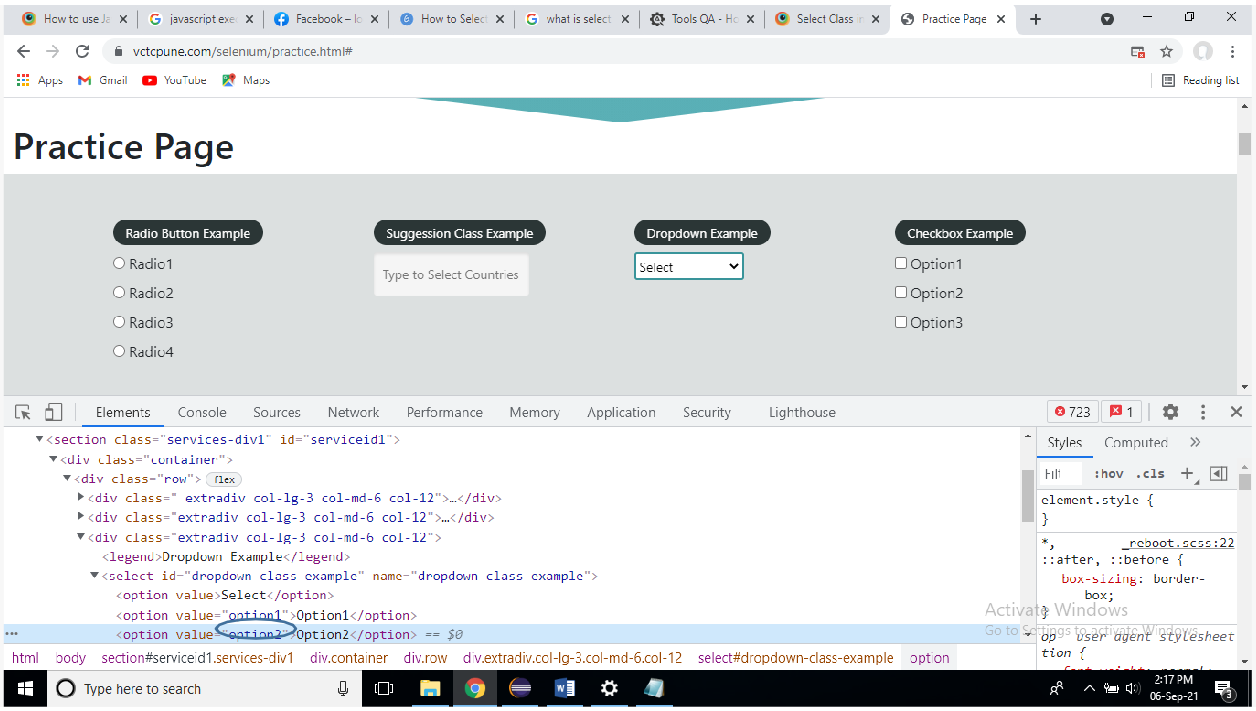


b) Select By Visible Text():

selectByVisibleText

This method enables one to select one option from the *dropdown* or multi-select *dropdown* based on the *dropdown text.* You need to pass the *String value* of the *<select>* element as an argument. It possesses the following syntax:

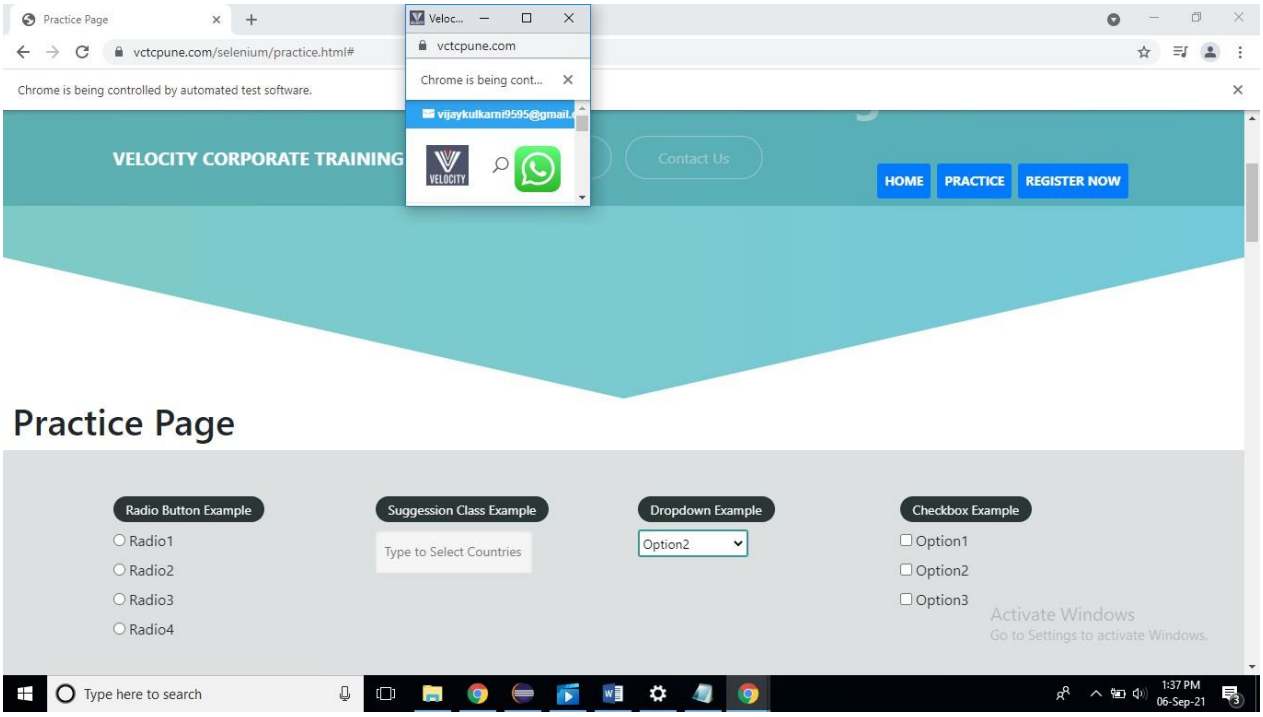
```
selectByVisibleText(String arg0): void
```



Program2: Selenium Script to drop down a following web page & Select option 2 from that drop down list by using Select class(By using ByVisibleText method)

```
package SeleniumWebElement;

/*Date:06/09/2021
 Selenium Script to drop down a following web page & Select option 2
from that drop down list
 by using Select class( By using ByVisibleText method)*/

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
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 WebElementDropDown2 {

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

```
System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

            //Step-1
            WebDriver driver =new ChromeDriver();

            //Step-2
            driver.manage().window().maximize();

            //Step-3
            driver.get("https://vctcpune.com/selenium/practice.html#");
            Thread.sleep(4000);

         //Step-4 Scroll down window
            JavascriptExecutor js=(JavascriptExecutor) driver;
            js.executeScript("window.scrollBy(0,300)");
            Thread.sleep(4000);

            //Handling Dropdown
            //5) By using the select class

            //a) Locate the web element
            WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));

             //b) Create the object of select class -(Parameter-
webelement)
             Select sel=new Select(dropdown);

             //c) Select the require value
             sel.selectByVisibleText("Option2");

             Thread.sleep(4000);
             driver.close();



            }

}
```



c) Select By Value:

selectByValue

This method selects the *dropdown* option by its *value.* We provide a string value as the value as an argument. It possesses the following syntax:

```
selectByValue(String arg0) : void
```

Program2: Selenium Script to drop down a following web page & Select option 2 from that drop down list by using Select class(By using ByValue method)

```
package SeleniumWebElement;

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

/*Date:06/09/2021
Selenium Script to drop down a following web page & Select option 2
from that drop down list
```

```
by using Select class( By using Value method) */

public class WebElementDropDown4 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		//Step-1
		WebDriver driver =new ChromeDriver();

		//Step-2
		driver.manage().window().maximize();

		//Step-3
		driver.get("https://vctcpune.com/selenium/practice.html#");
		Thread.sleep(4000);

	//Step-4 Scroll down window
		JavascriptExecutor js=(JavascriptExecutor) driver;
		js.executeScript("window.scrollBy(0,300)");
		Thread.sleep(4000);

		//Handling Dropdown
		//5) By using the select class

		//a) Locate the web element
		WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));

		 //b) Create the object of select class -(Parameter-
webelement)
		 Select sel=new Select(dropdown);

		 //c) Select the require value
		 sel.selectByValue("option1");

		 Thread.sleep(4000);
		 driver.close();
		}

}
```

9)Selenium 9 Lecture

Handling Radio buttons:

6 Ways to Write Relative (Dynamic) XPath in Selenium:

1. Basic XPath

XPath expression selects nodes or lists of nodes on the basis of attributes like ID, name, classname, etc. from the XML document as illustrated below.

```
XPath=//*[@name='username']
```

2. Contains()

Contains() is a method used in an XPath expression. It is used when the value of any attribute changes dynamically — for example, sign up information.

The contain feature has an ability to find the element with partial text as shown in the below example. In this example, we tried to identify the element by just using partial text value of the attribute. In the below XPath expression, partial value 'Name' is used.

Complete value of 'id' is 'FirstName' but using only partial value 'Name.'

```
XPath=//*[contains(@id,'Name')
```

In the above expression, we have taken the 'id' as an attribute and 'Name' as a partial value as shown in the below screenshot. This will find two elements (FirstName and LastName) as their 'id' attribute ending with 'Name.'

3. Using OR & AND

In **OR expression**, two conditions are used, whether the first condition OR second condition should be true. It is also applicable if any one condition is true, or maybe both. This means that any one condition should be true to find the element.

In the below XPath expression, it identifies the elements whose single or both conditions are true.

```
XPath=//*[@id='FirstName' or @name='LastName']
```

Highlight both elements as 'First Name' element having attribute 'id' and 'Last Name' element having attribute 'name'.

In **AND expression**, two conditions are used. Both conditions should be true to find the element. It fails to find the element if any one condition is false.

```
XPath=//*[@id='FirstName' and @name='FirstName']
```

In the below expression, we highlight the 'First Name' element as it has both attributes 'id' and 'name.'

4. Starts-With Function

Starts-with function finds the element whose attribute value changes on refresh or any operation on the webpage. In this expression, match the starting text of the attribute used to find the element whose attribute changes dynamically. You can also find the element whose attribute value is static (does not change).

For example, suppose the ID of a particular element changes dynamically like:

Id=" message12"

Id=" message345"

Id=" message8769"

And so on. But the initial text is same. In this case, we use Start-with expression “message.”

In the below expression, there are two elements with a class starting with "form." So, XPath finds those elements whose 'class' is starting with 'form.'

```
XPath=//input[starts-with(@class,'form')]
```

5. Text()

In this expression, with text function, we find the element with the exact text match as shown below. In this case, we find the element with text "Username or email."

```
XPath=//*[text()='Username or email']
```

6. Using Index:

This approach comes in use when you wish to specify a given tag name in terms of the index value you wish to locate.

```
XPath=(//*[@attribute='value'])[index]
```

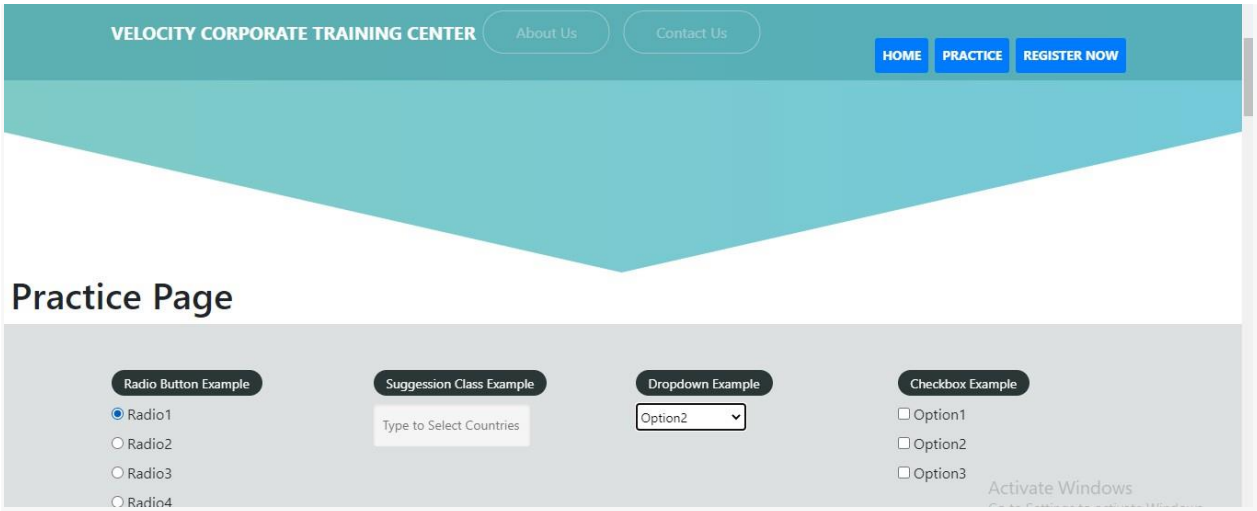
In the below expression, a DOM has multiple input tags for each different field value, and you wish to input text into the second field i.e. Last Name.

XPath=(//input[@type=”text”)[2]

Program1: Write a Script to handle **Radio button & drop down** of requested page by using **Basic Xpath.** By.*xpath*("//input[@value=\"Radio1\"]")

```
package SeleniumWebElement;

/*Date:07/09/2021
 Write a Script to Handle Radio button & drop down of requested page
using Basic Xpath.*/

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
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 WebElementRadioButton {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

        WebDriver driver =new ChromeDriver();

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

        driver.get("https://vctcpune.com/selenium/practice.html#");
           Thread.sleep(4000);

           JavascriptExecutor js=(JavascriptExecutor) driver;
           js.executeScript("window.scrollBy(0,300)");
           Thread.sleep(4000);


      driver.findElement(By.xpath("//input[@value=\"Radio1\"]")).click(
);
```

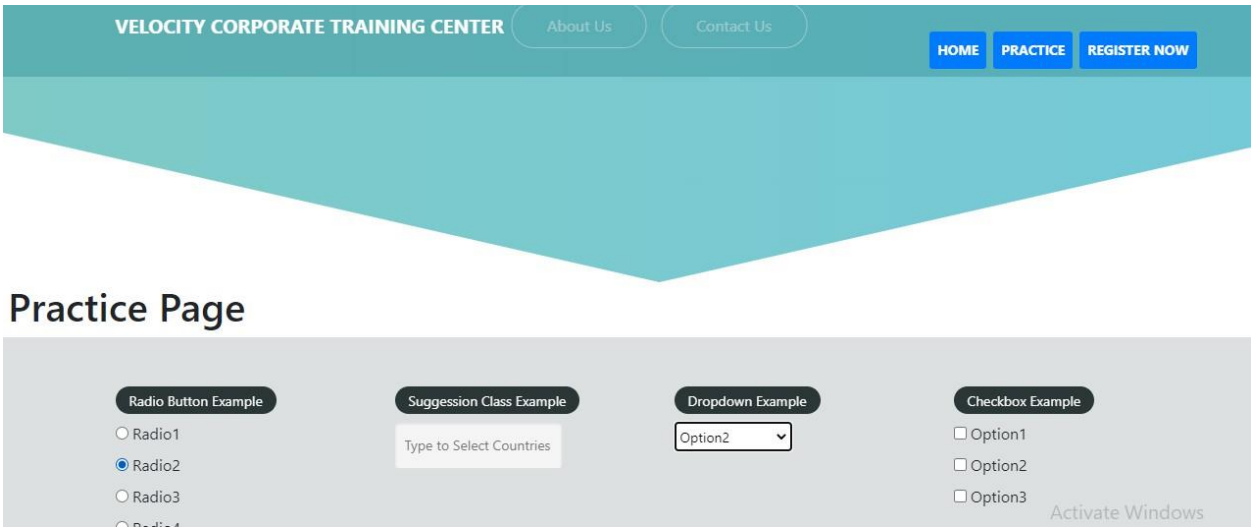
```
            WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));
            Select sel=new Select(dropdown);
            sel.selectByIndex(2);

      }

}
```



Program2: Write a Script to handle **Radio button & drop down** of requested page by using **AND Xpath.**

```
By.xpath("//input[@name='radio' and @value=\"Radio2\"]")
```

```java
package SeleniumWebElement;

/*Date:07/09/2021
 Write a Script to Handle Radio button & drop down of requested page
using AND Xpath.*/

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
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 WebElementRadioButton2 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		WebDriver driver =new ChromeDriver();

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

		driver.get("https://vctcpune.com/selenium/practice.html#");
		   Thread.sleep(4000);

		   JavascriptExecutor js=(JavascriptExecutor) driver;
		   js.executeScript("window.scrollBy(0,300)");
		   Thread.sleep(4000);

		   driver.findElement(By.xpath("//input[@name='radio' and
@value=\"Radio2\"]")).click();

		   WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));
		   Select sel=new Select(dropdown);
		   sel.selectByIndex(2);

	}

}
```



Program3: Write a Script to handle **Radio button & drop down** of requested page by using **Axes Xpath.**

//div[@class= 'extradiv col-lg-3 col-md-6 col-12'] : div is Parent class.

/child::input[2] : input is a tag name. 2 is index number.

Firstly we locate class to their parent then traveling from parent to child then child forward it to the tag name which contains specific value i.e. Child is of input category.

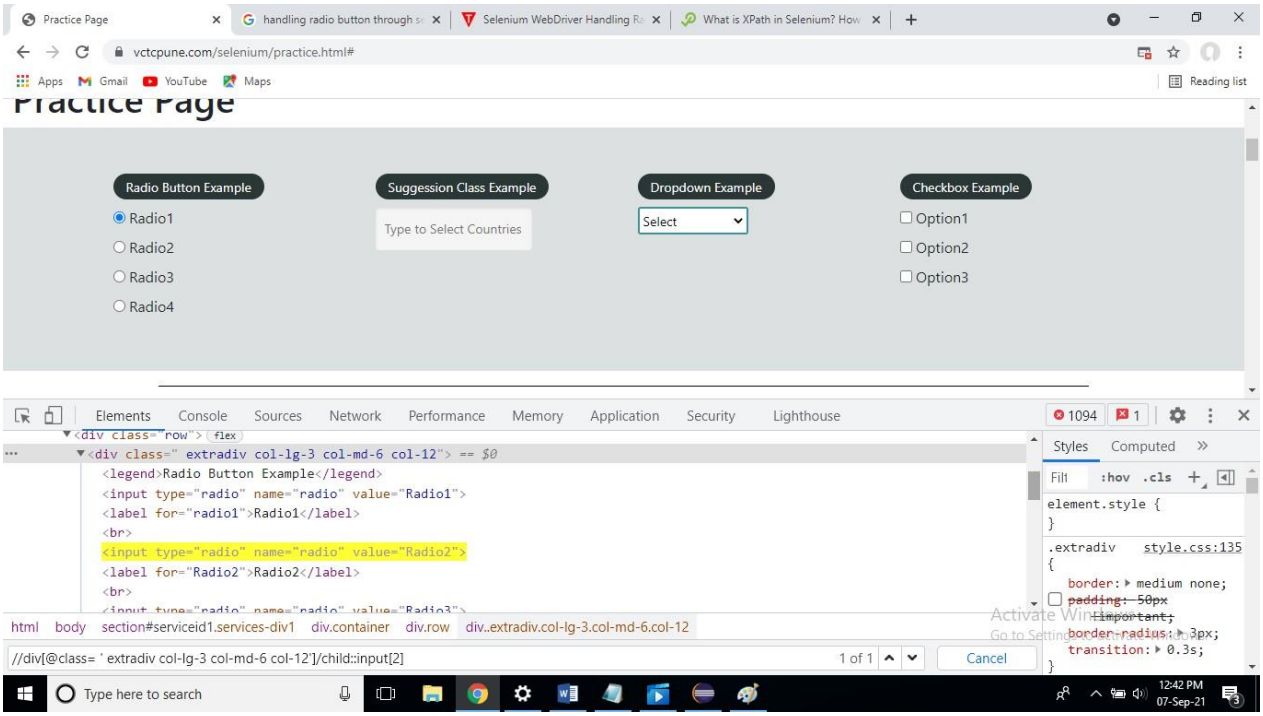
Advantages of Contains XPath in this Scenario is only by changing input value you are able to click on different radio buttons or different options.

i)//div[@class= ' extradiv col-lg-3 col-md-6 col-12']/child::input[2]

We can write above Xpath by using attribute as follows:

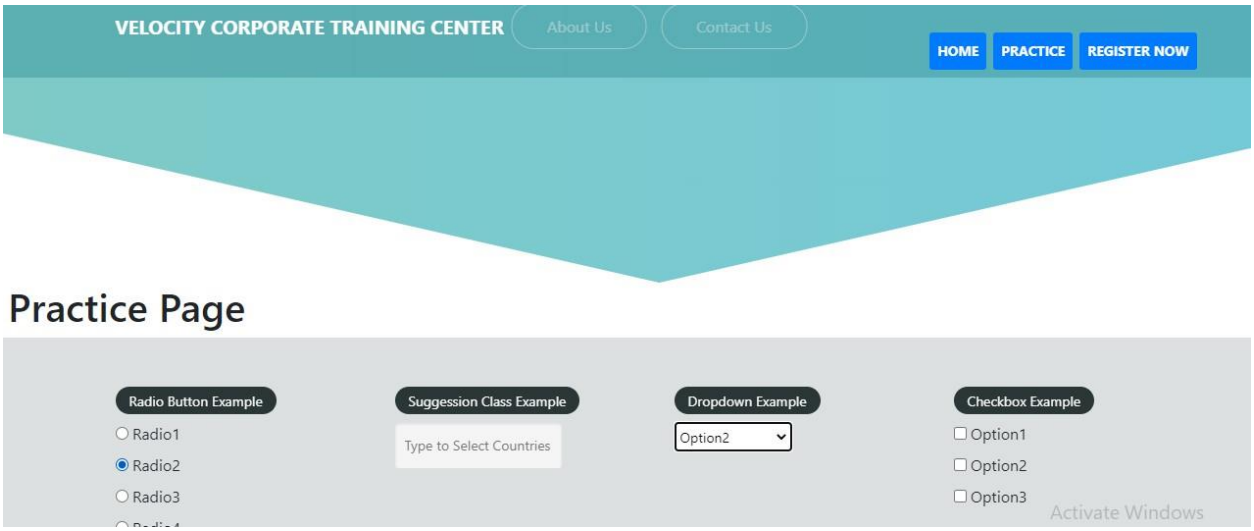
ii)//div[@class= ' extradiv col-lg-3 col-md-6 col-12']/child::input[@value='Radio2']

axesMethodName::tagname




```
package SeleniumWebElement;

/*Date:07/09/2021
 Write a Script to Handle Radio button & drop down of requested page
using Axes Xpath.*/

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
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 WebElementRadioButton2 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		WebDriver driver =new ChromeDriver();

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

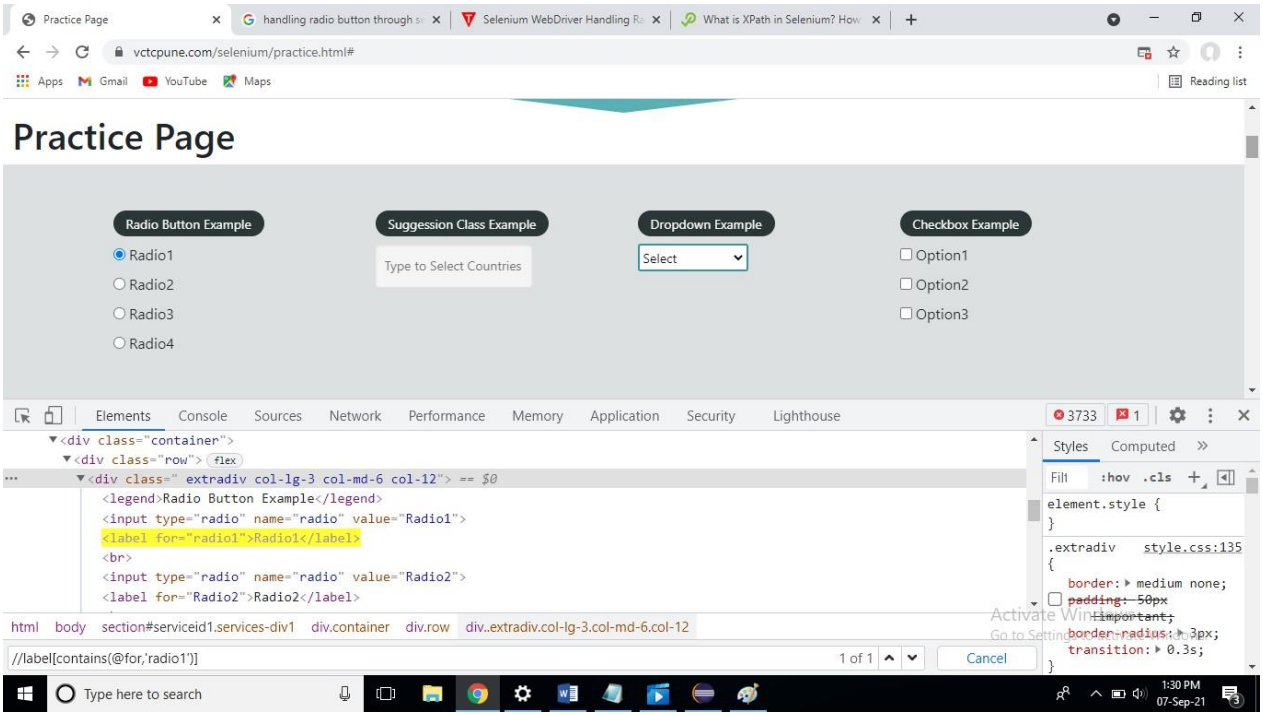
```
        driver.get("https://vctcpune.com/selenium/practice.html#");
            Thread.sleep(4000);

            JavascriptExecutor js=(JavascriptExecutor) driver;
            js.executeScript("window.scrollBy(0,300)");
            Thread.sleep(4000);

driver.findElement(By.xpath("//div[@class='extradiv col-lg-3 col-md-6 col-12']/child::input[2]")).click();
//driver.findElement(By.xpath("//div[@class='extradiv col-lg-3 col-md-6 col-12']/child::input[@value=\"Radio2\"]")).click();

            WebElement  dropdown =driver.findElement(By.id("dropdown-class-example"));
            Select sel=new Select(dropdown);
            sel.selectByIndex(2);

      }

}
```



Program4: Write a Script to handle **Radio button & drop down** of requested page by using **Contains Xpath.**

We are checking label of Radio1 button.

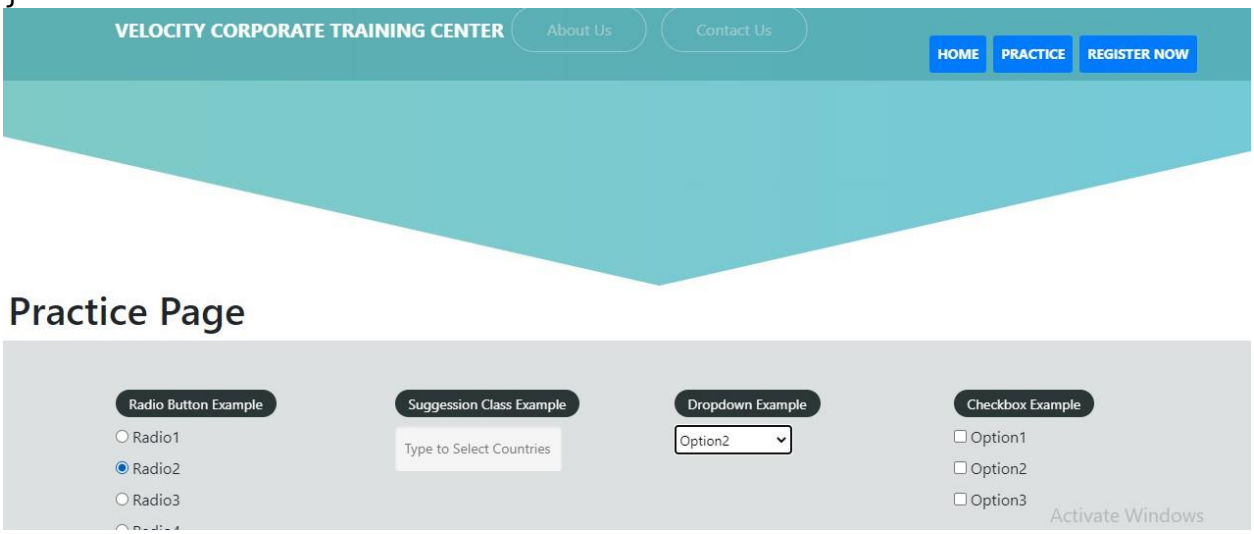
//label[contains(@for,'radio1')]

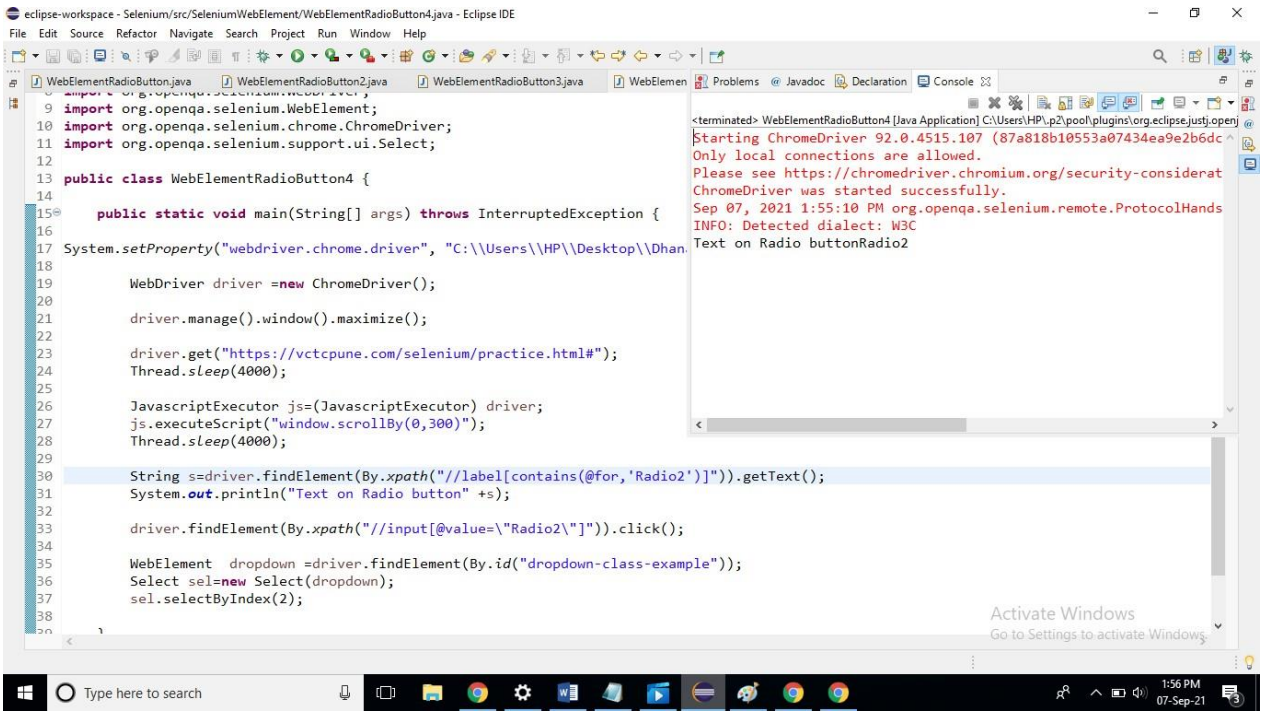


For taking text of Web Element we use getText() method like sendkeys()method.

```
package SeleniumWebElement;

/*Date:07/09/2021
Write a Script to Handle Radio button & drop down of requested page
using Contains Xpath.*/

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
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 WebElementRadioButton4 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");
```

```
        WebDriver driver =new ChromeDriver();

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

        driver.get("https://vctcpune.com/selenium/practice.html#");
           Thread.sleep(4000);

           JavascriptExecutor js=(JavascriptExecutor) driver;
           js.executeScript("window.scrollBy(0,300)");
           Thread.sleep(4000);

        String
s=driver.findElement(By.xpath("//label[contains(@for,'Radio2')]")).get
Text();
        System.out.println("Text on Radio button" +s);

        driver.findElement(By.xpath("//input[@name='radio' and
@value=\"Radio2\"]")).click();

           WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));
           Select sel=new Select(dropdown);
           sel.selectByIndex(2);

     }

}
```





Program5:Write a Script to Select multiple options at a time from a single drop down list.

IsMultiple() method:

isMultiple( ) command is used to verify whether the specified select element support selecting multiple options at the same time.

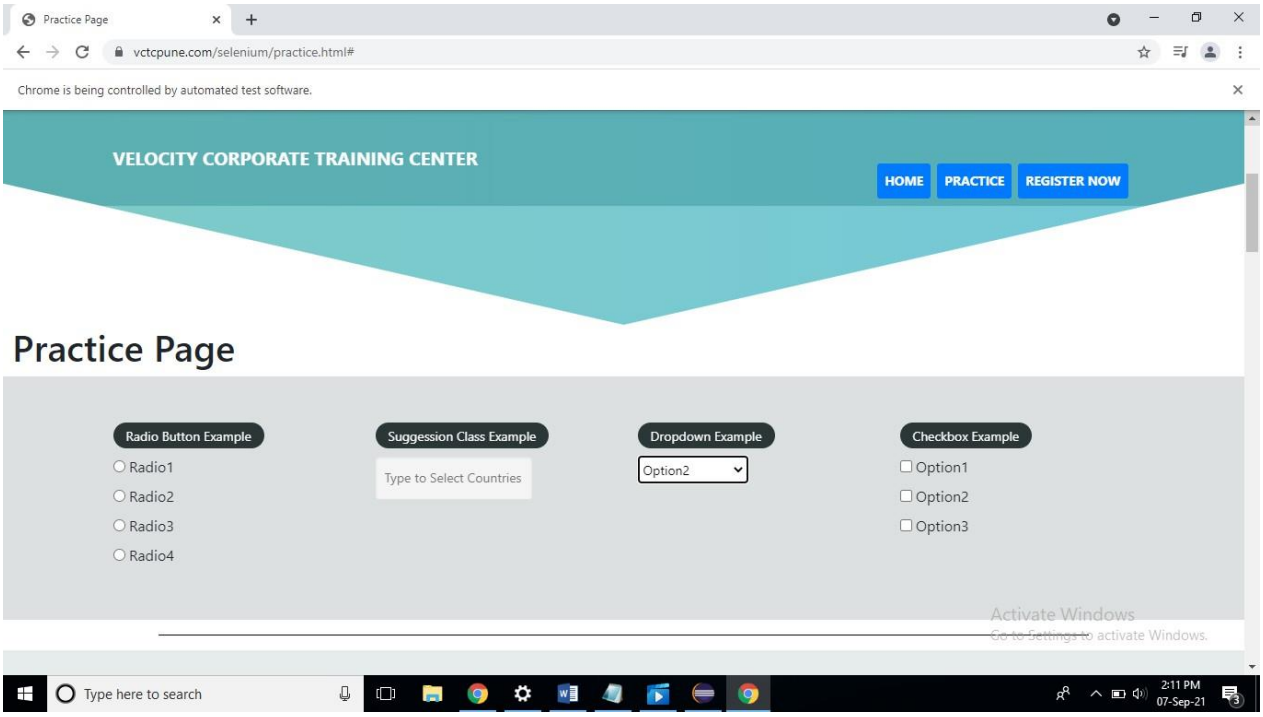
isMultiple( ) returns true when the specified select element support selecting multiple options else it will return false.

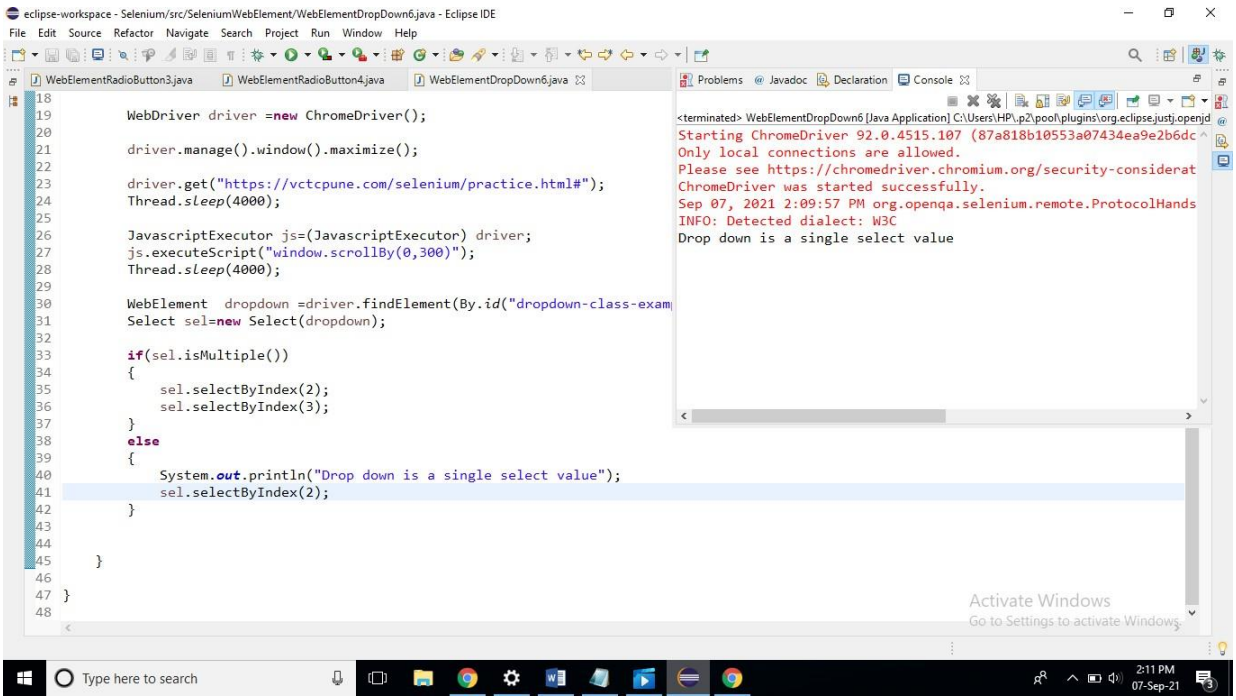
We have two kinds of select elements:

1) **Drop Down field** - Wont support selecting Multiple options (Can only select one option at a time)
2) Multi-Selection Box field - Support selecting Multiple options (Can select more than one option at a time)

```
package SeleniumWebElement;

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

```java
/*Date:07/09/2021
  Write a Script to Select multiple options at a time from a single
drop down list. */

public class WebElementDropDown6 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		WebDriver driver =new ChromeDriver();

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

		driver.get("https://vctcpune.com/selenium/practice.html#");
		   Thread.sleep(4000);

		   JavascriptExecutor js=(JavascriptExecutor) driver;
		   js.executeScript("window.scrollBy(0,300)");
		   Thread.sleep(4000);

		   WebElement  dropdown =driver.findElement(By.id("dropdown-
class-example"));
		   Select sel=new Select(dropdown);

		   if(sel.isMultiple())
		   {
			sel.selectByIndex(2);
			sel.selectByIndex(3);
		   }
		   else
		   {
			System.out.println("Drop down is a single select
value");
			sel.selectByIndex(2);
		   }


	}

}
```


Practice Page
vctcpune.com/selenium/practice.html#
Chrome is being controlled by automated test software.
VELOCITY CORPORATE TRAINING CENTER
HOME
PRACTICE
REGISTER NOW
Practice Page
Radio Button Example
Radio1
Radio2
Radio3
Radio4
Suggession Class Example
Type to Select Countries
Dropdown Example
Option2
Checkbox Example
Option1
Option2
Option3
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:11 PM
07-Sep-21


eclipse-workspace - Selenium/src/SeleniumWebElement/WebElementDropDown6.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
WebElementRadioButton3.java
WebElementRadioButton4.java
WebElementDropDown6.java
WebDriver driver =new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://vctcpune.com/selenium/practice.html#");
Thread.sleep(4000);
JavascriptExecutor js=(JavascriptExecutor) driver;
js.executeScript("window.scrollBy(0,300)");
Thread.sleep(4000);
WebElement dropdown =driver.findElement(By.id("dropdown-class-exam
Select sel=new Select(dropdown);
if(sel.isMultiple())
{
sel.selectByIndex(2);
sel.selectByIndex(3);
}
else
{
System.out.println("Drop down is a single select value");
sel.selectByIndex(2);
}
}
}
Problems
Javadoc
Declaration
Console
<terminated> WebElementDropDown6 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjd
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dc
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerat
ChromeDriver was started successfully.
Sep 07, 2021 2:09:57 PM org.openqa.selenium.remote.ProtocolHands
INFO: Detected dialect: W3C
Drop down is a single select value
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:11 PM
07-Sep-21

10) Selenium 10 Lecture

Date Handler or Calendar Handling:

Types Of Calendar Controls

A calendar control (date picker control) allows the user to select a date easily. Usually, it appears as an input field in an HTML form. Since the date selected by the user might be needed later, it is important to maintain the format. This is also why HTML forms are more widely used than entering date in a text-box.

There are two popular types of Calendar controls you'd need to automate calendar using Selenium WebDriver:

- **jQuery Calendar –** The jQuery calendar is a part of the open-source project at JS Foundation (previously called jQuery Foundation). There are a number of other elements like user interface interactions, widgets, effects, etc. that are also built on top of jQuery JavaScript Library.
- **Kendo Calendar –** Kendo Calendar is developed by Telerik.com. It is not an open-source project. Using Kendo UI, developers can build JavaScript apps faster. More details about Kendo UI are available here.

Kendo and jQuery Calendars work on all major web browsers. But there are a few exceptions- Kendo on IE works on IE7 (and above) whereas jQuery works on IE8 (and above). Both these controls are responsive and have mobile browser compatibility.

Handling 'JQuery Calendar'

Step 1: Set the path

Step2: Create reference of Web driver in the reference of object of chrome driver

Step 3: Use window method to maximize browser window

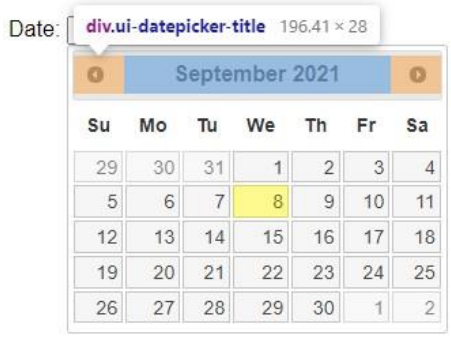
Step 4: Get a webpage by using get() method.

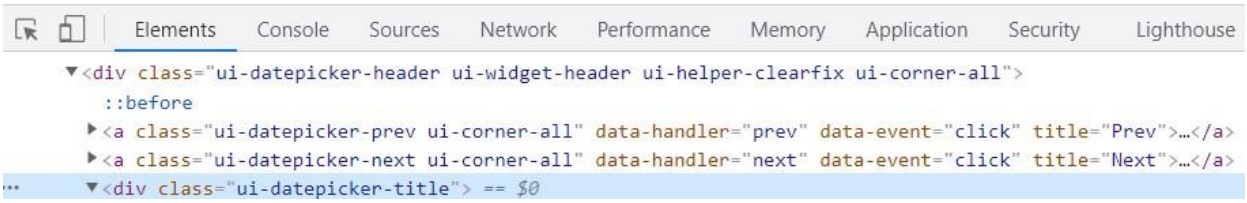
Step 5: Locate the ID & Perform Click operation

Step 6: i)Select the specific month i.e. December 2017 through the script.

ii) After opening the calendar in web page, we will get the text of Current text which shows.

And by getting current text, we will click on back arrow(we will go in previous months & years) until we get expected text(i.e. December 2017)





Step 7: In while loop

i) For getting title of text through web element we have to use

```
String s=driver.findElement(By.xpath("//div[@class='ui-datepicker-title']")).getText();
```

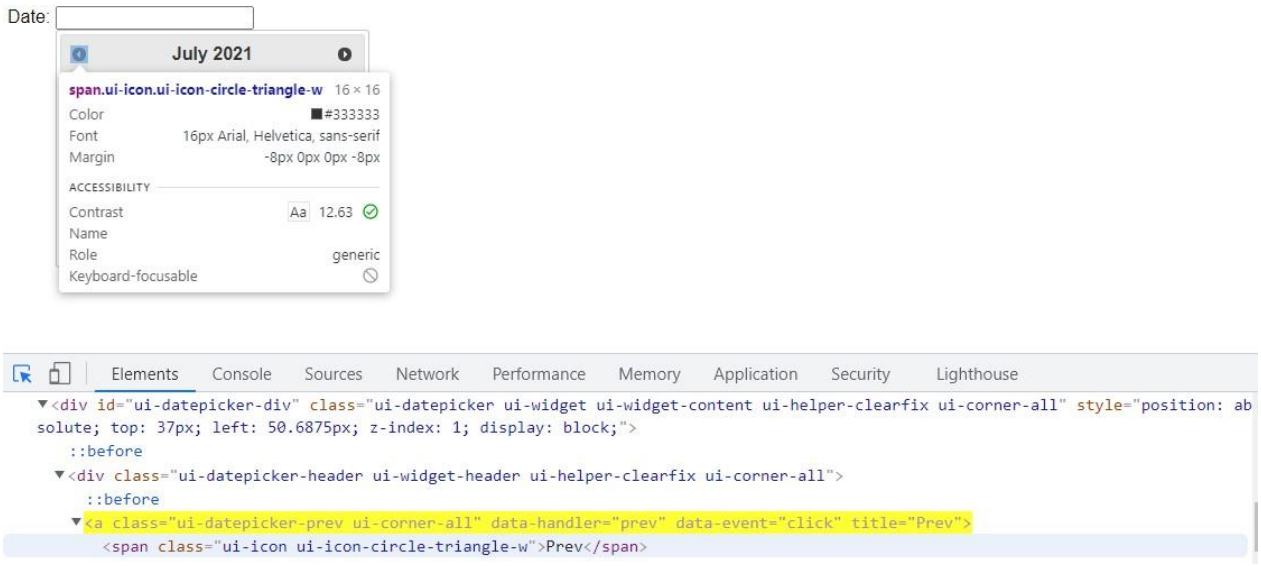
ii) Print that title

iii) Compare the title with December 2017 using if condition

```
if(!s.equalsIgnoreCase("December 2017"))
```

iv) If title is not equal to December 2017 then click on Back button. Locate back button using Xpath

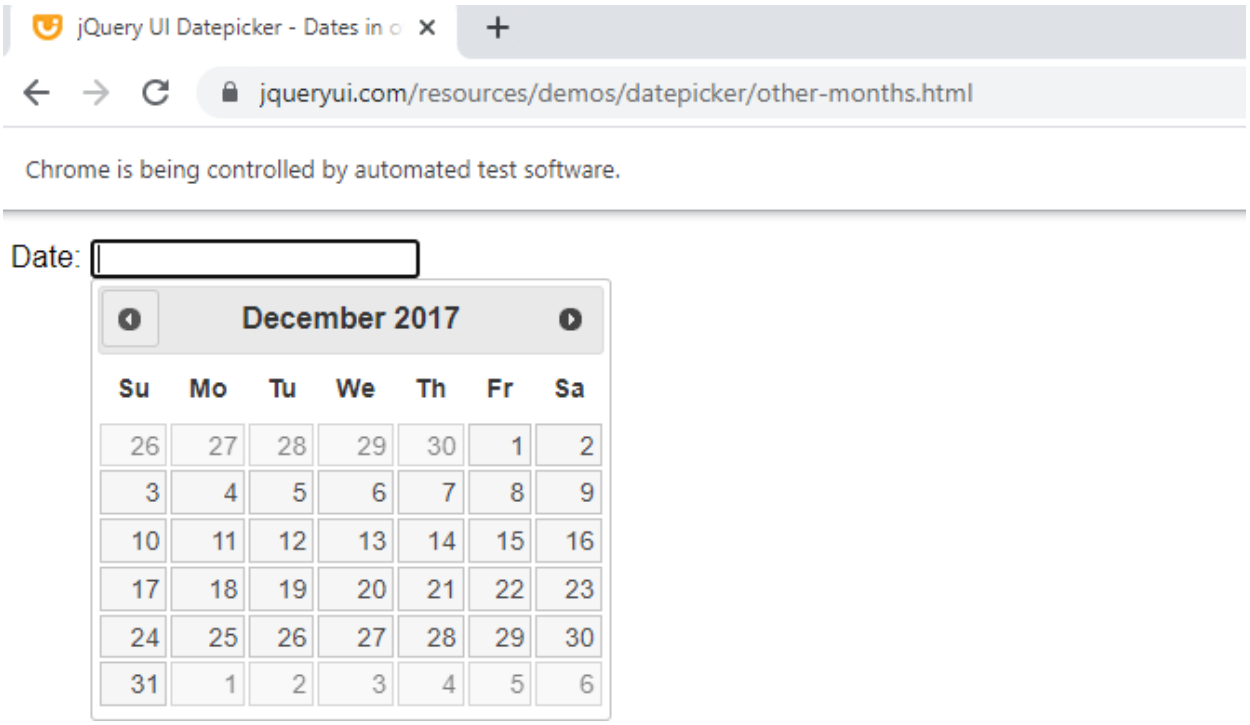
```
driver.findElement(By.xpath("//a[@title='Prev']")).click();
```

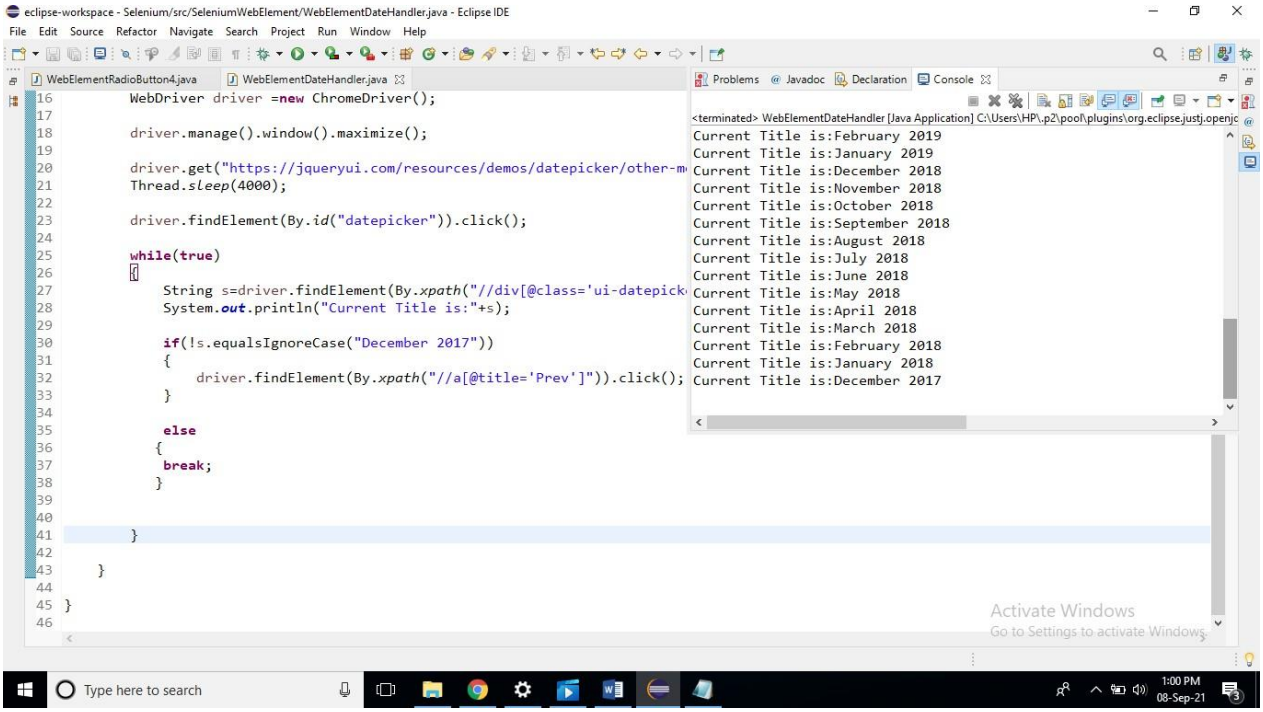


Program1:Write a Script to Handle a Date i.e. Select December 2017 from calendar

```
package SeleniumWebElement;

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

/*Date:08/09/2021
  Write a script to Handle Date on a web page i.e. select December
2018 from calendar */

public class WebElementDateHandler {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

        WebDriver driver =new ChromeDriver();

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


driver.get("https://jqueryui.com/resources/demos/datepicker/other-
months.html");
           Thread.sleep(4000);
```

```
            driver.findElement(By.id("datepicker")).click();

            while(true)
            {
                  String
s=driver.findElement(By.xpath("//div[@class='ui-datepicker-
title']")).getText();
                  System.out.println("Current Title is:"+s);

                  if(!s.equalsIgnoreCase("December 2017"))
                  {

      driver.findElement(By.xpath("//a[@title='Prev']")).click();
                  }

                 else
                {
                  break;
              }


            }

      }

}
```





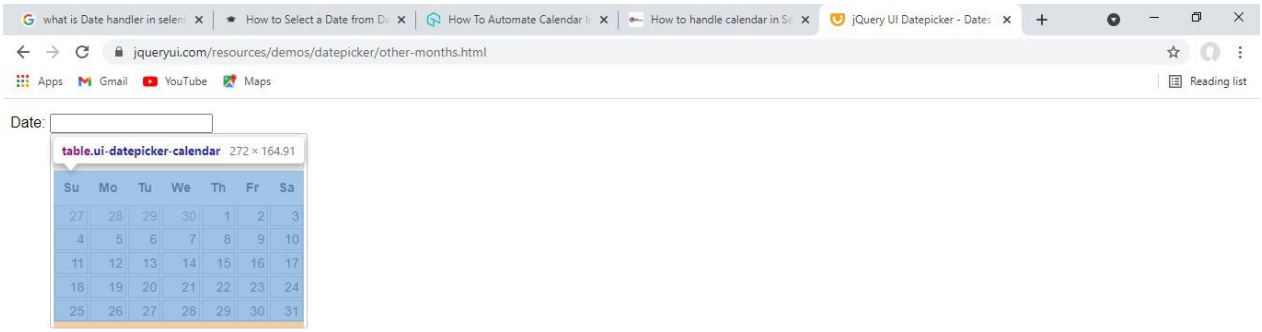
For Selecting Date from Calender:

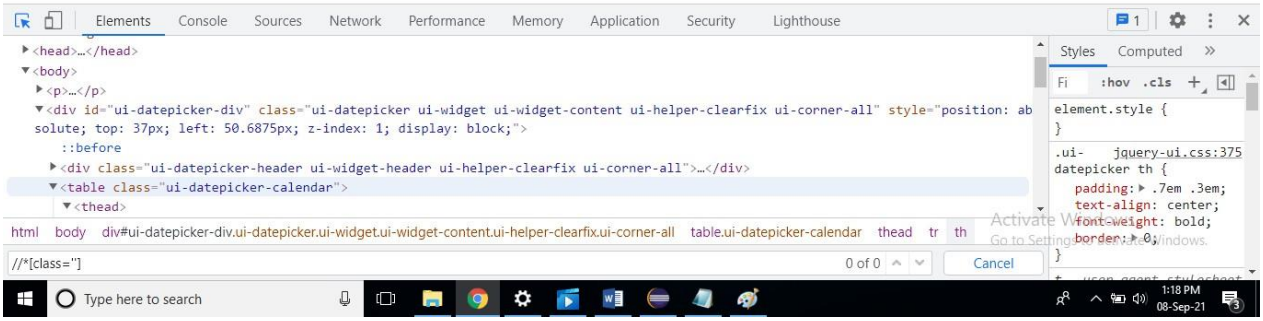
Step 1 to Step 7 are same as in Month Selection.

Step 8:i)First we have to take parent class of calendar by Xpath

When we don't want to use tag name in Xpath then we can use *.

We take class name





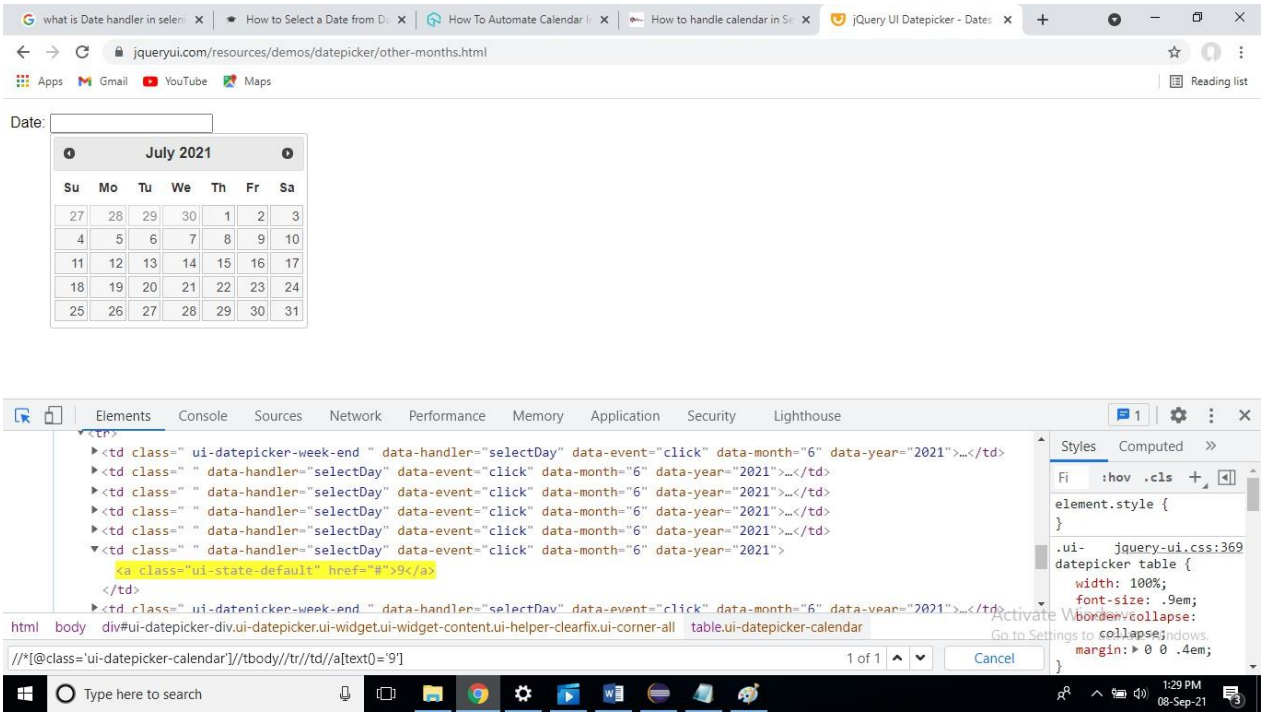
ii) Then we have to travel through <tbody> tag for Date selection. Use //tbody in Xapth

iii) Then <tbody> <tr> <td>

iv) Now we are rows of Date

v) Inside <td> there is a link for number like href

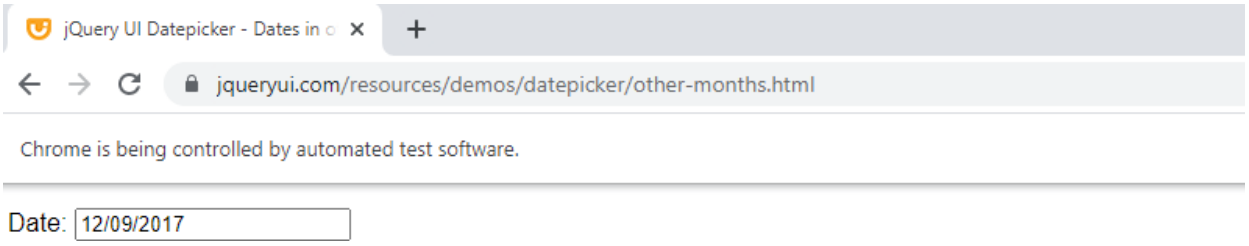
vi) Tage name as a[text()=9] text is a method & 9 is date which we have to selected.

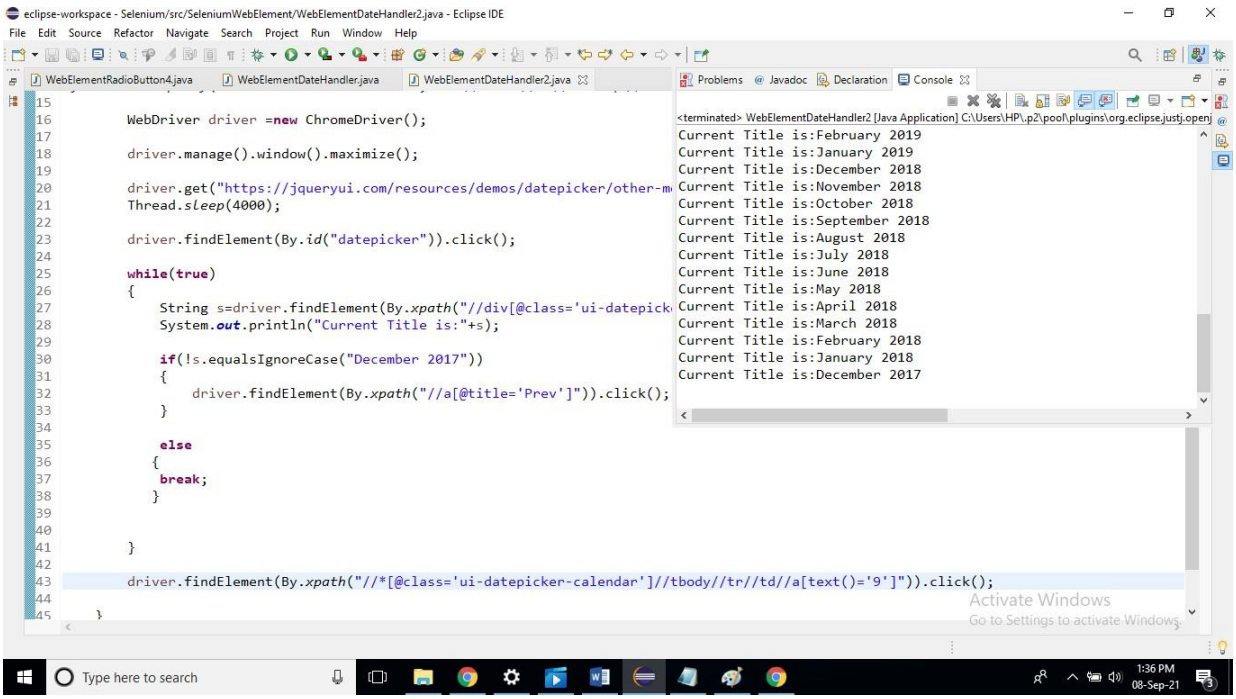


Program2:Write a Script to Handle a Date i.e. Select 9 December 2017 from calendar

```
package SeleniumWebElement;

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

/*Date:08/09/2021
  Write a script to Handle Date on a web page i.e. select 9 December
2017 from calendar */

public class WebElementDateHandler2 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		WebDriver driver =new ChromeDriver();
```

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


driver.get("https://jqueryui.com/resources/demos/datepicker/other-
months.html");
            Thread.sleep(4000);

            driver.findElement(By.id("datepicker")).click();

            while(true)
            {
                  String
s=driver.findElement(By.xpath("//div[@class='ui-datepicker-
title']")).getText();
                  System.out.println("Current Title is:"+s);

                  if(!s.equalsIgnoreCase("December 2017"))
                  {

      driver.findElement(By.xpath("//a[@title='Prev']")).click();
                  }

                 else
                {
                   break;
              }


            }

            driver.findElement(By.xpath("//*[@class='ui-datepicker-
calendar']//tbody//tr//td//a[text()='9']")).click();

      }

}
```


jQuery UI Datepicker - Dates in
jqueryui.com/resources/demos/datepicker/other-months.html
Chrome is being controlled by automated test software.
Date: 12/09/2017


eclipse-workspace - Selenium/src/SeleniumWebElement/WebElementDateHandler2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
WebElementRadioButton4.java
WebElementDateHandler.java
WebElementDateHandler2.java
WebDriver driver =new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://jqueryui.com/resources/demos/datepicker/other-m
Thread.sleep(4000);
driver.findElement(By.id("datepicker")).click();
while(true)
{
String s=driver.findElement(By.xpath("//div[@class='ui-datepick
System.out.println("Current Title is:"+s);
if(!s.equalsIgnoreCase("December 2017"))
{
driver.findElement(By.xpath("//a[@title='Prev']")).click();
}
else
{
break;
}
}
driver.findElement(By.xpath("//*[@class='ui-datepicker-calendar']//tbody//tr//td//a[text()='9']")).click();
Problems Javadoc Declaration Console
<terminated> WebElementDateHandler2 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openj
Current Title is:February 2019
Current Title is:January 2019
Current Title is:December 2018
Current Title is:November 2018
Current Title is:October 2018
Current Title is:September 2018
Current Title is:August 2018
Current Title is:July 2018
Current Title is:June 2018
Current Title is:May 2018
Current Title is:April 2018
Current Title is:March 2018
Current Title is:February 2018
Current Title is:January 2018
Current Title is:December 2017
Activate Windows
Go to Settings to activate Windows.
Type here to search
1:36 PM
08-Sep-21

Program3:Write a Script to Handle a Date i.e. Select 9 December 2017 from calendar

We have requirement that We have change date every time but Month & year should be same.
We can achieve this by using previous program but it is not easible to change our XPath every time so we just declare on int date=9.

The value stored in date variable is the date that we want. Passing that date variable in Xpath.

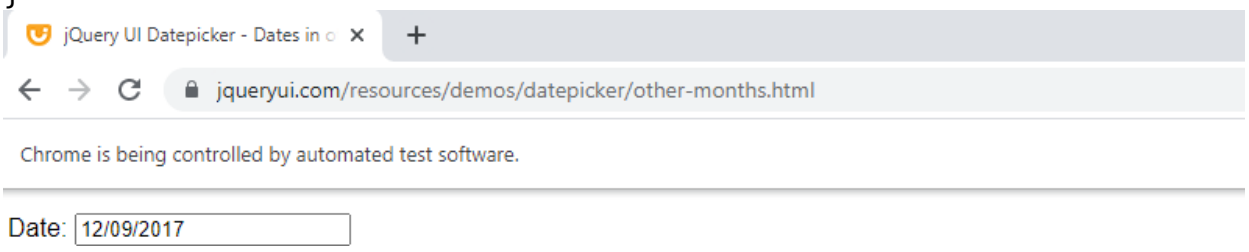
These XPath is common for every date. Just need to change date variable value as we want date.

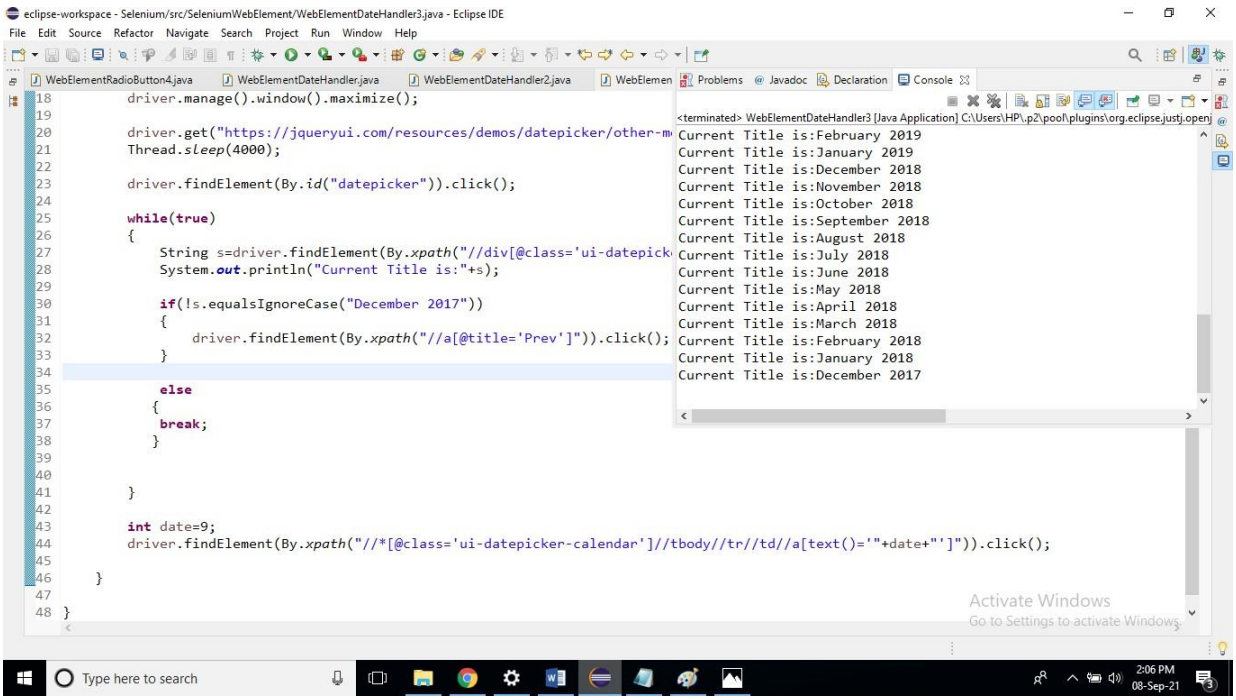
```
driver.findElement(By.xpath("//*[@class='ui-datepicker-
calendar']//tbody//tr//td//a[text()='"+date+"']")).click();
```

Here two Strings are aggregated or Appended or Concatenated with each other using + operator(i.e. date & month are going to be appended.)

```
package SeleniumWebElement;

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

/*Date:08/09/2021
  Write a script to Handle Date on a web page i.e. select 9 December
2017 from calendar */

public class WebElementDateHandler3 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

        WebDriver driver =new ChromeDriver();

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


driver.get("https://jqueryui.com/resources/demos/datepicker/other-
months.html");
           Thread.sleep(4000);

           driver.findElement(By.id("datepicker")).click();
```

```
            while(true)
            {
                  String
s=driver.findElement(By.xpath("//div[@class='ui-datepicker-
title']")).getText();
                  System.out.println("Current Title is:"+s);

                  if(!s.equalsIgnoreCase("December 2017"))
                  {

      driver.findElement(By.xpath("//a[@title='Prev']")).click();
                  }

                 else
                {
                   break;
              }


            }

            int date=9;
            driver.findElement(By.xpath("//*[@class='ui-datepicker-
calendar']//tbody//tr//td//a[text()='"+date+"']")).click();

      }

}
```

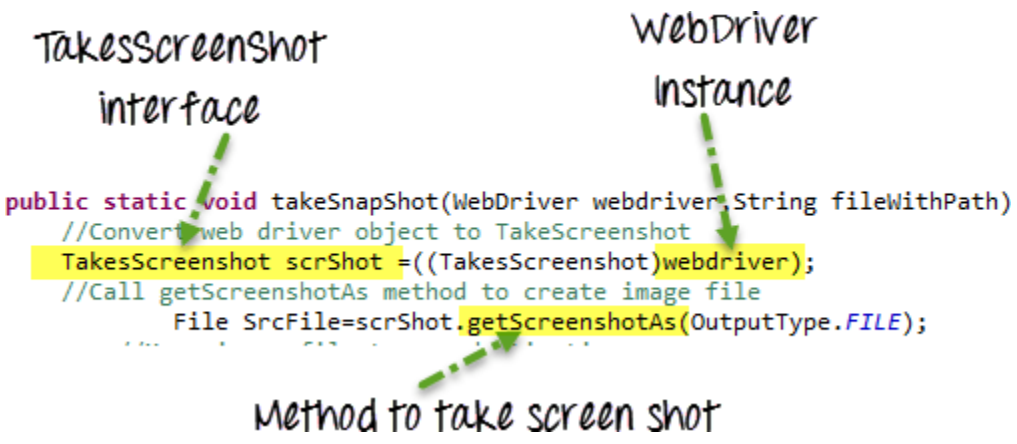



eclipse-workspace - Selenium/src/SeleniumWebElement/WebElementDateHandler3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
WebElementRadioButton4.java
WebElementDateHandler.java
WebElementDateHandler2.java
WebElemen
driver.manage().window().maximize();
driver.get("https://jqueryui.com/resources/demos/datepicker/other-m
Thread.sleep(4000);
driver.findElement(By.id("datepicker")).click();
while(true)
{
String s=driver.findElement(By.xpath("//div[@class='ui-datepick
System.out.println("Current Title is:"+s);
if(!s.equalsIgnoreCase("December 2017"))
{
driver.findElement(By.xpath("//a[@title='Prev']")).click();
}
else
{
break;
}
}
int date=9;
driver.findElement(By.xpath("//*[@class='ui-datepicker-calendar']//tbody//tr//td//a[text()='"+date+"']")).click();
}
}
Problems
Javadoc
Declaration
Console
<terminated> WebElementDateHandler3 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openj
Current Title is:February 2019
Current Title is:January 2019
Current Title is:December 2018
Current Title is:November 2018
Current Title is:October 2018
Current Title is:September 2018
Current Title is:August 2018
Current Title is:July 2018
Current Title is:June 2018
Current Title is:May 2018
Current Title is:April 2018
Current Title is:March 2018
Current Title is:February 2018
Current Title is:January 2018
Current Title is:December 2017
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:06 PM
08-Sep-21

11) Selenium 11 Lecture

Take Screenshot through Selenium

A **Screenshot in Selenium Webdriver** is used for bug analysis. Selenium webdriver can automatically take screenshots during the execution. But if users need to capture a screenshot on their own, they need to use the TakeScreenshot method which notifies the WebDrive to take the screenshot and store it in Selenium.



This is where Selenium screenshots come in.

Here are some typical cases in which Selenium screenshots would be required:

- When application issues occur
- When an assertion failure occurs
- When there is some difficulty in finding web elements on a page
- Where there is a Timeout in finding web elements on a web page

- While automating a web application, we can use an interface TakesScreenshot which signals the WebDriver to take a screenshot during test execution. The TakesScreenshot interface is extended by the WebElement interface (an interface extends another interface) and the browser driver classes such as FirefoxDriver, ChromeDriver, OperaDriver, etc implement the same interface (a class implements an interface).
- RemoteWebDriver() class implements interfaces like WebDriver(),TakesScreenshot() etc and to access these methods we have to downcast the driver object.

RemoteWebDriver() class implements interfaces like WebDriver(),TakesScreenshot() etc and to access these methods we have to downcast the driver object. **i.e. we have upcasted ts reference variable of TakeScreeenshot interface to driver reerence variable Web driver Interface.**

```
1 WebDriver driver = new ChromeDriver();

2 TakesScreenshot ts = (TakesScreenshot)driver;
```

The method getScreenshotAs() in the interface TakesScreenshot helps in capturing the screenshots and storing it in a specified path. We have another interface OutputType, which is used to specify the output format of the screenshot such as FILE, BYTES, etc.

```
File file = ts.getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(file,new File ("./ScreenShot_Folder/Test1_Login.png"));
```

In the above destination path "./" indicates the current directory and we can specify the subfolder to hold the screenshots taken and the name in which the screenshots have to be saved. As per the above line, the screenshot taken would be stored in **ScreenShot_Folder subfolder** inside the current working directory in the name of **Test1_Login.png** file. If the given subfolder doesn't exist, then on first-time execution of the script the folder gets automatically created.

How to Create a New Folder named as ScreenShot in Eclipse Project

Step1:Right click on Project Then New Then Folder

Step2:Right click on ScreenShot folder then Select Show in System Explorer option.

Step3:Then ScreenShot folder will open in System Explorer then double click & copy the full path

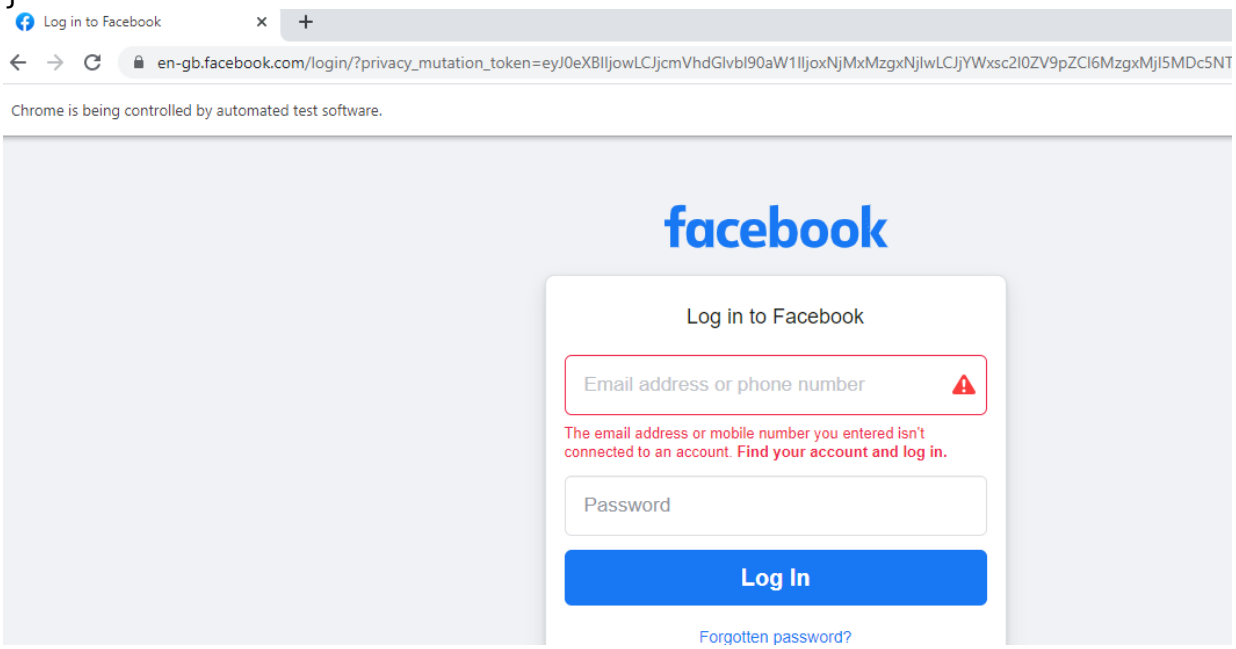
Step4: Paste the path in Destination in script.

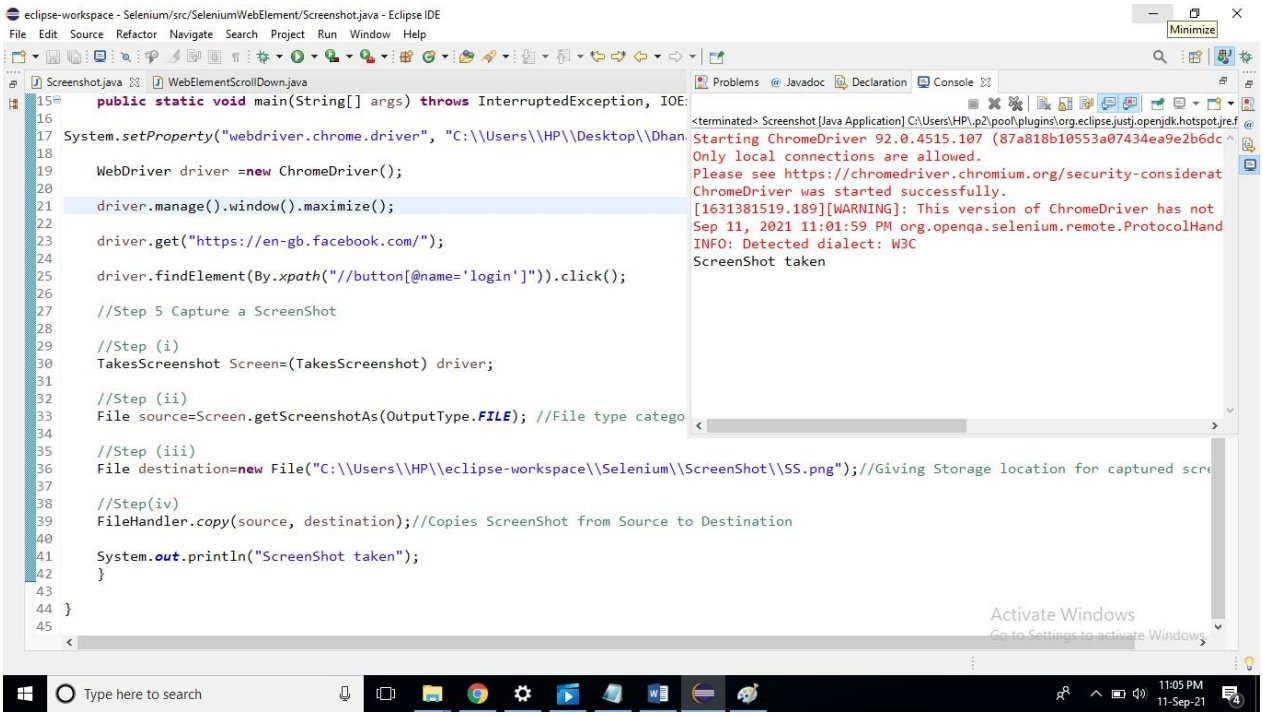
Step5:Add Screenshot Name at last in the path.

Program1:Script to Capture a Screenshot in local system.

```
package SeleniumWebElement;

/*Date:09/09/2021
Program to Capture a Screenshot in local system path */

import java.io.File;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.io.FileHandler;

public class Screenshot {

      public static void main(String[] args) throws
InterruptedException, IOException {

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

    WebDriver driver =new ChromeDriver();

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

    driver.get("https://en-gb.facebook.com/");

      driver.findElement(By.xpath("//button[@name='login']")).click();
      Thread.sleep(4000);

    //Step 5 Capture a ScreenShot

    //Step (i)
    TakesScreenshot Screen=(TakesScreenshot) driver;

    //Step (ii)
    File source=Screen.getScreenshotAs(OutputType.FILE); //File type
category

    //Step (iii)
```

```
    File destination=new File("C:\\Users\\HP\\eclipse-
workspace\\Selenium\\ScreenShot\\SS.png");//Giving Storage location
for captured screenshot.

    //Step(iv)
    FileHandler.copy(source, destination);//Copies ScreenShot from
Source to Destination

    System.out.println("ScreenShot taken");
    Thread.sleep(4000);
    driver.close();
      }

}
```





Program2:Script to capture a Screenshot in User directory i.e. Present working directory.

Suppose If we Push our ScrrenShot script in specific repository then if another user tries to execute that script then he will get an error because we are storing captured screenshot into our local system for overcome this issue we use

We want to take user director(i.e.present working directory) property as a argument in System.getProperty().

```
File                                                    destination=new
File(System.getProperty("user.dir")+"\\ScreenShot\\"+name+".png");
```

Here usr.dir is user directort that is our project.

ScreenShot is folder name in which we will store Screen Shot.

name is a String Variable in which ScreenShot name is stored that is used when we execute script every time then don't need to change in Path. Just change the value in name variable .

png is a ScreenShot file extension. We can use jpg & jpeg extension instead of png

```java
package SeleniumWebElement;

/*Date:09/09/2021
 Program to Capture a Screenshot in local system path */

import java.io.File;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.io.FileHandler;

public class ScreenShot2 {

	public static void main(String[] args) throws
InterruptedException, IOException {

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

    WebDriver driver =new ChromeDriver();

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

    driver.get("https://en-gb.facebook.com/");

      driver.findElement(By.xpath("//button[@name='login']")).click();
      Thread.sleep(4000);

    //Step 5 Capture a ScreenShot

    //Step (i)
    TakesScreenshot Screen=(TakesScreenshot) driver;

    //Step (ii)
    File source=Screen.getScreenshotAs(OutputType.FILE); //File type
category

    //Step (iii)
    String name="SS2";
    File destination=new
File(System.getProperty("user.dir")+"\\ScreenShot\\"+name+".png");
```

```
    //Step(iv)
    FileHandler.copy(source, destination);//Copies ScreenShot from
Source to Destination

    System.out.println("ScreenShot taken");
    Thread.sleep(4000);
    driver.close();
      }

}
```

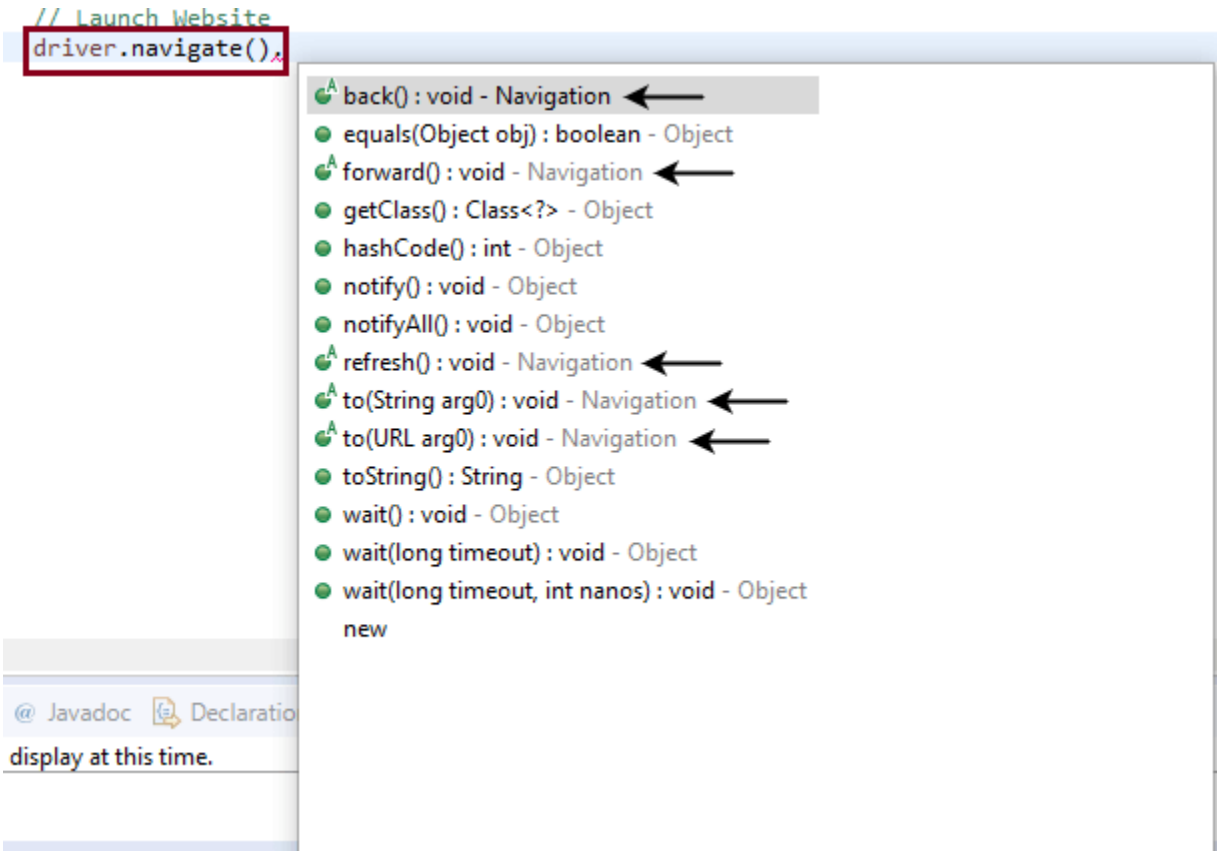
12) Selenium 12 Lecture

Navigate method():

Selenium WebDriver - Navigation Commands

WebDriver provides some basic Browser Navigation Commands that allows the browser to move backwards or forwards in the browser's history.

Just like the browser methods provided by WebDriver, we can also access the navigation methods provided by WebDriver by typing driver.navigate() in the Eclipse panel.



1. Navigate To Command

Method:

1. to(String arg0) : **void**

In WebDriver, this method loads a new web page in the existing browser window. It accepts *String* as parameter and returns *void*.

The respective command to load/navigate a new web page can be written as:

driver.navigate().to("www.javatpoint.com");

Note: The get command (driver.get(URL);) which lies in the browser commands section does the same function as the navigate command

```
package ActionElement;

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

/*Date: 11-09-2021

Program to Navigate(means go to a web page) a web page*/


public class Navigate {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

    WebDriver driver =new ChromeDriver();

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


	driver.navigate().to("https://vctcpune.com/selenium/practice.html
#");
	Thread.sleep(3000);

	}

}
```

2. Forward Command

Method:

1. to(String arg0) : **void**

   In WebDriver, this method enables the web browser to click on the **forward** button in the existing browser window. It neither accepts anything nor returns anything.

   The respective command that takes you forward by one page on the browser's history can be written as:

   driver.navigate().forward();

```
package ActionElement;

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

/*Date: 11-09-2021

Program to Navigate forward to a web page*/


public class NavigateForward1 {

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

System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

    WebDriver driver =new ChromeDriver();

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

```
    driver.get("https://www.google.co.in/");

    driver.navigate().to("https://vctcpune.com/selenium/practice.html#");
        Thread.sleep(3000);

        driver.navigate().back();
        driver.navigate().forward();

        }

}
```

3. Back Command

Method:

1. back() : **void**

   In WebDriver, this method enables the web browser to click on the **back** button in the existing browser window. It neither accepts anything nor returns anything.

   The respective command that takes you back by one page on the browser's history can be written as:

```
driver.navigate().back();
```

```
package ActionElement;

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

/*Date: 11-09-2021

Program to Navigate back to a web page*/
```

```
public class NavigateBackward {

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

System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

   WebDriver driver =new ChromeDriver();

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

   driver.get("https://www.google.co.in/");

   driver.navigate().to("https://vctcpune.com/selenium/practice.html#");
	Thread.sleep(3000);
	driver.navigate().back();


	}

}
```

4. Refresh Command

Method:

1. refresh() : **void**

   In WebDriver, this method refresh/reloads the current web page in the existing browser window. It neither accepts anything nor returns anything.

   The respective command that takes you back by one page on the browser's history can be written as:

driver.navigate().refresh();

```
package ActionElement;

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

/*Date: 11-09-2021

Program to Refresh a web page using Navigate method*/


public class NavigateRefresh {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

    WebDriver driver =new ChromeDriver();

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

    driver.get("https://www.google.co.in/");


driver.navigate().to("https://vctcpune.com/selenium/practice.html#");
      Thread.sleep(3000);

      driver.navigate().refresh();

      }

}
```

Actions Class in Selenium

What is Action Class in Selenium?

Actions class is an ability provided by Selenium for handling keyboard and mouse events. In Selenium WebDriver, handling these events includes operations such as drag and drop, clicking on multiple elements with the control key, among others. These operations are performed using the advanced user interactions API. It mainly consists of *Actions* that are needed while performing these operations.

Action class is defined and invoked using the following syntax:

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



action.moveToElement(element).click().perform();
```

Methods of Action Class

Action class is useful mainly for mouse and keyboard actions. In order to perform such actions, Selenium provides various methods.

Mouse Actions in Selenium:

1. **doubleClick()**: Performs double click on the element
2. **clickAndHold()**: Performs long click on the mouse without releasing it
3. **dragAndDrop()**: Drags the element from one point and drops to another
4. **moveToElement()**: Shifts the mouse pointer to the center of the element
5. **contextClick()**: Performs right-click on the mouse

Keyboard Actions in Selenium:

1. **sendKeys()**: Sends a series of keys to the element
2. **keyUp()**: Performs key release
3. **keyDown()**: Performs keypress without release

1) Keyboard down button functionality using Action class(Selecting drop down)

```
package ActionClass;


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.interactions.Actions;



/*Date:11/09/2021

 Program to Handling DropDown(Keyboard) using Action Class */


public class KeyBoardDropDown {


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

System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
```

```
WebDriver driver =new ChromeDriver();

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

driver.get("https://vctcpune.com/selenium/practice.html#");
        Thread.sleep(4000);

        JavascriptExecutor js=(JavascriptExecutor) driver;
        js.executeScript("window.scrollBy(0,300)");
        Thread.sleep(4000);

        WebElement  dropdown  =driver.findElement(By.id("dropdown-class-example"));

        //Step-6-Handling drop down using actions class
        //step-a
        Actions as=new Actions(driver);//Action class takes argument as driver
        as.click(dropdown).perform();//Click takes argument as Web Element
                    //Perform is used to execute action i.e. click action
```

```
//Till above step dropdown will open now we have select third option from that dropdown.i.e. in downward direction.


        //as.sendKeys(Keys.ARROW_DOWN).perform();//For selecting second option from drop down.

//Here select Sendkeys(char sequence...keys)Action method


//When we have to select third option from drop down then we have write like follows:

         as.sendKeys(Keys.ARROW_DOWN).sendKeys(Keys.ARROW_DOWN).build().perform();

        }


}
```

2) Mouse Double click on button.

```
package ActionClass;


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.interactions.Actions;
```

```
/*Date:11/03/2021

Program to Perform Right click operation*/



public class MouseDoubleClick {



        public static void main(String[] args) {



System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");



   WebDriver driver =new ChromeDriver();



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



   driver.get("https://demoqa.com/buttons");



    //Step-4- Double click using actions class

    //step-a

          Actions as=new Actions (driver);



    //Locating Web Element using By ID
```

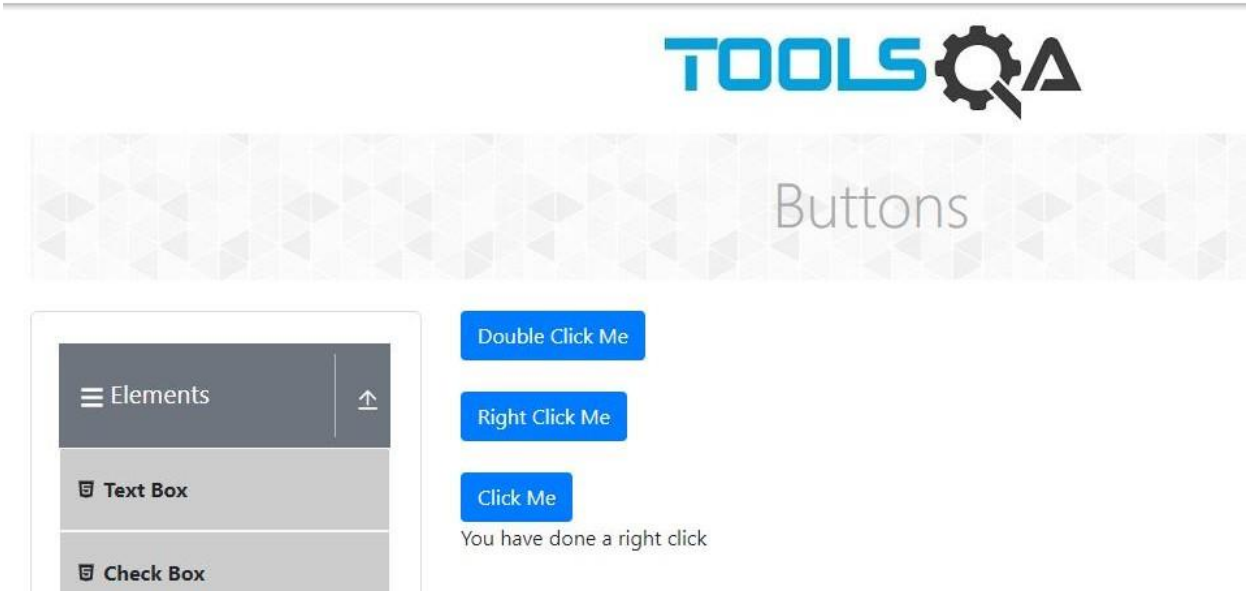
```
        //WebElement button=driver.findElement(By.id("doubleClickBtn"));
        //as.doubleClick(button).perform();

    //Locating Web Element By using XPath
        WebElement button1=driver.findElement(By.xpath("//button[@id='doubleClickBtn']"));
        as.doubleClick(button1).perform();
        }
}
```

3) Mouse Right click operation on button.

```
package ActionClass;

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.interactions.Actions;

/*Date:11/03/2021
 Program to Perform Right click operation*/

public class MouseRightClick {

                public static void main(String[] args) {

                 System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay                Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

                  WebDriver driver =new ChromeDriver();

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

                  driver.get("https://demoqa.com/buttons");

                   //Step-4- Right click using actions class
                   //step-a
                  Actions as=new Actions (driver);

                   // Locating Web Element using By ID
                  //WebElement button=driver.findElement(By.id("rightClickBtn"));
                  //as.contextClick(button).perform();
```

```
                    //Locating Web Element By using XPath
 WebElement button1=driver.findElement(By.xpath("//button[@id='rightClickBtn']"));
                    as.contextClick(button1).perform();
                   }

}
```

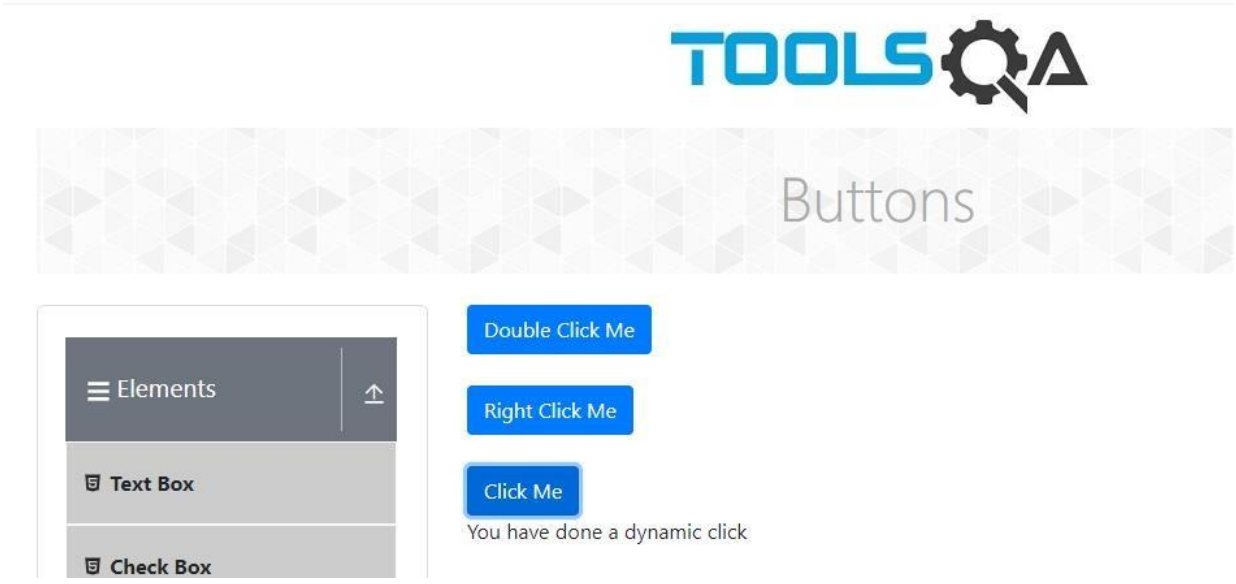


3)Mouse Single click operation on button.

```
package ActionClass;


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.interactions.Actions;

public class MouseClick {

                public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

                WebDriver driver =new ChromeDriver();

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

                driver.get("https://demoqa.com/buttons");

                 //Step-4- Double click using actions class

                 //step-a

                Actions as=new Actions (driver);

                 //Locating Web Element using By ID

                //WebElement button=driver.findElement(By.xpath("//button[text()='Click Me']"));

                //as.click(button).perform();

 //as.moveToElement(button).click().build().perform(); This method also gives same output

                 //Locating Web Element By using XPath

WebElement button1=driver.findElement(By.xpath("//button[text()='Click Me']"));

                as.click(button1).perform();

//as.moveToElement(button1).click().build().perform(); This method also gives same output
```

```
        }

}
```


TOOLSQA
Buttons
Elements
Text Box
Check Box
Double Click Me
Right Click Me
Click Me
You have done a dynamic click

13) Selenium 13 Lecture

Drag and Drop in Selenium

dragAndDrop(WebElement source, WebElement target): This method performs left click, hold the click to hold the source element, moves to the location of the target element and then releases the mouse click.

Let's see how to use Action class methods to perform drag-drop action:

First, instantiate an Actions class:

Actions actions = new Actions(driver);

As you can see, the dragAndDrop(WebElement source, WebElement target) method has two arguments to pass. One is a source web element and another is target web element. This source web element is any web element that needs to be dragged. Target web element is any web element on which dragged object needs to be placed or dropped. To find the source and target element use the below command:

WebElement source = driver.findElement(Any By strategy & locator);

WebElement target = driver.findElement(Any By strategy & locator);

Here, you can use any By strategy to locate the WebElement like find element by its id, name attribute, etc.

Now, when we have got the actions class object and the element as well, just invoke perform() method for the drag & drop:

actions.dragAndDrop(source,target).perform();

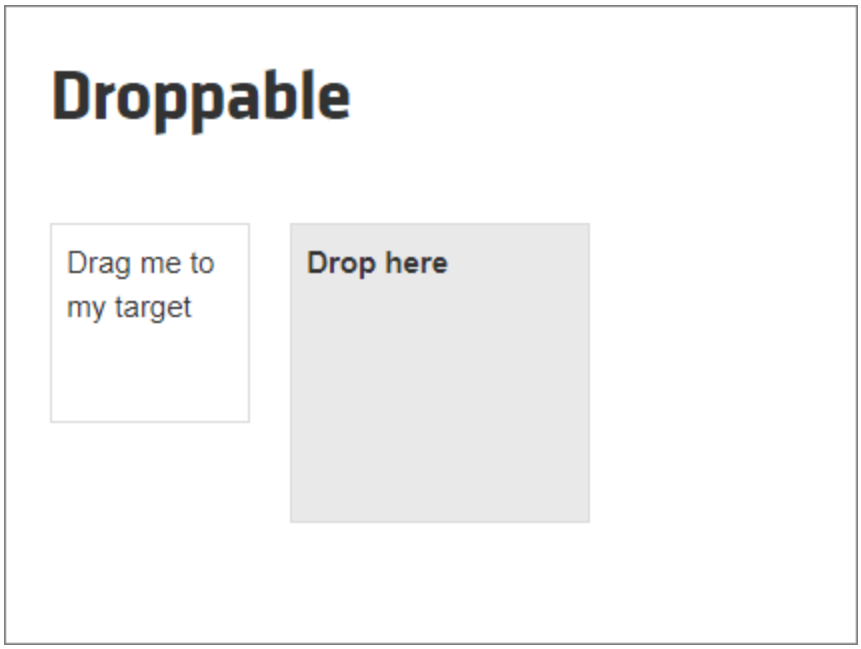
Let's see what happens internally when invoke the perform() method above:

- Click And Hold Action: dragAndDrop() method first performs click-and-hold at the location of the source element
- Move Mouse Action: Then source element gets moved to the location of the target element
- Button Release Action: Finally, it releases the mouse

- **Build:** build() method is used to generate a composite action containing all actions. But if you observe, we have not invoked it in our above command. The build is executed in the perform method internally
- **Perform:** perform() method performs the actions we have specified. But before that, it internally invokes build() method first. After the build, the action is performed.

Practice Exercise to Perform Drag and Drop using Actions Class in Selenium

Let's consider an example from an already available demo page on Toolsqa as **http://demoqa.com/droppable/**



In the above image, *'Drag me to my target'* object can be selected and dragged and dropped at the *'Drop here'* object. Following message gets displayed once the object is dropped at '*Drop here*'

Find below the steps of the scenario to be automated:

1. Launch the web browser and launch our practice page https://demoqa.com/droppable/
2. Find the required source element i.e. 'Drag me to my target' object in our sample
3. Find the required target element i.e. 'Drop Here' object in our sample
4. Now Drag and Drop 'Drag me to my target' object to 'Drop Here' object
5. Verify message displayed on 'Drop Here' to verify that source element is dropped at the target element
6. Close the browser to end the program

In following program1, we get we page but our required web element is in the particular one frame. That Frame is bind (place) inside the web page.

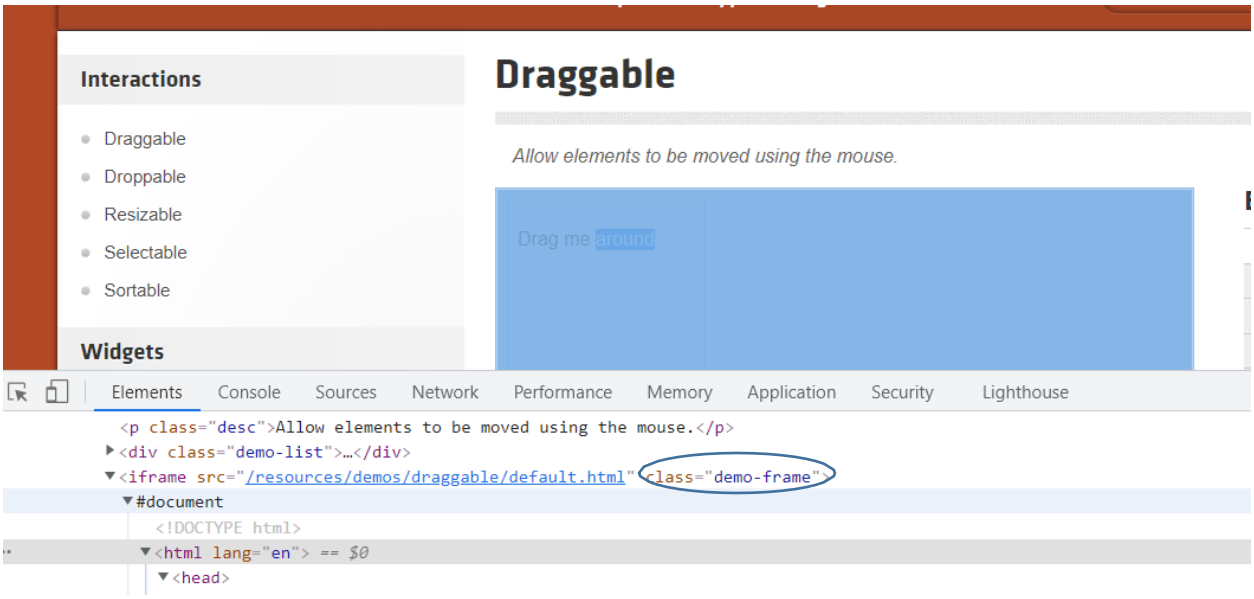
Therefore we need to go that Frame i.e. We use **SwitchTo()** method for switching from web page to Frame.

SwitchTo() is a method of Web Driver.

In following program1,we have performed various steps for dragging.

Strp4:Use of JavaScriptExecutor Interface for scrolling down window.

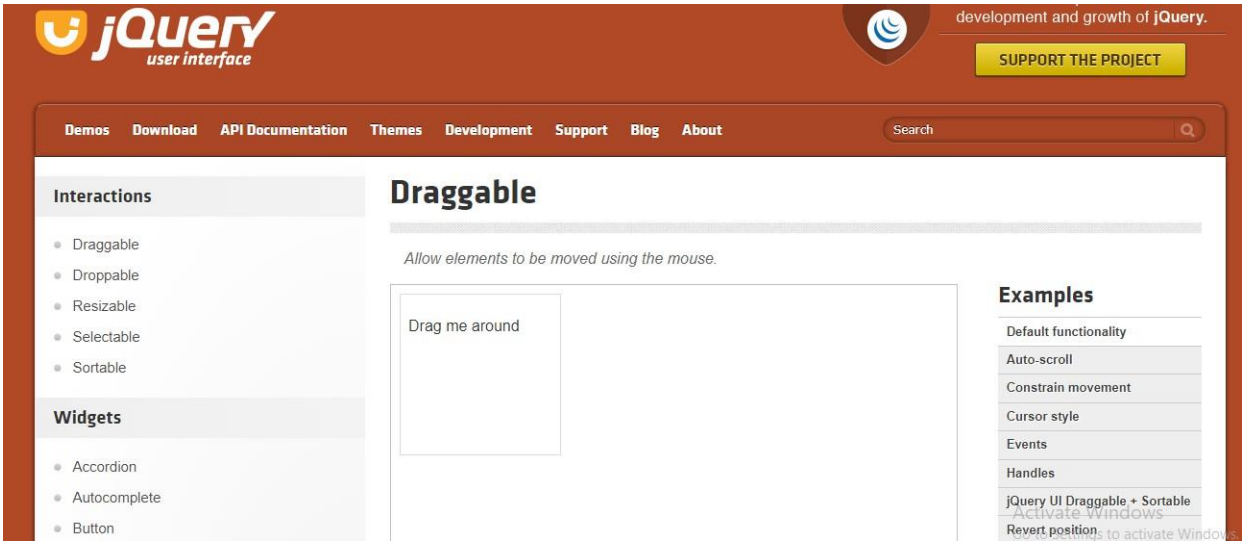
Step5:Identifying frame within a web page.



Step6:Switching to Frame from Web page.

Step7:Locate Drag Element by ID

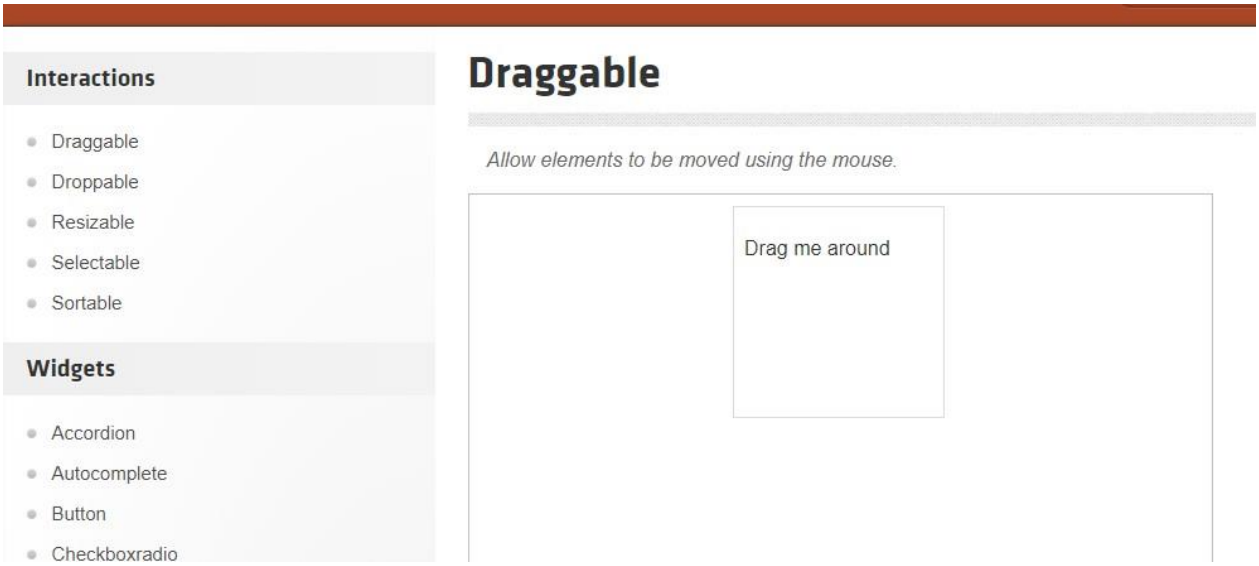
Before Program



Program1:Script to perform Drag operation using Action Class

```
package ActionClass;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

/*Date:12/029/2021

 Script to Perform Drag operation using Action class*/

public class Drag {

     public static void main(String[] args) throws
InterruptedException {
System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

          WebDriver driver =new ChromeDriver();

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

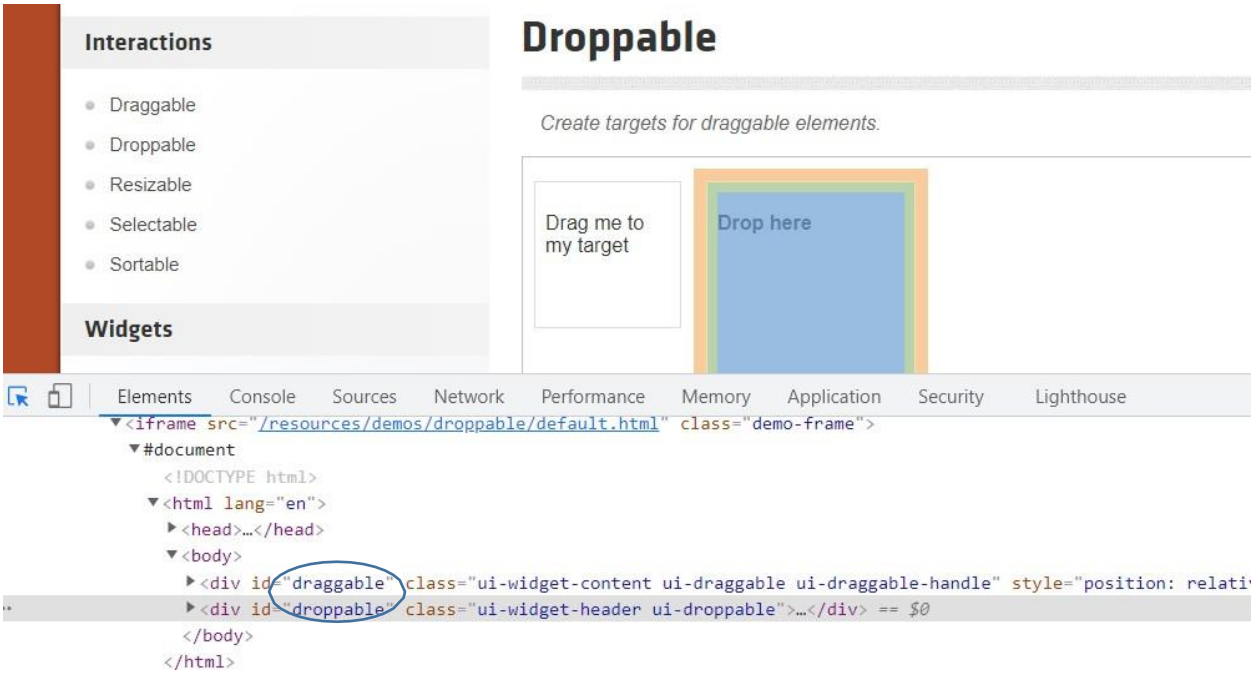
```
            driver.get("https://jqueryui.com/draggable/");

            //Step4 Use of JavascriptExecutor Interface.
            JavascriptExecutor je=(JavascriptExecutor) driver;
            je.executeScript("window.scrollBy(0,200)");

            //Step5
            //Identifying Frame
            WebElement
    frame2=driver.findElement(By.xpath("//*[@class='demo-frame']"));

            //Step6
            //Switching to Frame
            driver.switchTo().frame(frame2);

            //Step7
            WebElement drag2=driver.findElement(By.id("draggable"));

            //Step8 Action Class & Object Creation.
            Actions as=new Actions(driver);

            //Step9 Dragging operation
            as.dragAndDropBy(drag2, 200, 0).build().perform();

            //Step10
            Thread.sleep(5000);
            driver.close();




        }

    }
```



Program2:Script to perform Drag & Drop Operation using Action Class

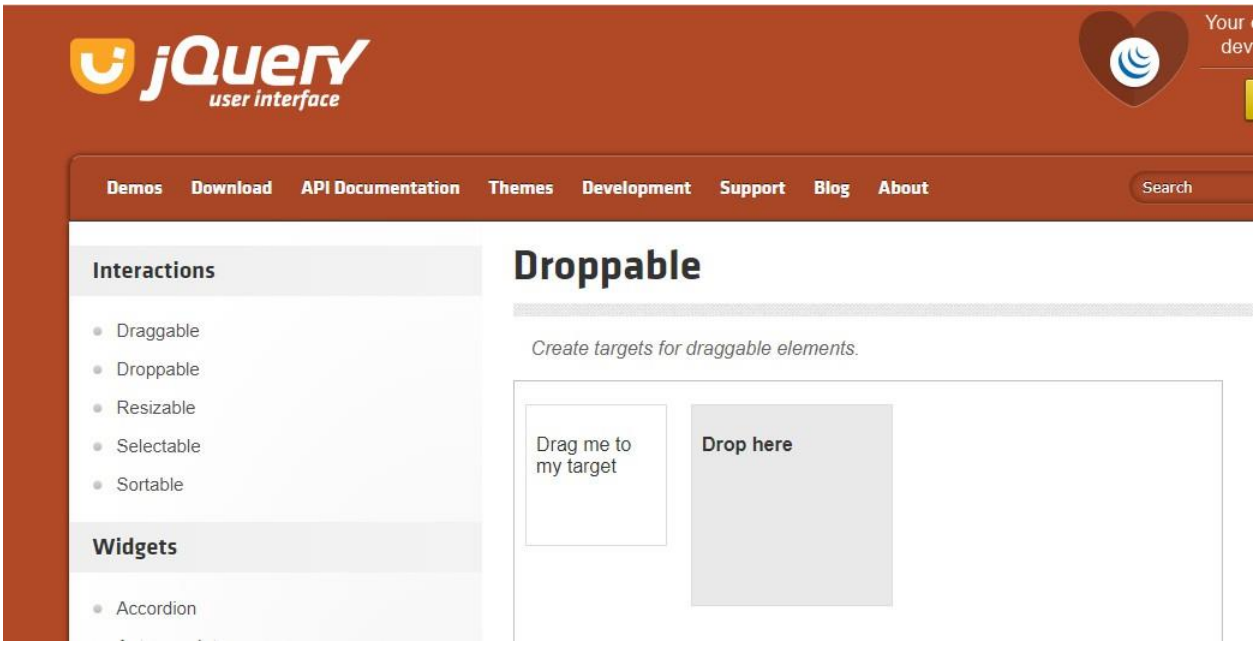
Step7:i)Locate Drag Element by ID

ii)Locate Drop Element by ID



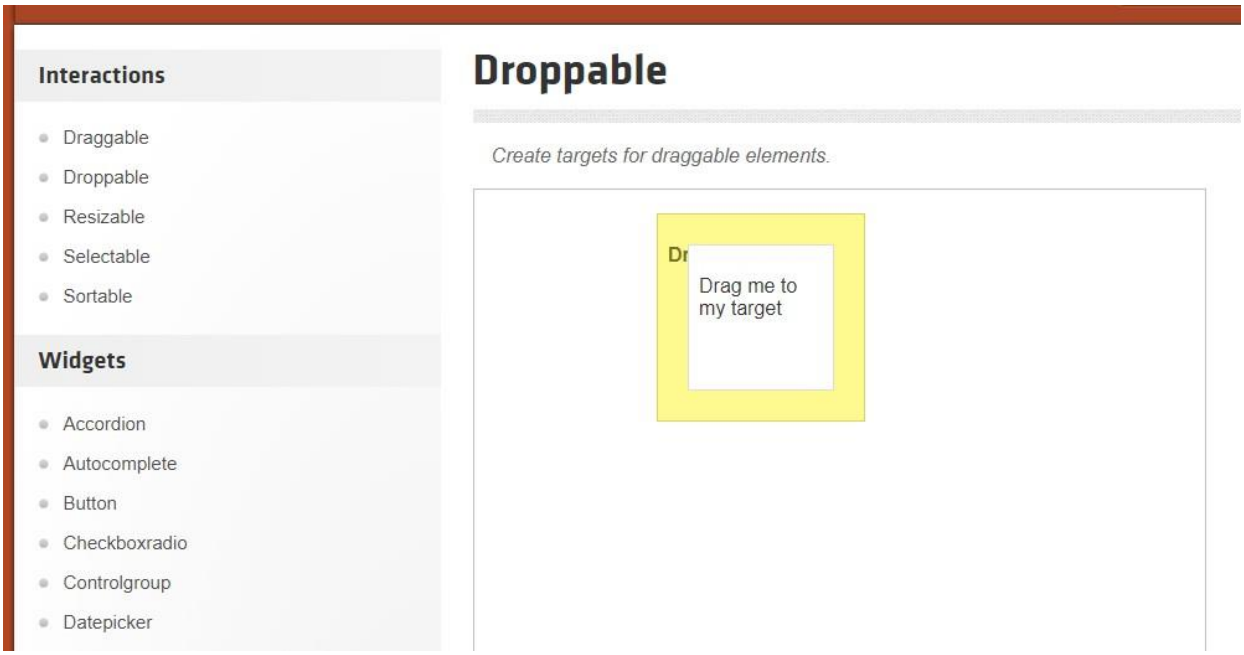
Step9:Use dragAnddrop(WebElement source,WebElement target) method

Before Program:




```
package ActionClass;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

/*Date:12/029/2021

 Script to Perform Drag & Drop operation using Action class*/

public class DragAndDrop {

     public static void main(String[] args) throws
InterruptedException {
System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

          WebDriver driver =new ChromeDriver();

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

          driver.get("https://jqueryui.com/droppable/");
```

```
            //Step4 Use of JavascriptExecutor Interface.
            JavascriptExecutor je=(JavascriptExecutor) driver;
            je.executeScript("window.scrollBy(0,200)");

            //Step5
            //Identifying Frame
            WebElement
    frame2=driver.findElement(By.xpath("//*[@class='demo-frame']"));

            //Step6
            //Switching to Frame
            driver.switchTo().frame(frame2);

            //Step7
            WebElement drag2=driver.findElement(By.id("draggable"));
            WebElement drop2=driver.findElement(By.id("droppable"));

            //Step8 Action Class & Object Creation.
            Actions as=new Actions(driver);

            //Step9 Drag & Drop operation
            as.dragAndDrop(drag2, drop2).build().perform();

            //Step10
            Thread.sleep(5000);
            driver.close();

            }

    }
```


Interactions
Draggable
Droppable
Resizable
Selectable
Sortable
Widgets
Accordion
Autocomplete
Button
Checkboxradio
Controlgroup
Datepicker
Droppable
Create targets for draggable elements.
Drag me to my target

Slider Using Action Class

Sliders allow users to select a value by dragging and dropping a handle.

This value can be a price, a quantity, a year.

The web page could use a textbox for getting the same information from users.

But **with sliders, the page becomes much more interesting.**

Actually, many web pages do not use the slider or the text box but both of them as in the following image:

The slider stores its current position (as percentage) at all times.

The current position can be found in one of the slider's attributes.

The current slider position corresponds to a value.
For example, if the slider is at 50%, this corresponds to a price of 500000.
If the slider is at 10%, the corresponding price is 100000.

The value that corresponds to the slider's position is usually saved in a hidden element.

Use **ClickAndHold() & Release()** methods for Slider.

In following program2, we get we page but our required web element is in the particular one frame. That Frame is bind (place) inside the web page.

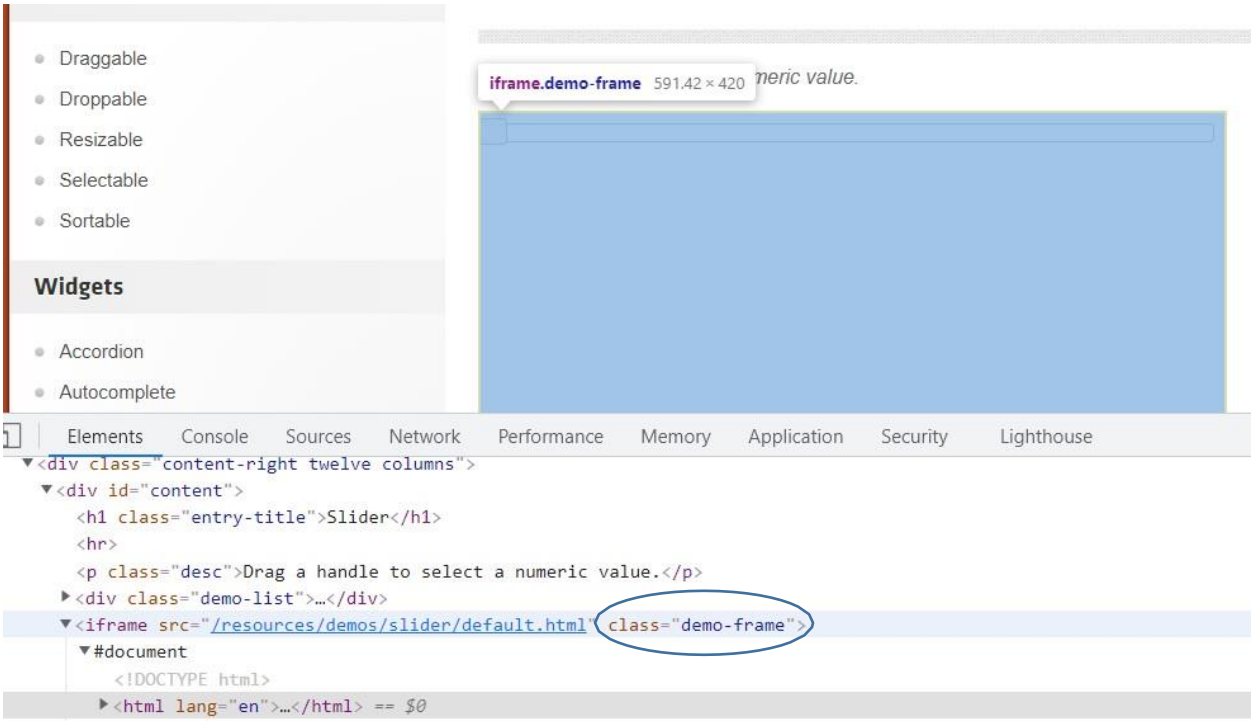
Therefore we need to go that Frame i.e. We use **SwitchTo()** method for switching from web page to Frame.

SwitchTo() is a method of Web Driver.

In following program2,we have performed various steps for dragging.

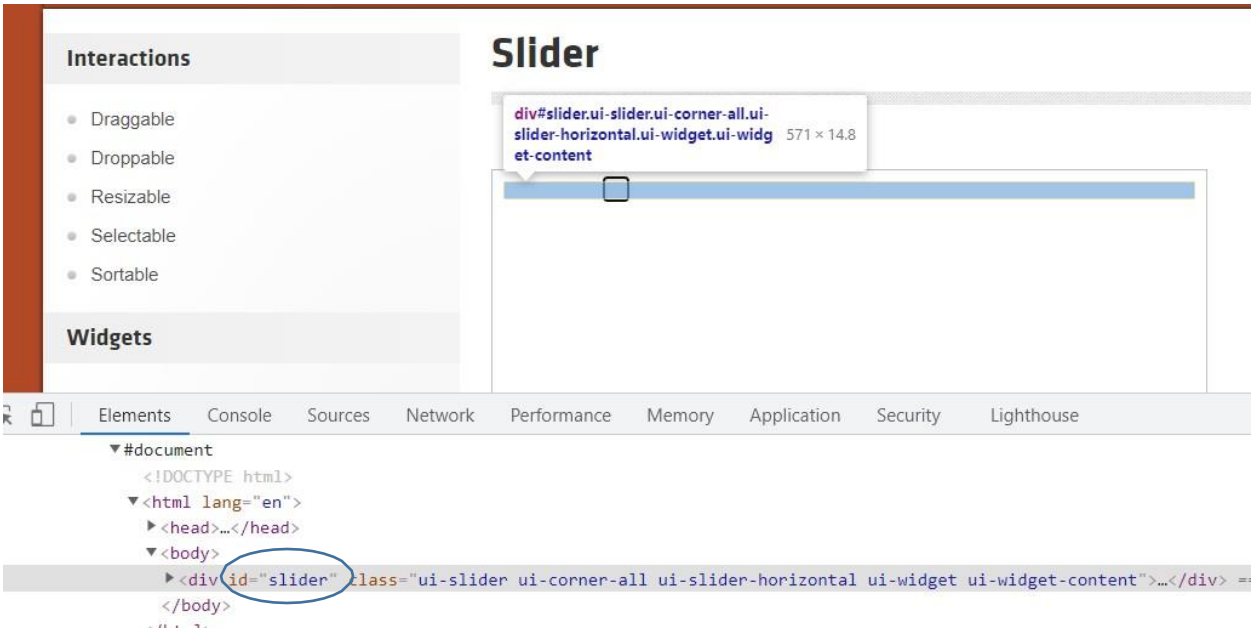
Strp4:Use of JavaScriptExecutor Interface for scrolling down window.

Step5:Identifying frame within a web page.

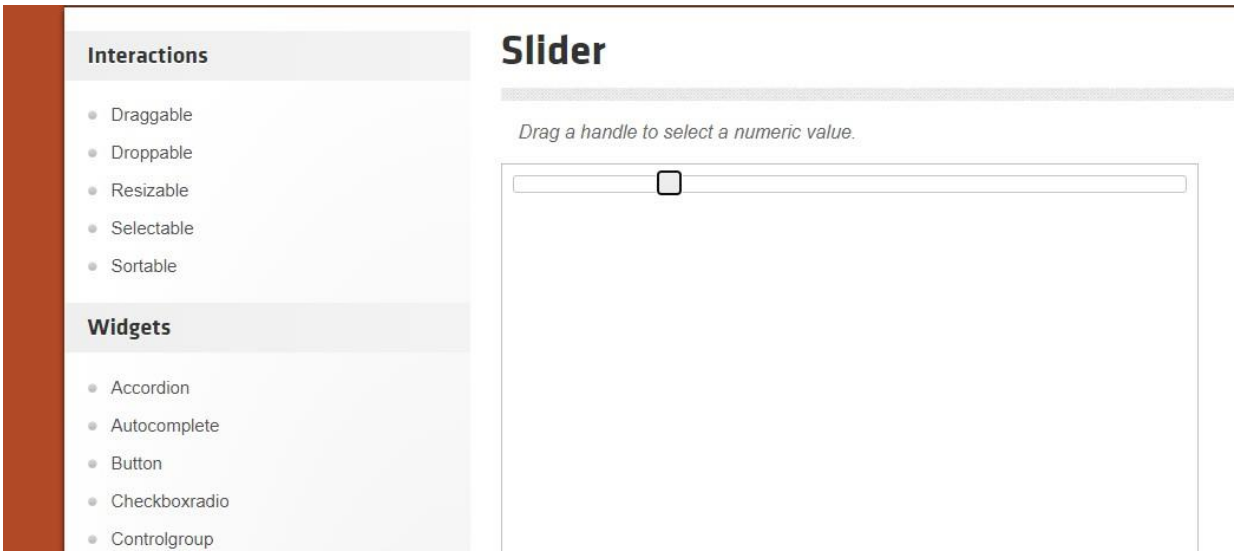


Step6:Switching to Frame from Web page.

Step7:Locate Slider Element by ID



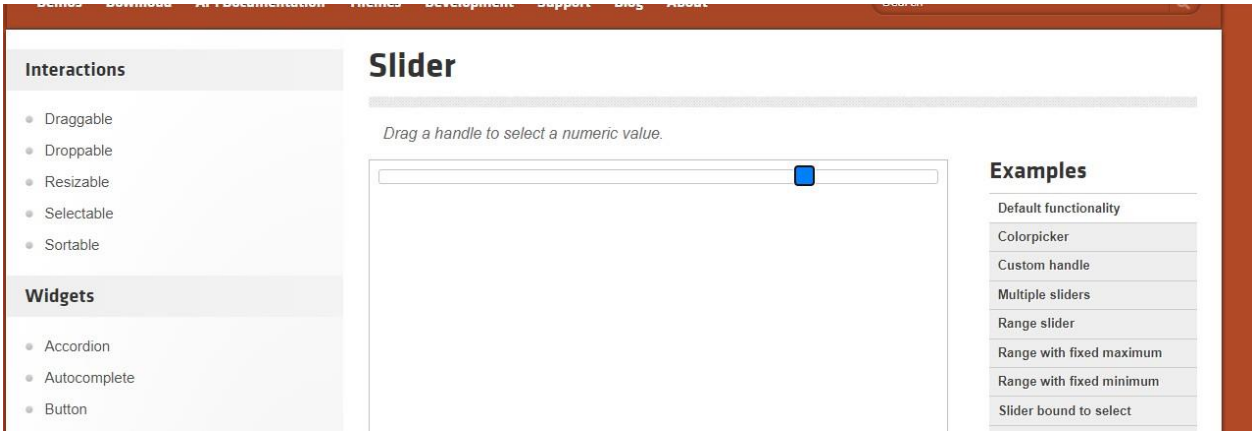
Before Program:



Program2: Script to Perform Slider operation(going forward direction & Release by element is in frame of a web page) using Action Class.

```
package ActionClass;


import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;


/*Date 12/09/2021

 Script to Perform Slider operation(going forward direction only) using Action Class without Release.*/


public class Slider {


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

System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

WebDriver driver =new ChromeDriver();


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

driver.get("https://jqueryui.com/slider/");


//Step4 Use of JavascriptExecutor Interface.

JavascriptExecutor je=(JavascriptExecutor) driver;

je.executeScript("window.scrollBy(0,200)");
```

```
 //Step5
 //Identifying Frame
 WebElement   frame3=driver.findElement(By.xpath("//*[@class='demo-frame']"));

//Step6
//Switching to Frame
driver.switchTo().frame(frame3);

//Step7
 WebElement slider2=driver.findElement(By.id("slider"));

 //Step8 Action Class & Object Creation.
Actions as=new Actions(driver);

//Step9 Perform Slider operation
  as.clickAndHold(slider2).moveByOffset(150, 0).perform();

//Step10
 Thread.sleep(5000);
 driver.close();
}

}
```

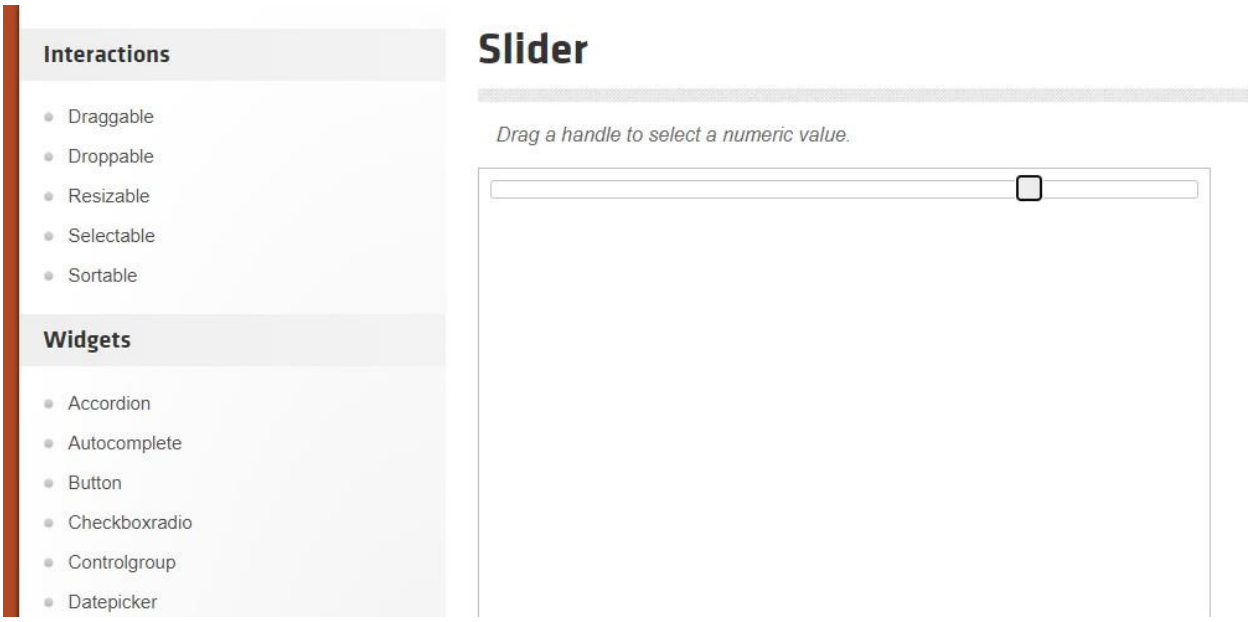


Program3: Script to Perform Slider operation(going forward direction only & release) using Action Class*/

```
package ActionClass;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

/*Date 12/09/2021
 Script to Perform Slider operation(going forward direction only & release) using Action Class*/

public class Slider2 {

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

```
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
 WebDriver driver =new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://jqueryui.com/slider/");

 //Step4 Use of JavascriptExecutor Interface.
JavascriptExecutor je=(JavascriptExecutor) driver;
je.executeScript("window.scrollBy(0,200)");

//Step5
//Identifying Frame
WebElement   frame3=driver.findElement(By.xpath("//*[@class='demo-frame']"));

//Step6
//Switching to Frame
driver.switchTo().frame(frame3);

//Step7
WebElement slider2=driver.findElement(By.id("slider"));

//Step8 Action Class & Object Creation.
Actions as=new Actions(driver);

//Step9 Perform Slider operation
as.clickAndHold(slider2).moveByOffset(150, 0).release().perform();
```

```
//Step10
Thread.sleep(5000);
driver.close();
}
}
```

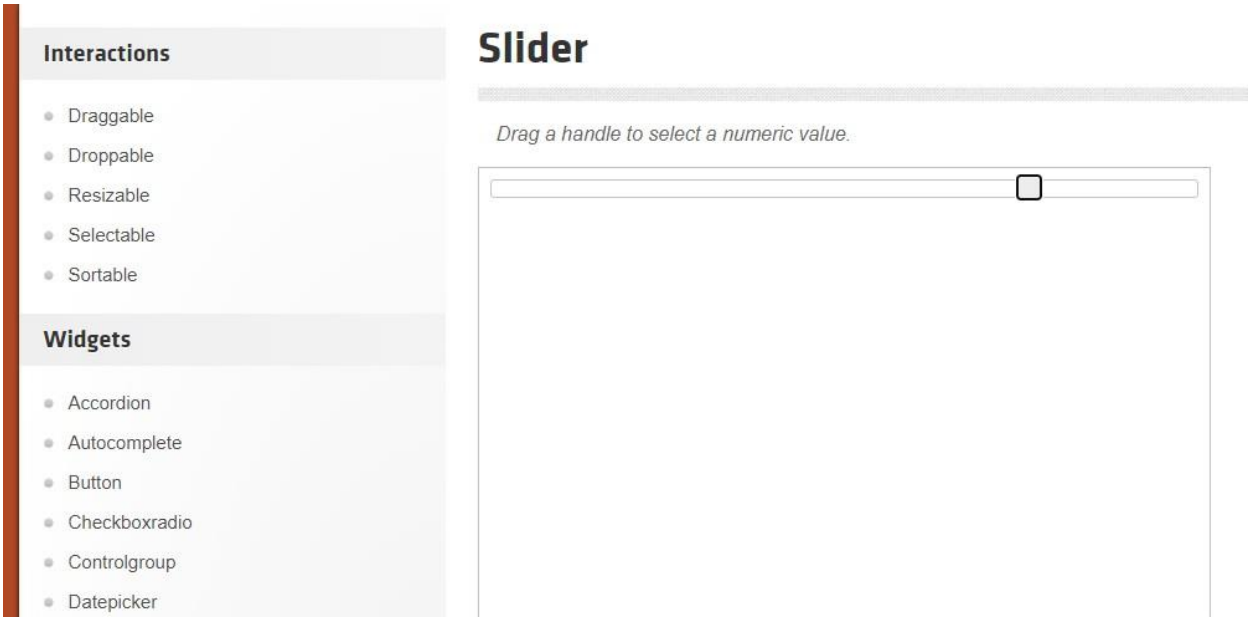


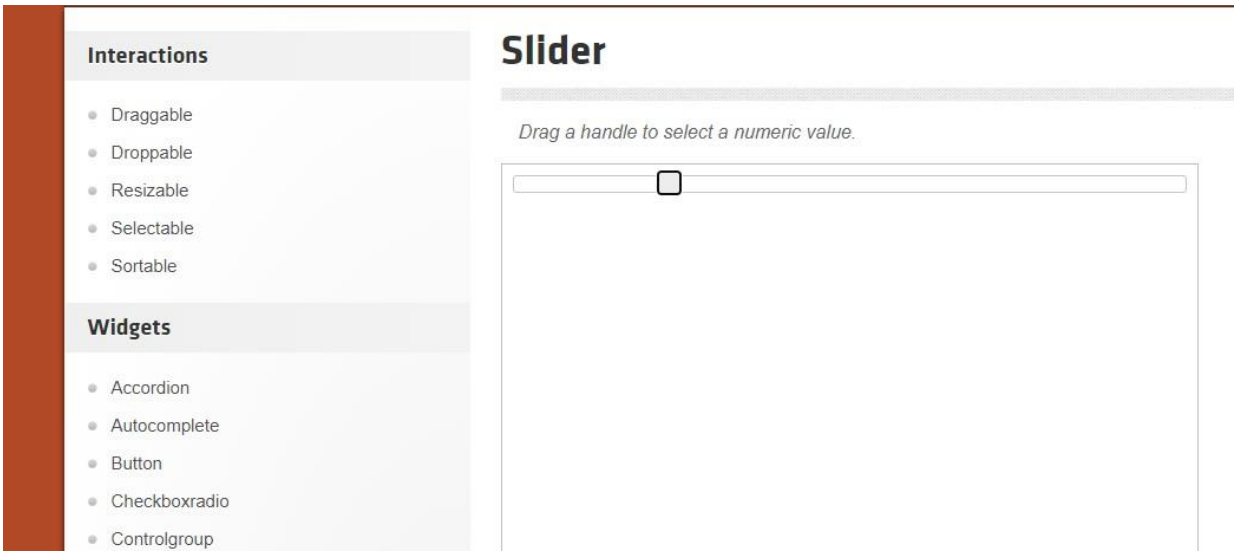
Program3: Script to Perform Slider operation(going forward direction & Release & backward direction by element is in frame of a web page) using Action Class.

```
package ActionClass;


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

```
/*Date 12/09/2021
 Script to Perform Slider operation(going forward direction only & release & in backward direction)
 i.e.Moving back to original position using Action Class*/

public class Slider3 {
public static void main(String[] args) throws InterruptedException {
System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

WebDriver driver =new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://jqueryui.com/slider/");

 //Step4 Use of JavascriptExecutor Interface.
JavascriptExecutor je=(JavascriptExecutor) driver;
je.executeScript("window.scrollBy(0,200)");

//Step5
//Identifying Frame
WebElement  frame3=driver.findElement(By.xpath("//*[@class='demo-frame']"));

//Step6
//Switching to Frame
driver.switchTo().frame(frame3);

//Step7
```

```
WebElement slider2=driver.findElement(By.id("slider"));

//Step8 Action Class & Object Creation.
Actions as=new Actions(driver);

//Step9 Perform Slider operation
as.clickAndHold(slider2).moveByOffset(150, 0).release().perform();
Thread.sleep(5000);

//Step-10- Going back to original position
as.clickAndHold(slider2).moveByOffset(-150, 0).release().perform();
Thread.sleep(5000);

//Step11
driver.close();
}

}
```


Interactions
Draggable
Droppable
Resizable
Selectable
Sortable
Widgets
Accordion
Autocomplete
Button
Checkboxradio
Controlgroup
Datepicker
Slider
Drag a handle to select a numeric value.

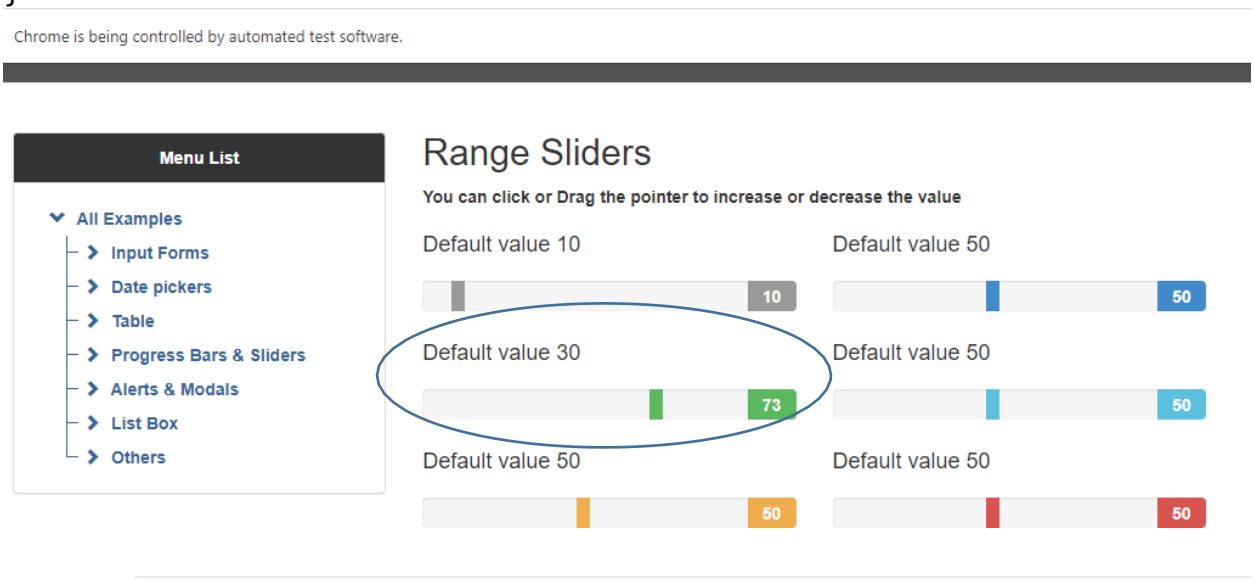

Interactions
Draggable
Droppable
Resizable
Selectable
Sortable
Widgets
Accordion
Autocomplete
Button
Checkboxradio
Controlgroup
Slider
Drag a handle to select a numeric value.

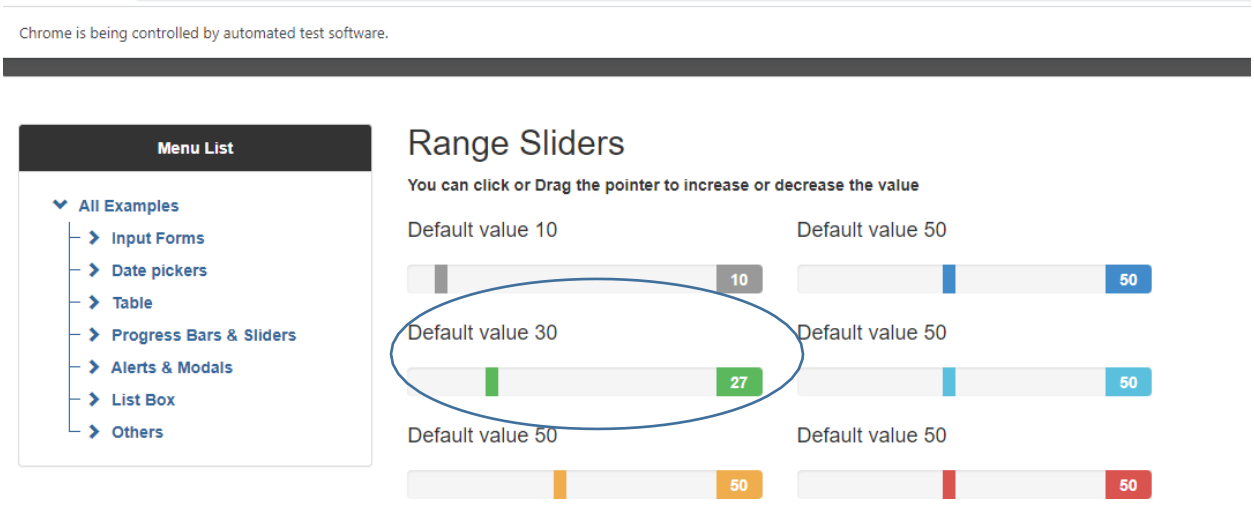
Program4: Script to Perform Slider operation(going forward direction & Release & backward direction without frame means all elements are in single web page) using Action Class.

Web Page URL: https://www.seleniumeasy.com/test/drag-drop-range-sliders-demo.html");

```
package ActionClass;

/*Date 12/09/2021
 Script to perform Slider operation without frame web page
 Web Page:https://www.seleniumeasy.com/test/drag-drop-range-sliders-demo.html*/

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

public class Slider4 {

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

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

		WebDriver driver =new ChromeDriver();

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

		driver.get("https://www.seleniumeasy.com/test/drag-drop-
range-sliders-demo.html");


		JavascriptExecutor je=(JavascriptExecutor) driver;
		je.executeScript("window.scrollBy(0,200)");

		WebElement
slider4=driver.findElement(By.xpath("//input[@min='1'][@max='100'][@va
lue='30']"));
```

```
        Actions as= new Actions(driver);


as.clickAndHold(slider4).moveByOffset(60,0).release().perform();
        Thread.sleep(5000);

        as.clickAndHold(slider4).moveByOffset(-60,
0).release().perform();
        Thread.sleep(5000);

        driver.close();
        }

}
```





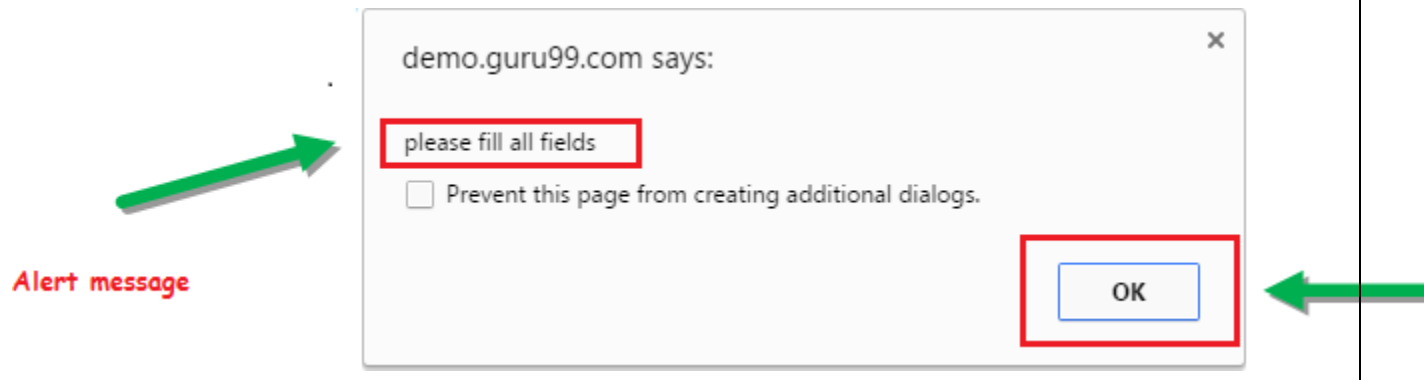
14) Selenium 14 Lecture

Alert Message Popup Handling in Selenium

An **Alert in Selenium** is a small message box which appears on screen to give the user some information or notification. It notifies the user with some specific information or error, asks for permission to perform certain tasks and it also provides warning messages as well.
Here are few alert in Selenium types:

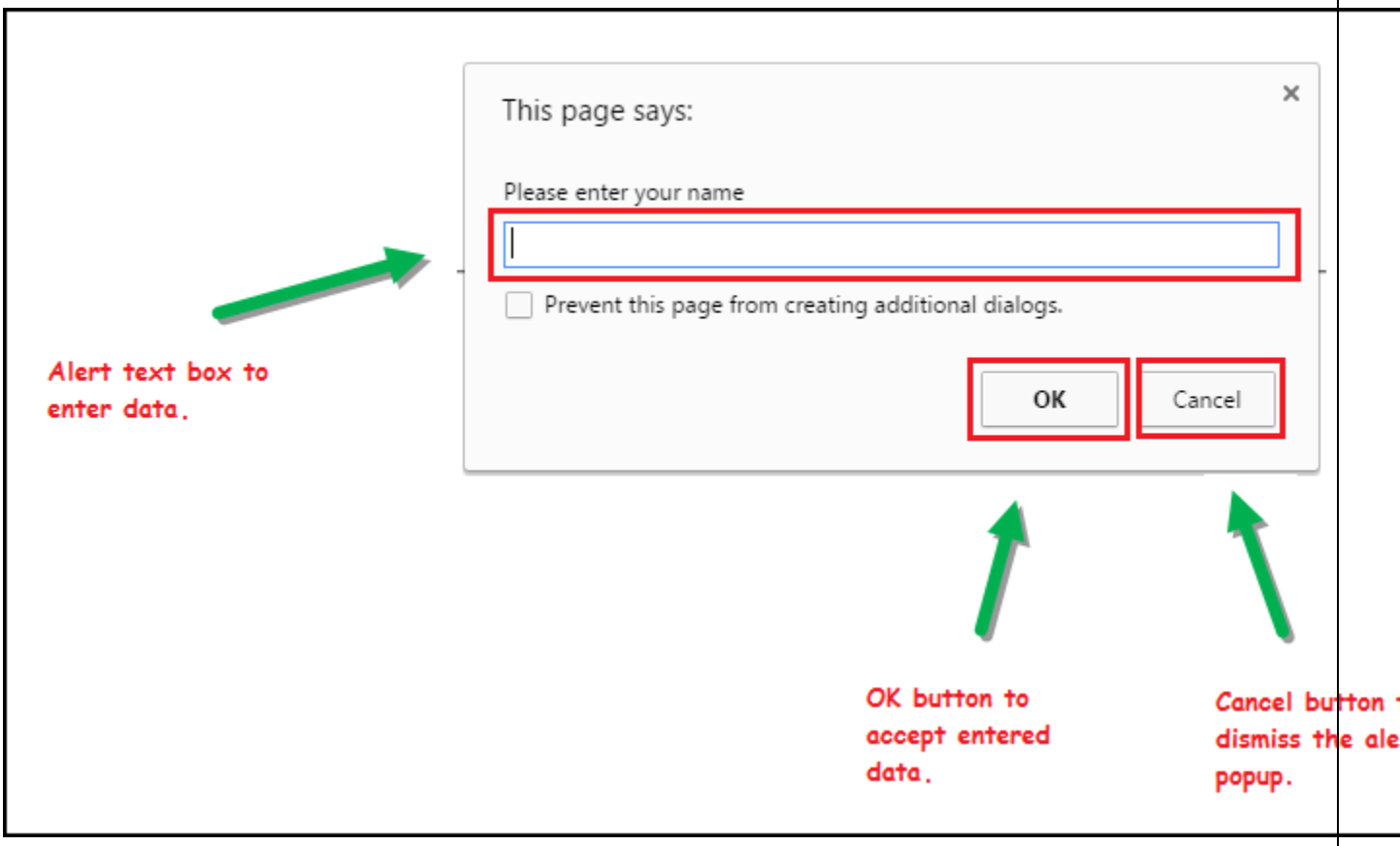
1) Simple Alert

The simple alert class in Selenium displays some information or warning on the screen.



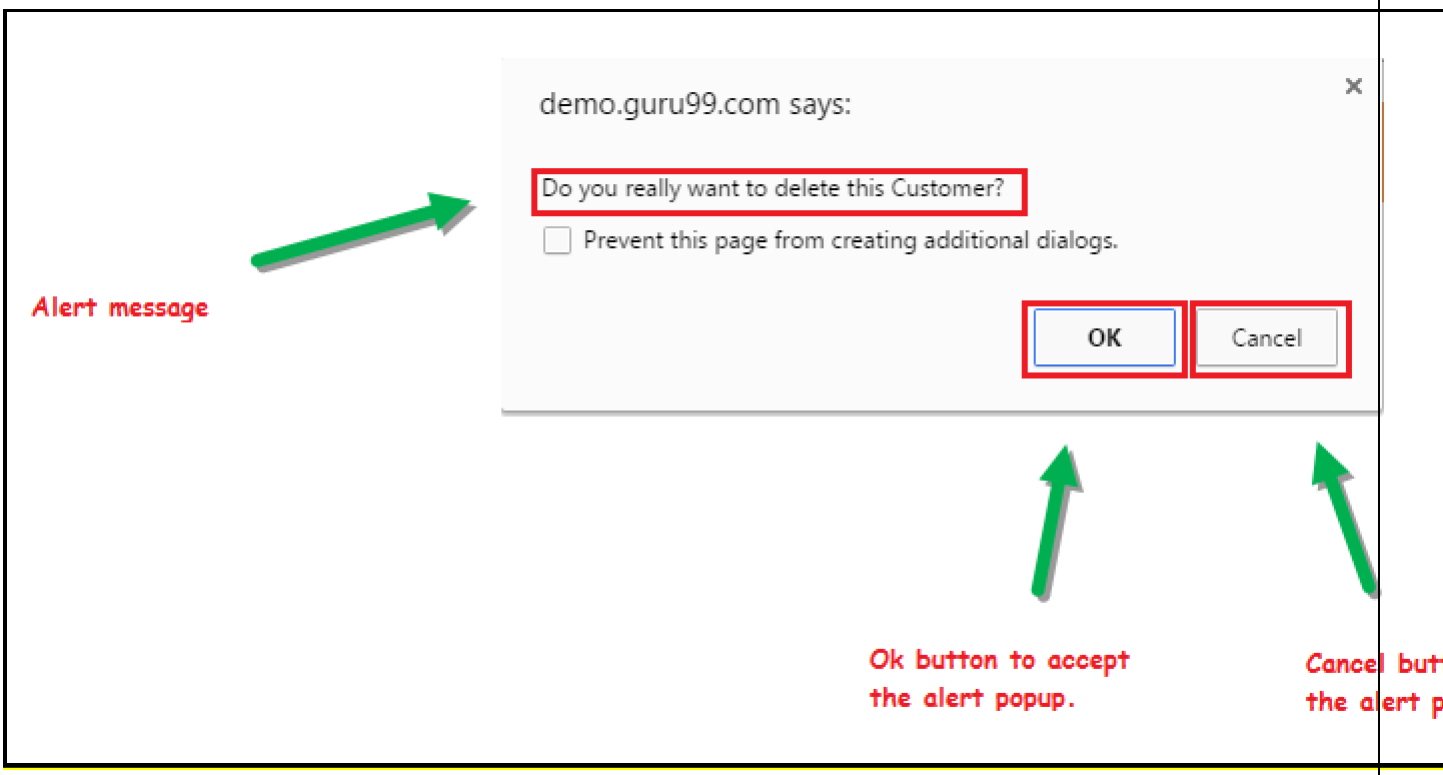
2) Prompt Alert.

This Prompt Alert asks some input from the user and Selenium webdriver can enter the text using sendkeys(" input…. ").



3) Confirmation Alert.

This confirmation alert asks permission to do some type of operation.



How to handle Alert in Selenium WebDriver

Alert interface provides the below few methods which are widely used in Selenium Webdriver.

1) void dismiss() **// To click on the 'Cancel' button of the alert.**

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

2) void accept() **// To click on the 'OK' button of the alert.**

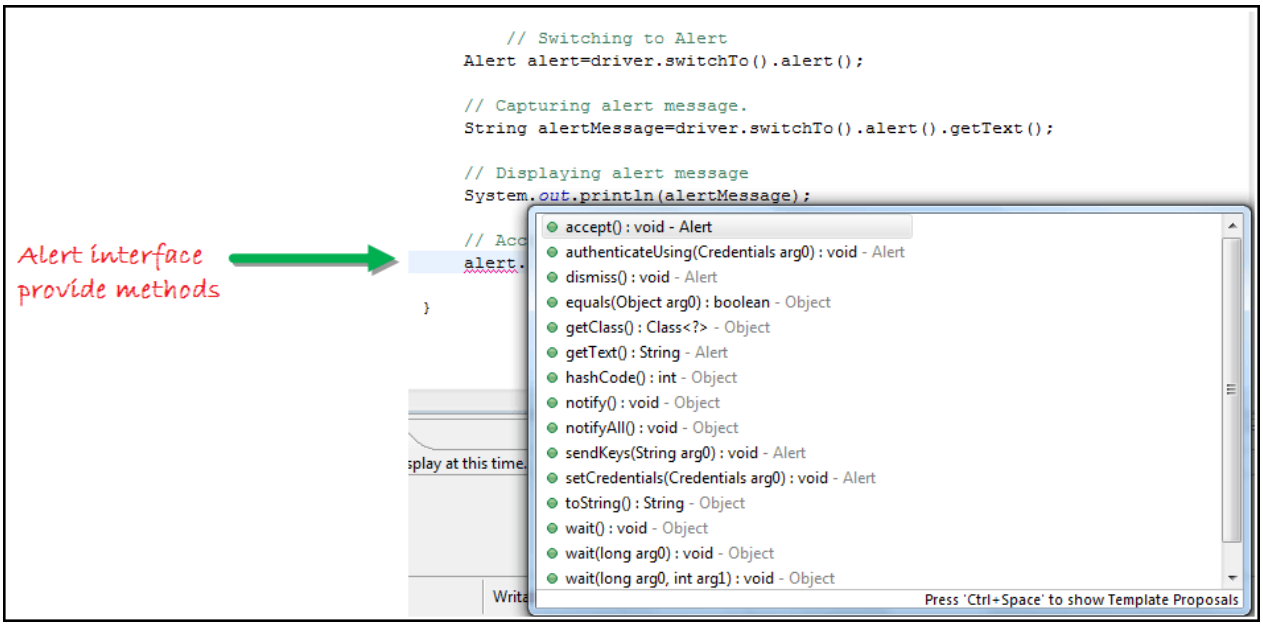
driver.switchTo().alert().accept();

3) String getText**() // To capture the alert message.**

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

4) void sendKeys(String stringToSend) **// To send some data to alert box.**

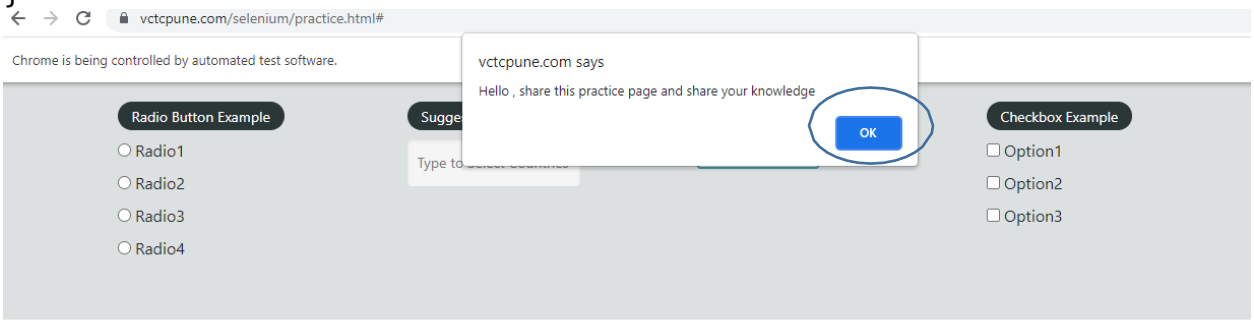
driver.switchTo().alert().sendKeys("Text");

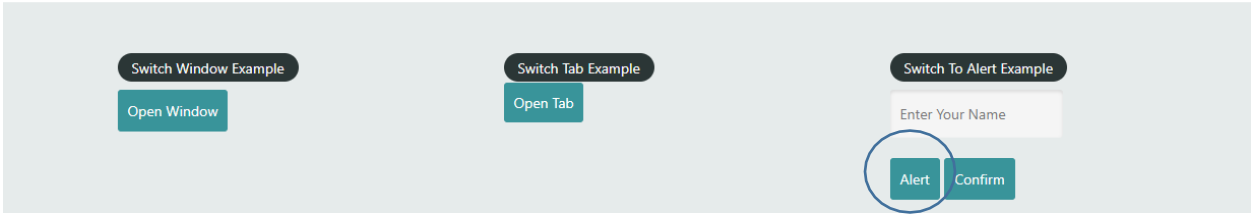


Program1:Alert handling by Accept Alert message.

```
package ActionClass;

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

/*Date:14/09/2021

Script for Alert Handling.Accept an Alert message*/

public class AlertPopup {

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

```
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

    WebDriver driver=new ChromeDriver();

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

    driver.get("https://vctcpune.com/selenium/practice.html#");

    JavascriptExecutor je=(JavascriptExecutor) driver;
    je.executeScript("window.scrollBy(0,700)");
    Thread.sleep(4000);

    driver.findElement(By.id("alertbtn")).click();
    Thread.sleep(4000);

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


    }

}
```

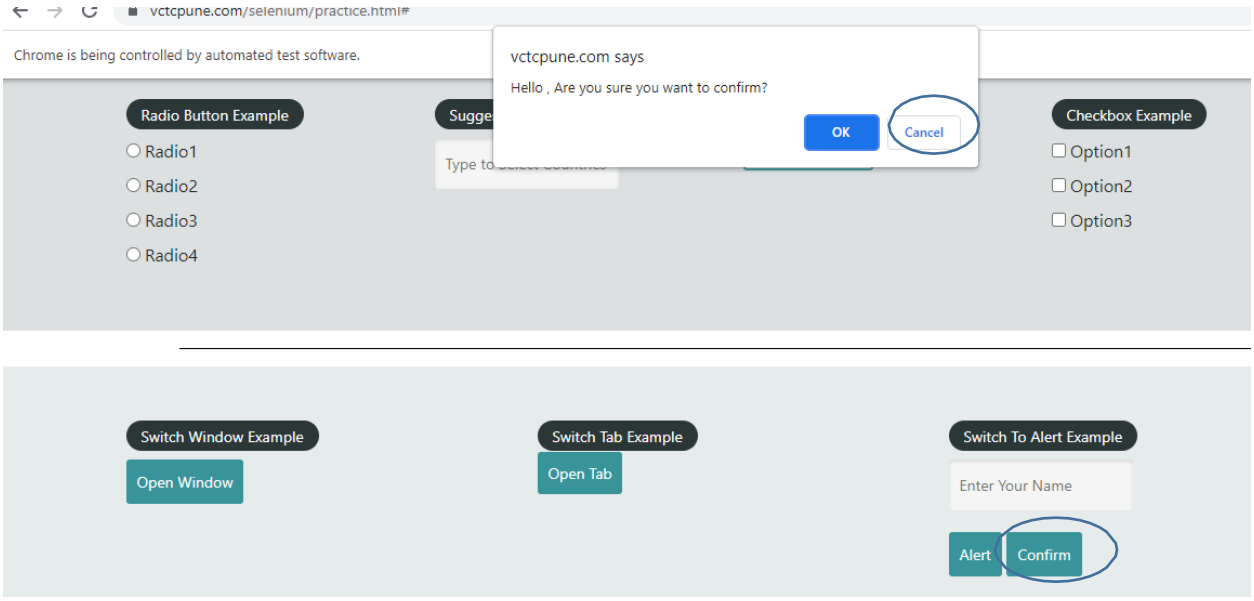




Program2:Alert handling by Dismiss or Cancel Alert message.

```
package ActionClass;

import org.openqa.selenium.By;
```

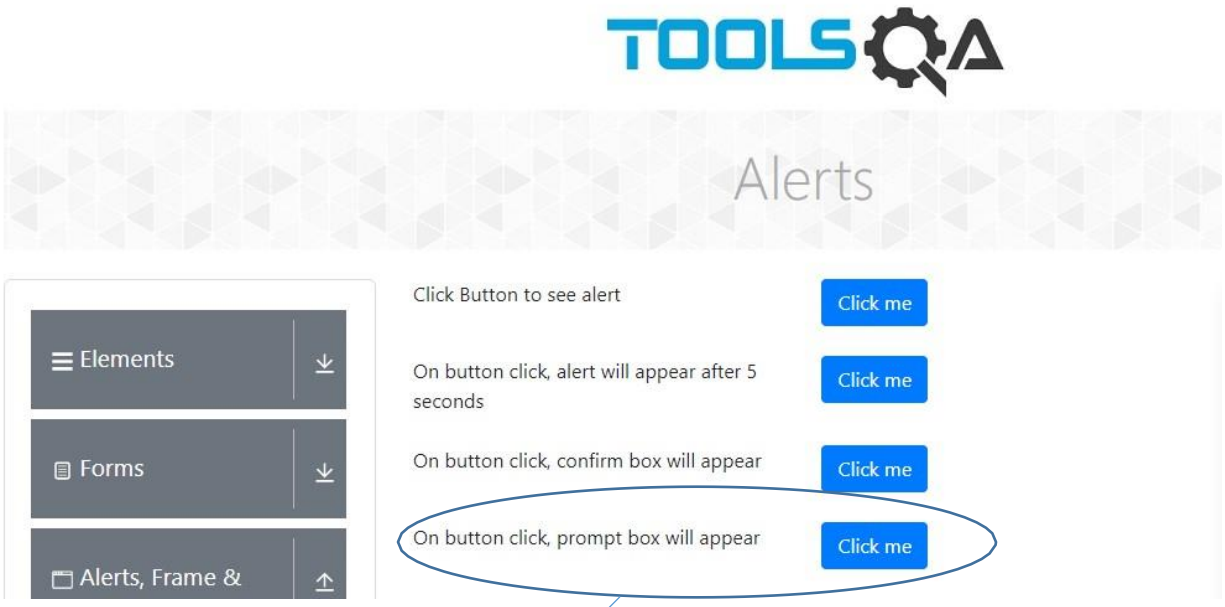
```java
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

/*Date:14/09/2021

Script for Alert Handling.Dissmiss an Alert message*/

public class AlertPopup2 {

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

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

    WebDriver driver=new ChromeDriver();

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

    driver.get("https://vctcpune.com/selenium/practice.html#");

    JavascriptExecutor je=(JavascriptExecutor) driver;
    je.executeScript("window.scrollBy(0,700)");
    Thread.sleep(4000);

    driver.findElement(By.id("confirmbtn")).click();
    Thread.sleep(4000);

    driver.switchTo().alert().dismiss();
    Thread.sleep(4000);

    driver.close();


    }

}
```



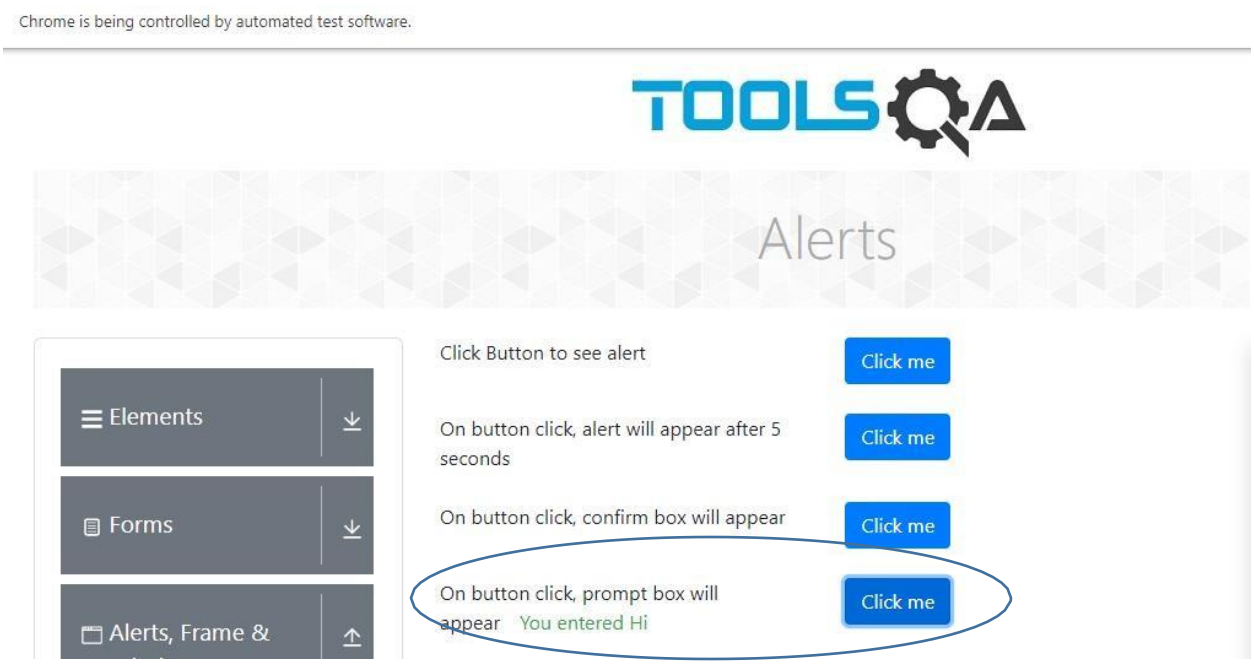
Program3:Alert handling by Entering value in text box of Alert Message.

```
package ActionClass;

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

public class AlertPopup4 {

	public static void main(String[] args) throws
InterruptedException {
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://demoqa.com/alerts");

		driver.findElement(By.id("promtButton")).click();

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

		driver.switchTo().alert().accept();
```

```
            Thread.sleep(4000);

            driver.close();


        }
}
```



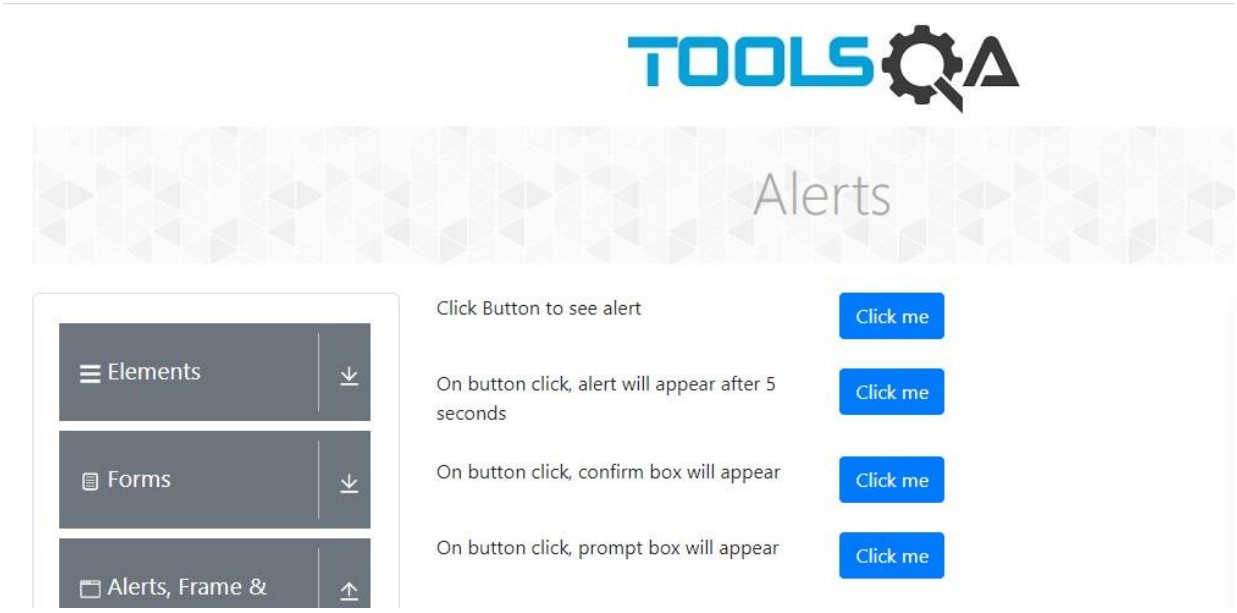
We are performing alert operations on this Alert.

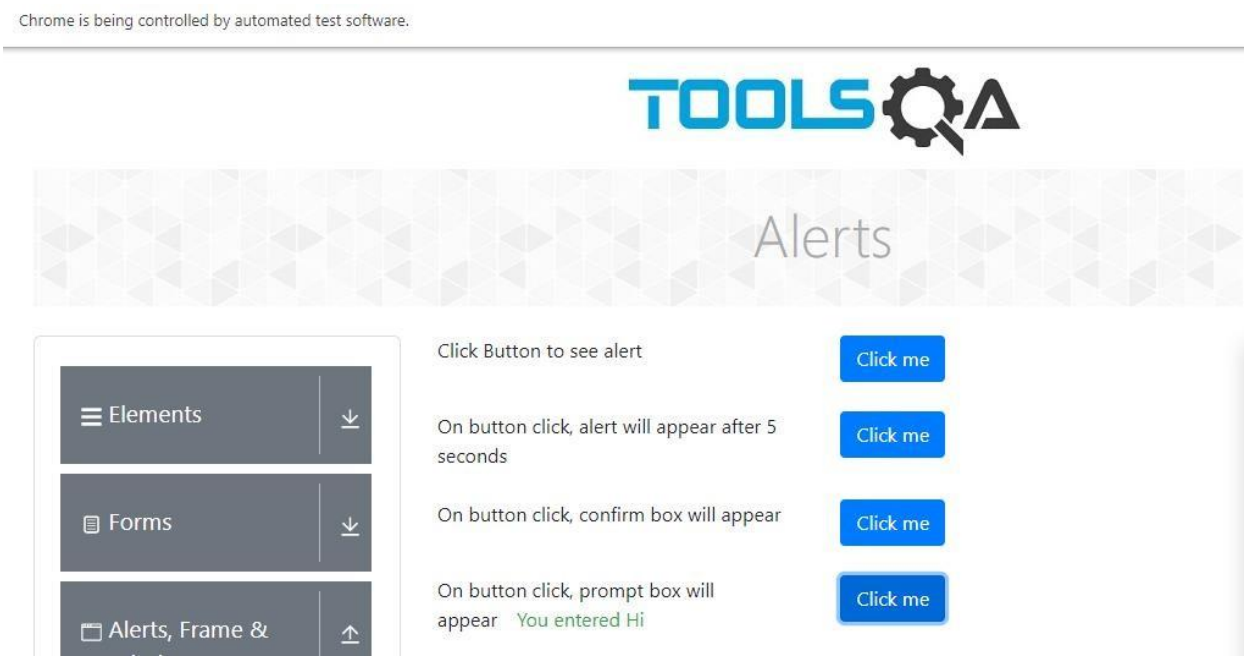


Program4:Alert handling by Entering value in text box of Alert Message by using Alert Class.

```
package ActionClass;

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

public class AlertPopup3 {

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

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

	WebDriver driver=new ChromeDriver();

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

	driver.get("https://demoqa.com/alerts");

	driver.findElement(By.id("promtButton")).click();
```

```
        Alert alert1=driver.switchTo().alert();

        alert1.sendKeys("Hi");

        alert1.accept();

        Thread.sleep(4000);
        driver.close();
         }

}
```




Chrome is being controlled by automated test software.
TOOLSQA
Alerts
Elements
Forms
Alerts, Frame &
Click Button to see alert
Click me
On button click, alert will appear after 5 seconds
Click me
On button click, confirm box will appear
Click me
On button click, prompt box will appear
You entered Hi
Click me

How to handle window or tab (Windows Handling) using Webdriver

In automation, when we have multiple windows in any web application, the activity may need to switch control among several windows from one to other in order to complete the operation. After completion of the operation, it has to return to the main window i.e. parent window in Selenium. We will see this further in the article with an example.

In Selenium web driver there are methods through which we can handle multiple windows.

Driver.getWindowHandles();

To handle all opened windows by web driver, we can use "Driver.getWindowHandles()" and then we can switch window from one window to another in a web application. Its return type is Iterator<String> i.e. **It returns set of Strings.**

Driver.getWindowHandle();

When the site opens, we need to handle the main window by **driver.getWindowHandle()**. This will handle the current window that uniquely identifies it within this driver instance. **Its return String.**

Step4:Open the web page

Step5:Use getWindowdriver() method which creates the string of window which is opened in the form of ID

Step6:Scroll down the window.

Step7:Locate element on tab button & perform click operation.Then new tab will open in browser

Step8:Use getWindowHandles() which returns set of Strings

Step9:Use for each loop

i) Print all Id's

ii) Use If condition for comparing first window ID with Second Window ID

Step10: Back to First window

Step11: Print Current Webpage URL

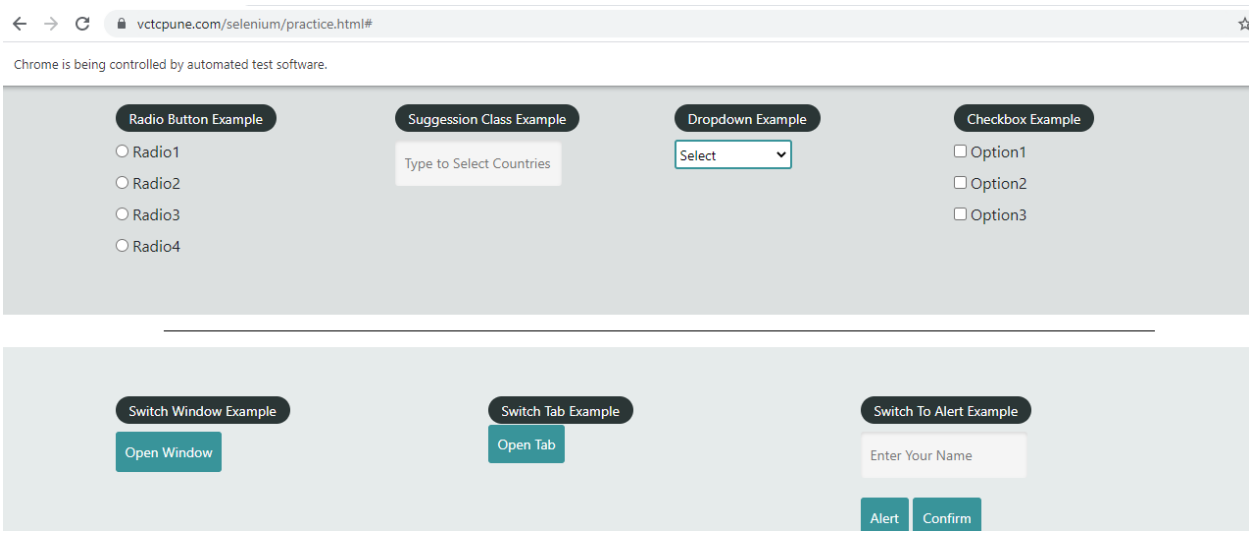
Step 12: Close the browser.

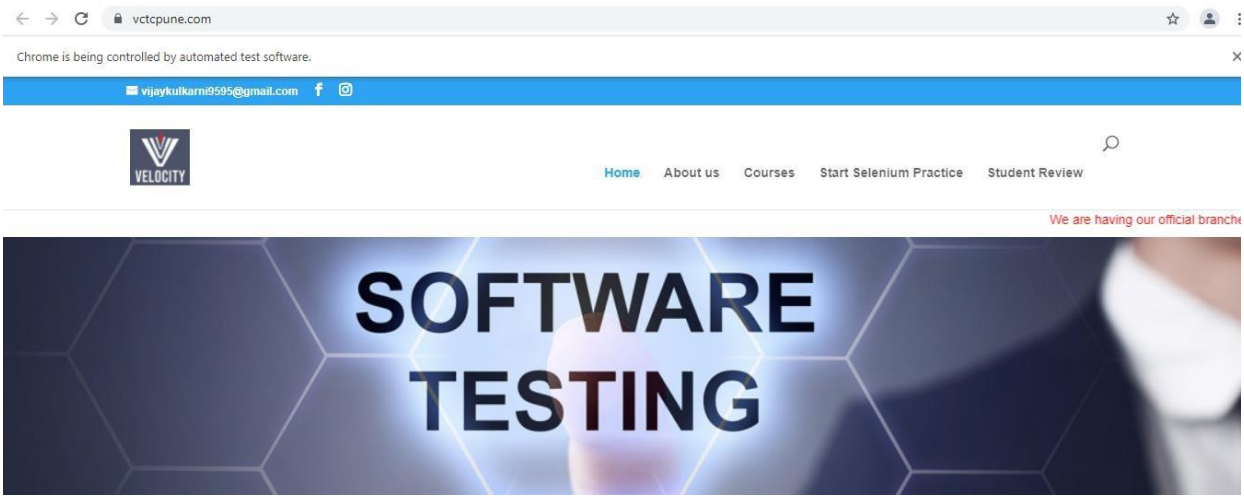
Program1: Window Handling means going from first to second tab & Coming back from second to first tab using Selenium Script

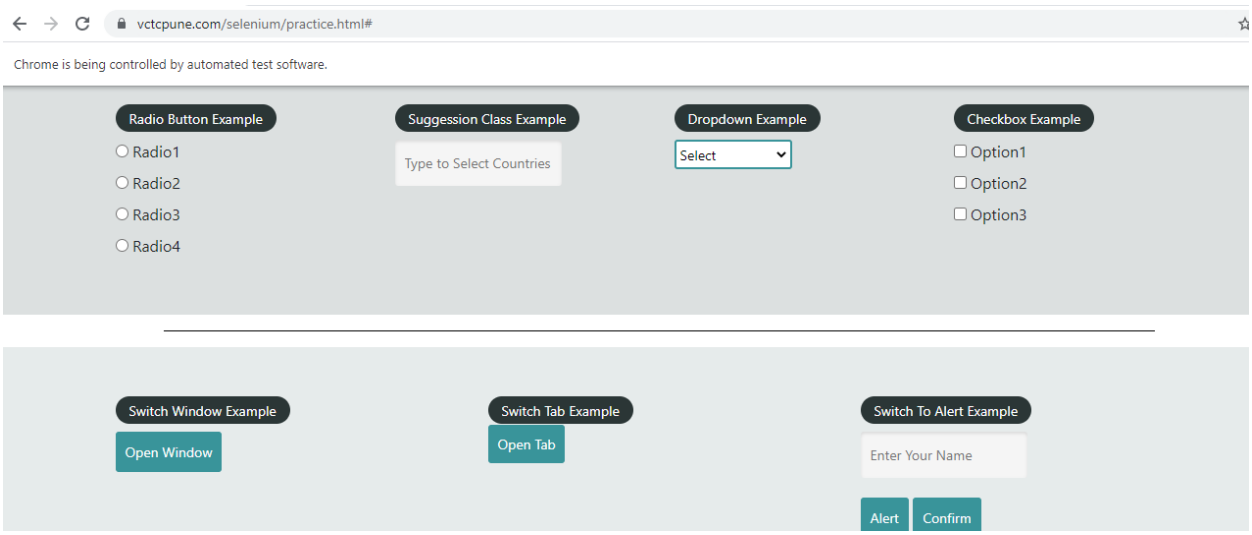
```
package ActionClass;

/*Date:14/09/2021
 Script for window or tab handling*/
import java.util.Set;

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

public class WindowHandling {

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

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://vctcpune.com/selenium/practice.html#");

		String parent=driver.getWindowHandle();
		System.out.println("Parent Id:" +parent);


		JavascriptExecutor je=(JavascriptExecutor) driver;
		je.executeScript("window.scrollBy(0,700)");
		Thread.sleep(4000);

		driver.findElement(By.id("opentab")).click();

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

		for(String id:all_tab)
```

```
          {
            System.out.println(id);
            if(!parent.equalsIgnoreCase(id))//Comparing that if current
window(parent) not equal with the Second(child) window id
            {
                  driver.switchTo().window(id);//Then Switch to child
window i.e second window which is opened in new tab
                  String title=driver.getTitle();//Getting title of
Second window
                  System.out.println("Title of the Current Page is:"
+title);
                  Thread.sleep(4000);
                  driver.close();//Close the second tab or window which
is opened

            }

          }

          driver.switchTo().window(parent);//Getting focus back to
Parent i.e. First window or tab
          String URL=driver.getCurrentUrl();
          System.out.println("Current URL is:"+URL);

          Thread.sleep(4000);
            driver.close();//Close the Parent i.e. first tab or window
which is opened

      }

}
```

```
Problems  Javadoc  Declaration  Console
<terminated> WindowHandling [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (Sep 14, 2021, 2:49:34 PM – 2:50:04 PM)
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634}) on port 48907
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1631611178.448][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 14, 2021 2:49:38 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Parent Id:CDwindow-9388A462C8C2009D641D57B7F87E9D19
CDwindow-9388A462C8C2009D641D57B7F87E9D19
CDwindow-7B436DD9E7E372964AF097129B25C7D4
Title of the Current Page is:Velocity | Pune's Best Software Testing Center
Current URL is:https://vctcpune.com/selenium/practice.html#
```


vctcpune.com/selenium/practice.html#
Chrome is being controlled by automated test software.
Radio Button Example
Radio1
Radio2
Radio3
Radio4
Suggession Class Example
Type to Select Countries
Dropdown Example
Select
Checkbox Example
Option1
Option2
Option3
Switch Window Example
Open Window
Switch Tab Example
Open Tab
Switch To Alert Example
Enter Your Name
Alert
Confirm


vctcpune.com
Chrome is being controlled by automated test software.
vijaykulkarni9595@gmail.com
VELOCITY
Home
About us
Courses
Start Selenium Practice
Student Review
We are having our official branche
SOFTWARE
TESTING


vctcpune.com/selenium/practice.html#
Chrome is being controlled by automated test software.
Radio Button Example
Radio1
Radio2
Radio3
Radio4
Suggession Class Example
Type to Select Countries
Dropdown Example
Select
Checkbox Example
Option1
Option2
Option3
Switch Window Example
Open Window
Switch Tab Example
Open Tab
Switch To Alert Example
Enter Your Name
Alert
Confirm

15) Selenium 15 Lecture

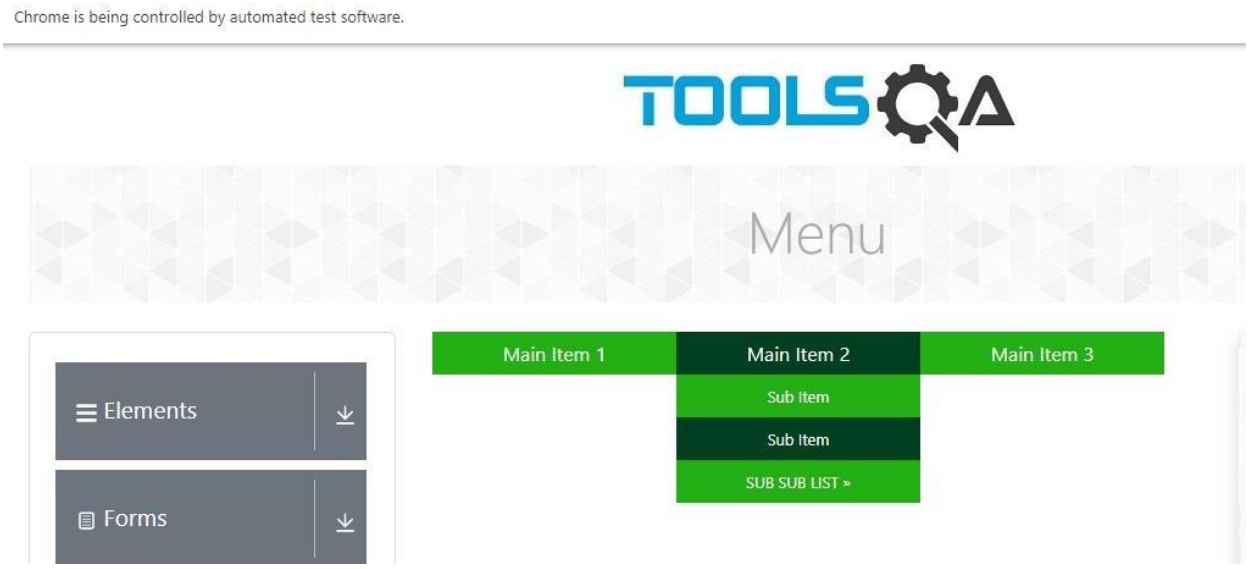
Mouse Hover Action

A mouse hover is also called as hover. Mouse hover action is basically an action where a user places a mouse over a designated area like a hyperlink. It can cause some event to get triggered.

Program1: Perform Mouse Hover operation.

```
package ActionClass;

/*Date:15/09/2021
 Program to perform Mouse Hover operation using selenium script.*/
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.interactions.Actions;

public class MouseHover {

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

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

	WebDriver driver=new ChromeDriver();

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

	driver.get("https://demoqa.com/menu/#");

	Actions ac=new Actions(driver);

	WebElement hover=driver.findElement(By.linkText("Main Item 2"));

	ac.moveToElement(hover).perform();

	WebElement Sub_part=driver.findElement(By.xpath("//*[text()='Main
Item 2']//following::a[2]"));
	Sub_part.click();

	Thread.sleep(4000);
	driver.close();
```

```
        }
}
```


Chrome is being controlled by automated test software.
TOOLSQA
Menu
Elements
Forms
Main Item 1
Main Item 2
Main Item 3
Sub Item
Sub Item
SUB SUB LIST »

16) Selenium 16 Lecture

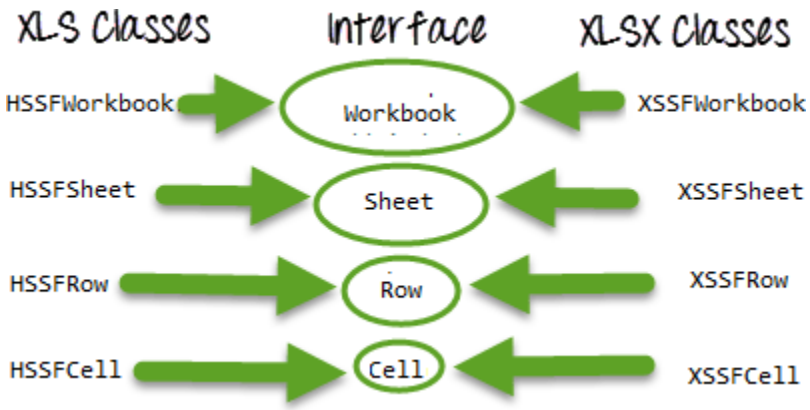
Apache POI in Selenium

The **Apache POI in Selenium** is a widely used API for selenium data driven testing. It is a POI library written in Java that gives users an API for manipulating Microsoft documents like .xls and .xlsx. Users can easily create, modify and read/write into excel files. POI stands for "Poor Obfuscation Implementation."
To Read and Write Excel file in Java, Apache provides a very famous library POI. This library is capable enough to read and write both **XLS** and **XLSX** file format of Excel.

To read **XLS** files, an **HSSF** implementation is provided by POI library.

To read **XLSX, XSSF** implementation of **POI library** will be the choice

Classes and Interfaces in POI:



Following is a list of different Java Interfaces and classes in **POI** for reading **XLS** and **XLSX** file-

- **Workbook**: XSSFWorkbook and HSSFWorkbook classes implement this interface.
- **XSSFWorkbook**: Is a class representation of XLSX file.
- **HSSFWorkbook**: Is a class representation of XLS file.
- **Sheet**: XSSFSheet and HSSFSheet classes implement this interface.
- **XSSFSheet**: Is a class representing a sheet in an XLSX file.
- **HSSFSheet**: Is a class representing a sheet in an XLS file.

- **Row**: XSSFRow and HSSFRow classes implement this interface.
- **XSSFRow**: Is a class representing a row in the sheet of XLSX file.
- **HSSFRow**: Is a class representing a row in the sheet of XLS file.
- **Cell**: XSSFCell and HSSFCell classes implement this interface.
- **XSSFCell**: Is a class representing a cell in a row of XLSX file.
- **HSSFCell:** Is a class representing a cell in a row of XLS file.

How to read/import data from Excel

Step1: Create Excel file in Eclipse workspace which is in C drive

Step2: After excel file creation. Copy the path of that Excel file by

Right click on it in Eclipse then Show in then Select System Explorer then Select file & right click then select copy as Path & Paste it into our program.

Or we can get the path by using **getProperty()** method.

Step3: We have to handle file(i.e. reading in this case) by using file class.

Step4: We have to use File Input Stram for giving input to the filoe i.e. for reading the data from that file.

A **FileInputStream** obtains input bytes from a file in a file system. What files are available depends on the host environment.

FileInputStream is meant for reading streams of raw bytes such as image data. For reading streams of characters, consider using FileReader.

Step5: Use XSSF Workbook for reading data within Excel file

Step6: Use XSSF Sheet for reading data within sheet Of Excel Workbook.

Sheet Index starts from 0.

Step7: Locate the row & column value in Excel sheet & after that Locate the String means excat data within that row & column.

Step8:In main method

i) Create the object of Web Driver

ii) Manage & Maximize the window

iii) Open the web page in browser by using get method

iv) Locate the username & insert value into it by using send keys & that value has came from Excel Sheet

v) Locate the password & insert value into it by using send keys & that value has came from Excel Sheet

vi) Close the browser

Below are the classes will use to read the data:

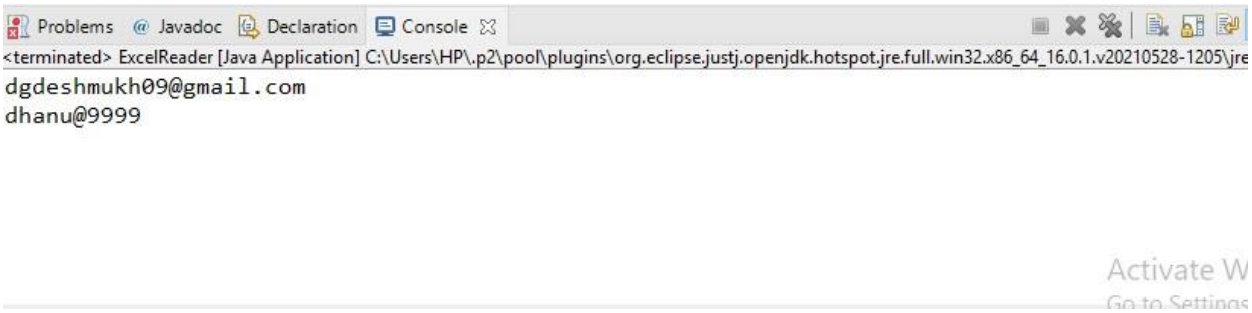
1. FileInputStream – A FileInputStream is an inputstream for reading data from a File.
2. XSSFWorkbook
3. XSSFSheet
4. XSSFRow
5. XSSFCell

Below are the methods will use to read the data which are available in the above classes:

1. getSheet("sheetName") – Get sheet with the given name
2. getLastCellNum() – Get the index of the last cell contained in the row Plus one as index starts from ZERO
3. getStringCellValue() – Get the value of the cell as a String
4. getRow(int) – Returns the row.
5. getCell(int) – Get the cell representing a given column
6. getNumericCellValue() – Get the value of the cell as a number.
7. getDateCellValue() – Get the value of the cell as a date.
8. getBooleanCellValue() – Get the value of the cell as a boolean.

Program1: Write a Script to read data from Excel file & Print it onto the console of Eclipse.

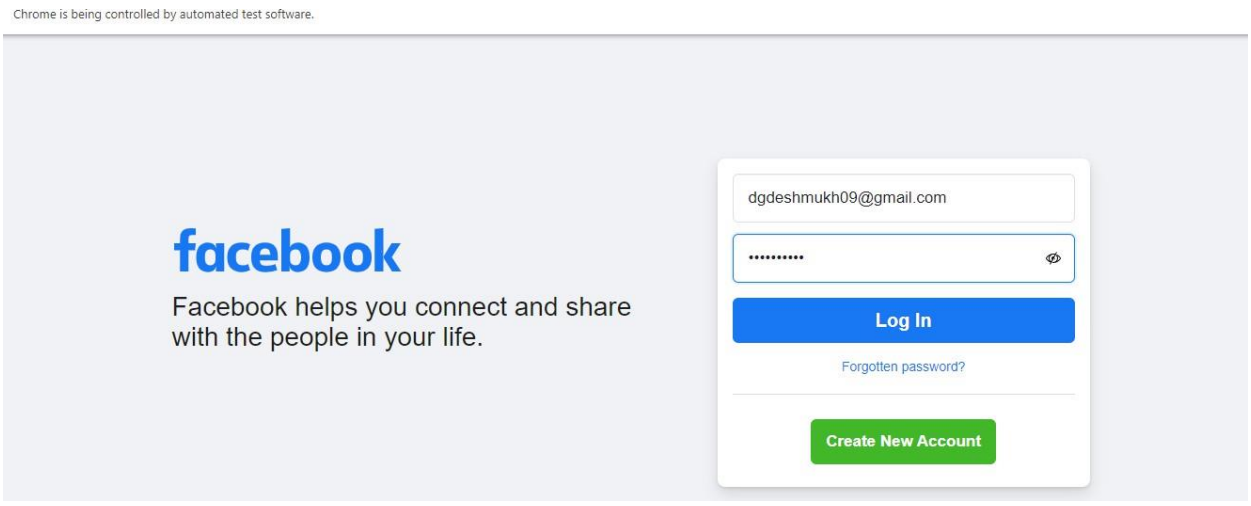
```
package ExcelOperation;

/*Date:16/09/2021
 Write a Script to read data from Excel file & Print it onto the
console of Eclipse.*/

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

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

public class ExcelReader {

	public static String readData(int row,int col) throws IOException
	{
		String Path="C:\\Users\\HP\\eclipse-
workspace\\Selenium\\Excel\\ExcelReader.xlsx";

		File file=new File(Path);

		FileInputStream fis=new FileInputStream(file);

		XSSFWorkbook wb=new XSSFWorkbook(fis);

		XSSFSheet sheet=wb.getSheetAt(0);

		String
s=sheet.getRow(row).getCell(col).getStringCellValue();
		return s;
	}

	public static void main(String[] args) throws IOException {

		System.out.println(readData(0,0));
		System.out.println(readData(0,1));
	}


}
```



Program2: Write a Script to read data from Excel file & Forward that data to Web page.

```
package ExcelOperation;

/*Date:16/09/2021
 Write a Script to read data from Excel file & Forward that data to
Web page.*/

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class ExcelReader2 {

	public static String readData(int row,int col) throws IOException
	{
		String Path="C:\\Users\\HP\\eclipse-
workspace\\Selenium\\Excel\\ExcelReader.xlsx";

		File file=new File(Path);

		FileInputStream fis=new FileInputStream(file);

		XSSFWorkbook wb=new XSSFWorkbook(fis);

		XSSFSheet sheet=wb.getSheetAt(0);
```

```
            String
s=sheet.getRow(row).getCell(col).getStringCellValue();
            return s;
      }

      public static void main(String[] args) throws IOException,
InterruptedException {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

         WebDriver driver=new ChromeDriver();

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

         driver.get("https://en-gb.facebook.com/");

         driver.findElement(By.id("email")).sendKeys(readData(0,0));

         driver.findElement(By.id("pass")).sendKeys(readData(0,1));

         Thread.sleep(4000);

         driver.close();
      }


}
```



17) Selenium 17 Lecture

Write Data into Excel Sheet using Selenium Webdriver

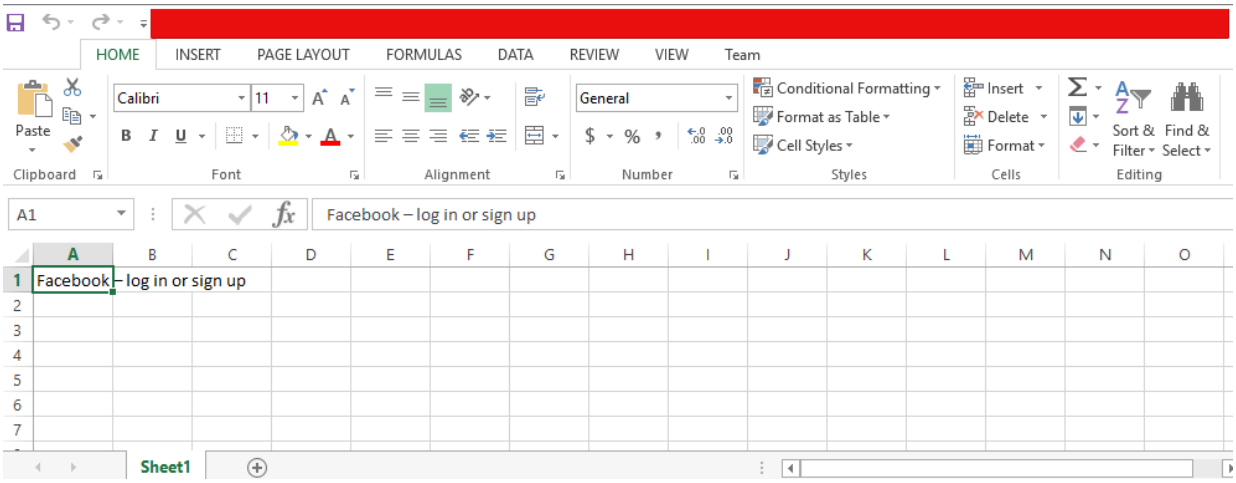
Step 1: Specify the file path where you wish the Excel Sheet to show.
Step 2: Use File class
Step 3: Use File Input Stream class
Step 4: Time to create a Workbook now.
Step 5: Time to create a sheet. Type the following for creating a sheet:
Step 6: Use getRow(int rownum), createCell(int column index), setCellValue(String): void
Step 7: Use FileOutputStram class. Give path of Excel file in which we have to write data.
Step8: We have to write data into workbook so use
Wb.write() method where Wb is reference variable of WorkBook.

Program1: Write a data which is coming from web page(i.e. Title of web page)to Blank Excel file .

```
package ExcelOperation;


import java.io.File;


import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;


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

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

import org.openqa.selenium.WebDriver;

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

```
/*Date:17/09/2021

 Program to write a data to blank Excel file from the  data which is coming from Web page(i.e.title of web page)

 using Selenium Script*/


public class ExcelWriter {


        public static void writeData(int row,int col, String Value) throws IOException

        {

      //String  Path="C:\\Users\\HP\\eclipse-workspace\\Selenium\\Excel\\ExcelWriter.xlsx";


      String Path=".\\Excel\\ExcelWriter.xlsx";


                  File file=new File(Path);


                  FileInputStream fis=new FileInputStream(file);


                  XSSFWorkbook wb=new XSSFWorkbook(fis);


                  XSSFSheet sheet=wb.getSheet("Sheet1");
```

```
                sheet.createRow(row).createCell(col).setCellValue(Value);


                FileOutputStream fs=new FileOutputStream(Path);


                wb.write(fs);
        }


        public static void main(String[] args) throws InterruptedException, IOException {


System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay
Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");


       WebDriver driver=new ChromeDriver();
     driver.manage().window().maximize();
     driver.get("https://en-gb.facebook.com/");
     String title=driver.getTitle();
     System.out.println(title);
     writeData(0,0,title);
    Thread.sleep(4000);
   driver.close();
    }
}
```

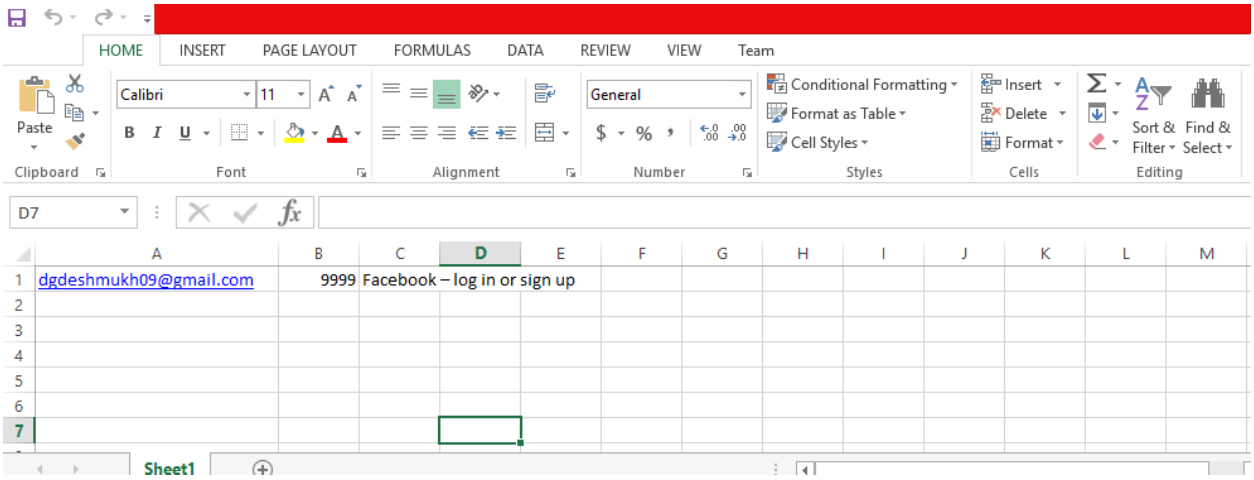

HOME
INSERT
PAGE LAYOUT
FORMULAS
DATA
REVIEW
VIEW
Team
Paste
Calibri
11
General
Conditional Formatting
Format as Table
Cell Styles
Insert
Delete
Format
Sort & Filter
Find & Select
Clipboard
Font
Alignment
Number
Styles
Cells
Editing
A1
Facebook – log in or sign up
Facebook – log in or sign up
Sheet1

Program2:Script to read a data from Excel file(i.e. Password which is numeric value) & String value(Username) from Program & Insert it onto web page

& Write a data(Title of web page) to same Excel sheet which is coming from web page through Selenium Script.

```
package ExcelOperation;

import java.io.File;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

/*Date:17/09/2021
 Program to read a data from Excel file by passing String value through program(username)
 & write a data to Excel file from the  data which is coming from Web page(i.e.title of web page)
 using Selenium Script*/

public class ExcelReaderWriter {

	static String i2;

	public static int readData(int row,int col) throws IOException
	{
		String Path="C:\\Users\\HP\\eclipse-workspace\\Selenium\\Excel\\ExcelReader2.xlsx";
		File file=new File(Path);
		FileInputStream fis=new FileInputStream(file);
		XSSFWorkbook wb=new XSSFWorkbook(fis);
		XSSFSheet sheet=wb.getSheetAt(0);
		Int i=(int)sheet.getRow(row).getCell(col).getNumericCellValue();
		i2=String.valueOf(i);
		return i;

	}
```

```
	public static void writeData(int row,int col,String i) throws
IOException
	{
	   String Path="C:\\Users\\HP\\eclipse-
workspace\\Selenium\\Excel\\ExcelReader2.xlsx";
		File file=new File(Path);
		FileInputStream fis=new FileInputStream(file);
		XSSFWorkbook wb=new XSSFWorkbook(fis);
		XSSFSheet sheet=wb.getSheetAt(0);
		sheet.getRow(row).createCell(col).setCellValue(i);
		FileOutputStream fos=new FileOutputStream(Path);
		wb.write(fos);

	}

public static void main(String[] args) throws IOException,
InterruptedException {

		System.out.println(readData(0,1));
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
	    WebDriver driver=new ChromeDriver();
	    driver.manage().window().maximize();
	    driver.get("https://en-gb.facebook.com/");
	    String title=driver.getTitle();
	    writeData(0,2,title);
	    Thread.sleep(4000);

driver.findElement(By.id("email")).sendKeys("dgdeshmukh09@gmail.com");
	    driver.findElement(By.id("pass")).sendKeys(i2);
	    Thread.sleep(4000);
	    driver.close();
	}

}
```


HOME
INSERT
PAGE LAYOUT
FORMULAS
DATA
REVIEW
VIEW
Team
Paste
Calibri
11
General
Conditional Formatting
Format as Table
Cell Styles
Insert
Delete
Format
Sort & Filter
Find & Select
Clipboard
Font
Alignment
Number
Styles
Cells
Editing
D7
dgdeshmukh09@gmail.com
9999
Facebook – log in or sign up
Sheet1

18) Selenium 18 Lecture

TestNG

Test-Ng-Next Generation-

It is framework designed to simplify broad range of testing needs, from unit testing to Integration testing.

It is unique framework which is use to design test classes.

It supports different annotation and generate script in test format.

It also provide the test case report to use.

It support the data driven framework.

What is TestNG?

TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired from JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy.
Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

For example:

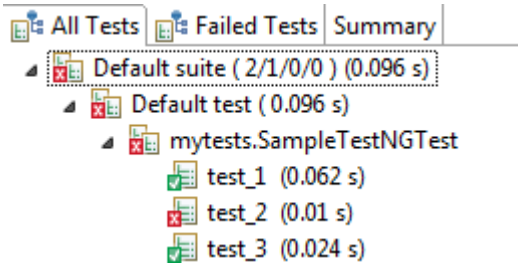
- Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method is failed. Then correct the errors present in the fourth method, now you want to run only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG.
- The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

Why Use TestNG with Selenium?

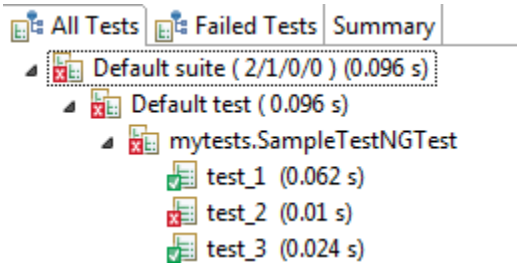
Default Selenium tests do not generate a proper format for the test results. Using TestNG in Selenium, we can generate test results.

Most Selenium users use this more than Junit because of its advantages. There are so many features of TestNG, but we will only focus on the most important ones that we can use in Selenium. Following are the key features of Selenium TestNG:

- Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
- Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
- The same test case can be executed multiple times without loops just by using keyword called 'invocation count.'
- Using testng, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.
- The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
- Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
- WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.



- WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.

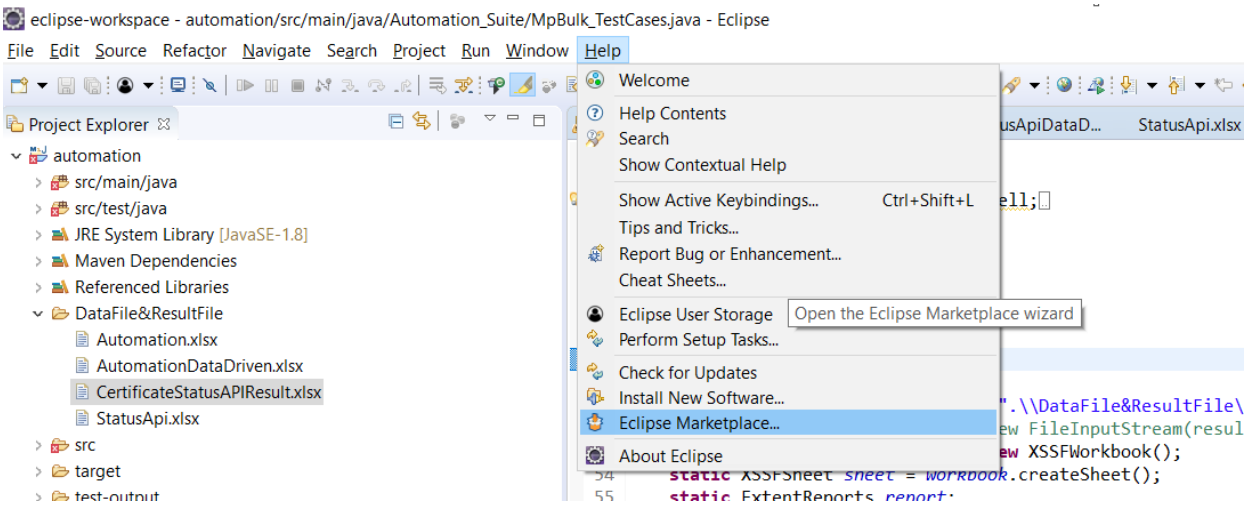


- TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static.

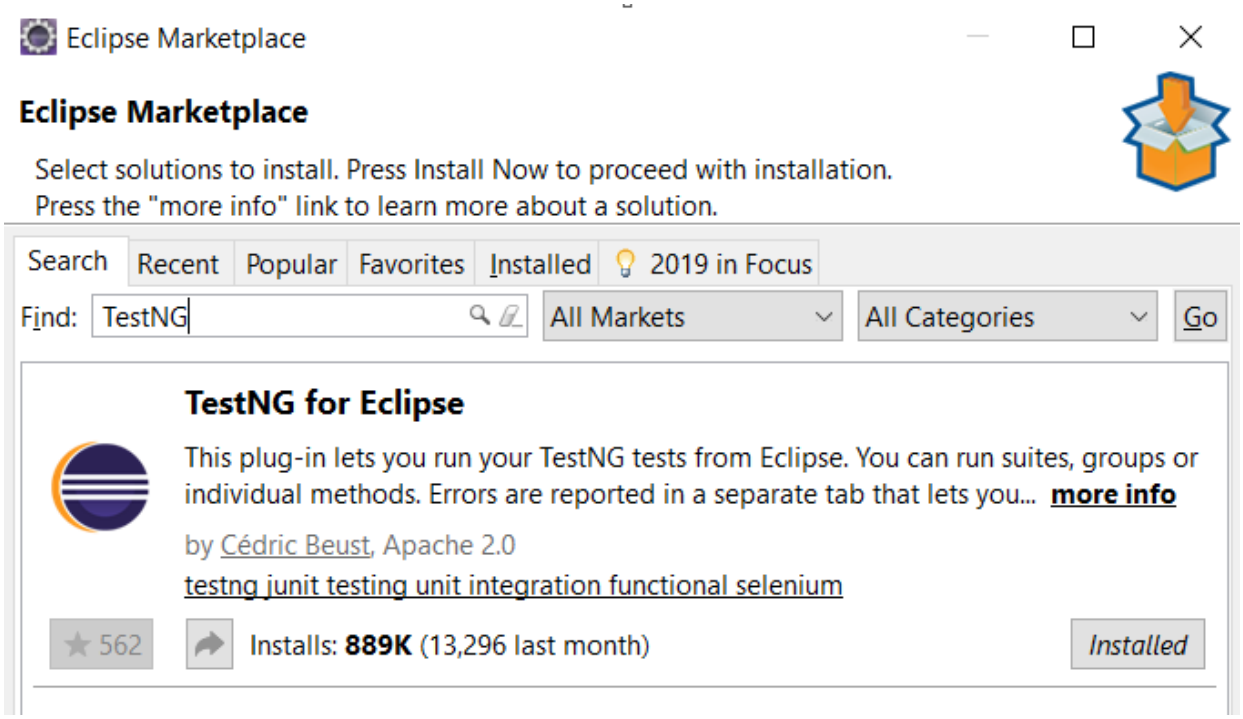
TestNG: Installation and Setup

1. Install Eclipse IDE from the [Eclipse website](). It serves as a platform for you to code on and write your test cases

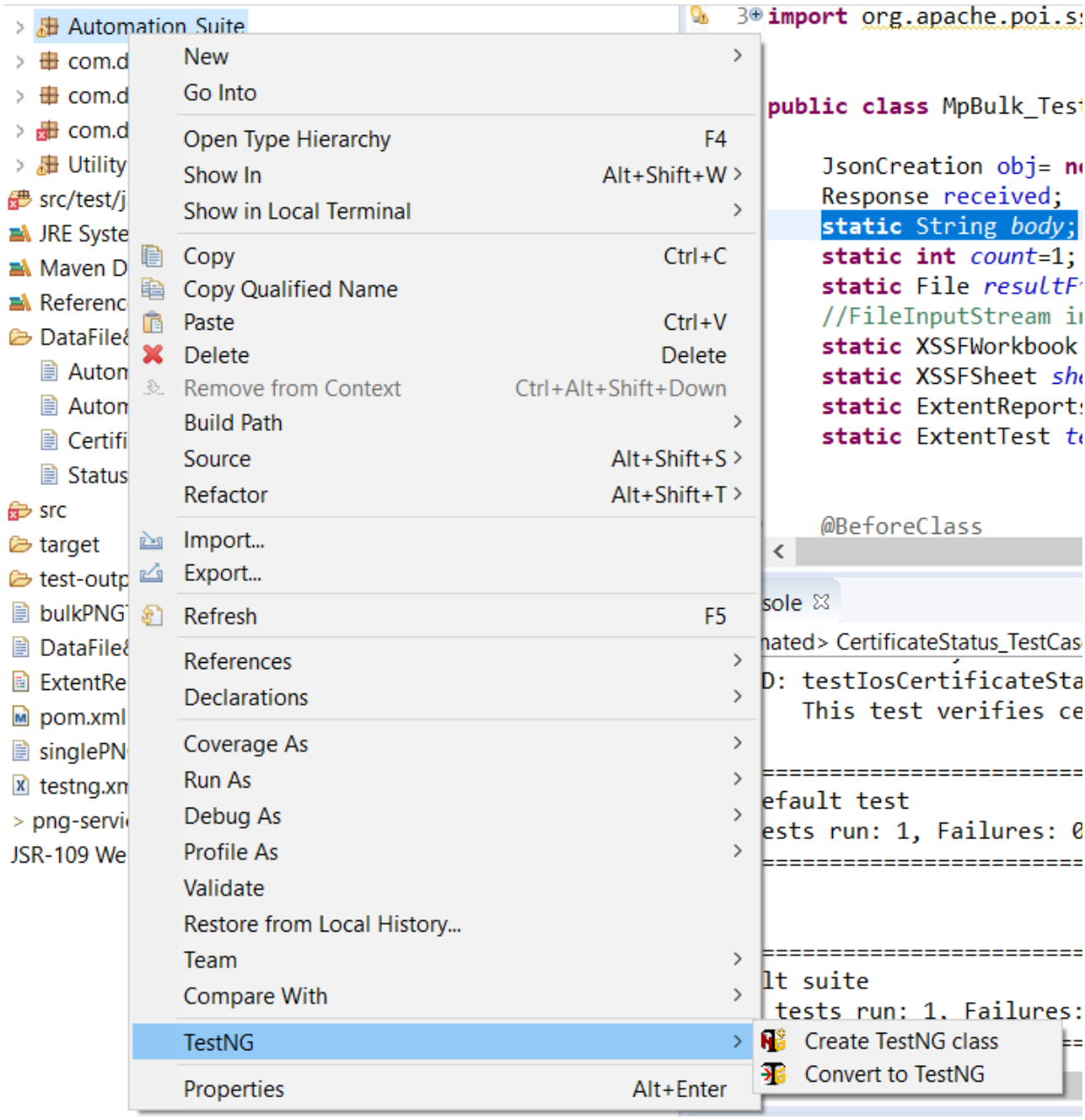
2. Once installed, go to help and navigate to the 'Eclipse Marketplace'. The referenced snapshot is below:



3. Click on ‘Eclipse Marketplace’. You will be directed to the marketplace modal. Type TestNG in the search keyword and hit ‘Go’. The referenced snapshot is below:

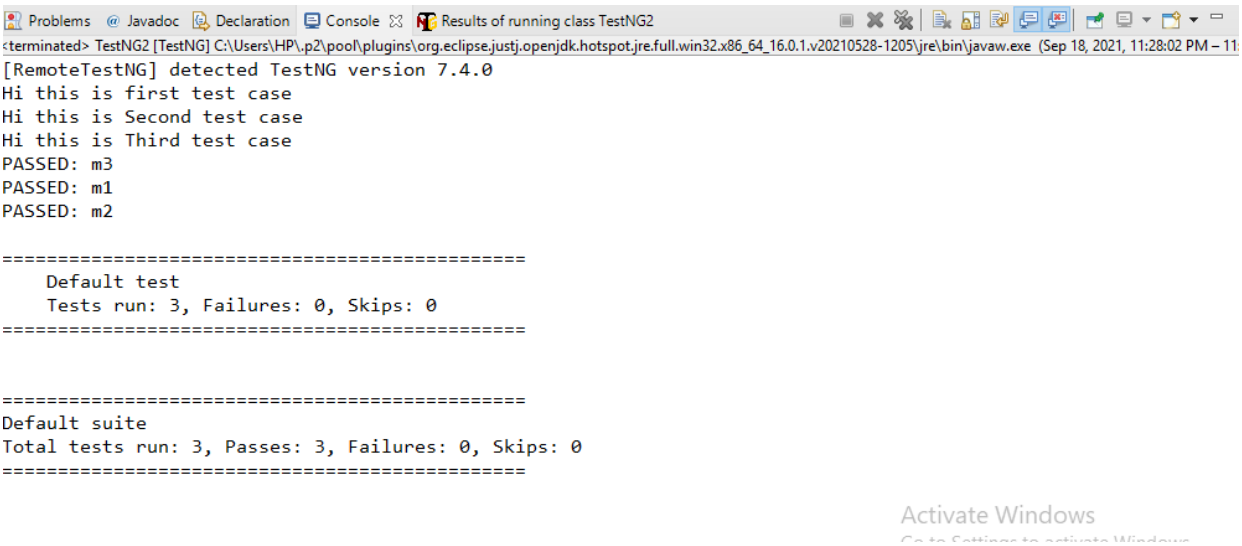


4. Post-restarting your Eclipse, re-verify whether you can see options for creating a TestNG class or not as below:



Program1: First Program of TestNG.

```
package TestNGBasic;

/*Date:18/09/2021*/

import org.testng.annotations.Test;

public class TestNG2 {

	@Test
	public static void m1() {
		System.out.println("Hi this is first test case");
	}

	@Test
	public static void m2() {
		System.out.println("Hi this is Second test case");

	}

	@Test
	public static void m3() {
		System.out.println("Hi this is Third test case");
	}


}
```



19) Selenium 19 Lecture

What is TestNG Annotation?

TestNG Annotation is a piece of code which is inserted inside a program or business logic used to control the flow of execution of test methods. Annotations are pre condition & post conditions.

TestNG Annotation	Description
@BeforeSuite	The @BeforeSuite annotated method will run before the execution of all the test methods in the suite.
@AfterSuite	The @AfterSuite annotated method will run after the execution of all the test methods in the suite.
@BeforeTest	The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder.
@AfterTest	The @AfterTest annotated method will be executed after the execution of all the test methods of available classes belonging to that folder.
@BeforeClass	The @BeforeClass annotated method will be executed before the first method of the current class is invoked.
@AfterClass	The @AfterClass annotated method will be invoked after the execution of all the test methods of the current class.
@BeforeMethod	The @BeforeMethod annotated method will be executed before each test method will run.
@AfterMethod	The @AfterMethod annotated method will run after the execution of each test method.
@BeforeGroups	The @BeforeGroups annotated method run only once for a group before the execution of all test cases belonging to that group.
@AfterGroups	The @AfterGroups annotated method run only once for a group after the execution of all test cases belonging to that group.

Hierarchy of the TestNG Annotations:

- @BeforeSuite
- @BeforeTest
- @BeforeClass
- @BeforeMethod
- @Test
- @AfterMethod
- @AfterClass
- @AfterTest
- @AfterSuite

What Is TestNG.Xml?

TestNG.xml file is a configuration file that helps in organizing our tests. It allows testers to create and handle multiple test classes, define test suites and tests.

It makes a tester's job easier by controlling the execution of tests by putting all the test cases together and run it under one XML file. This is a beautiful concept, without which, it is difficult to work in TestNG.

Advantages Of TestNG.xml

Major advantages of TestNG.xml file are:

- It provides parallel execution of test methods.
- It allows the dependency of one test method on another test method.
- It helps in prioritizing our test methods.
- It allows grouping of test methods into test groups.
- It supports the parameterization of test cases using @Parameters annotation.
- It helps in Data-driven testing using @DataProvider annotation.
- It has different types of assertions that help in validating the expected results with the actual results.
- It has different types of HTML reports, Extent reports, etc. for a better and clear understanding of our test summary.
- It has listeners who help in creating logs.

Execution order of Annotation-

1) @BeforeSuite

2) @BeforeTest

3) @BeforeClass

4) @BeforeMethod

5) @Test

6) @AfterMethod

7) @AfterClass

8) @AfterTest

9) @AfterSuite

1) How to create testNg classes.

Right Click on package

Go to testNG Option

Click on Create TestNg Class

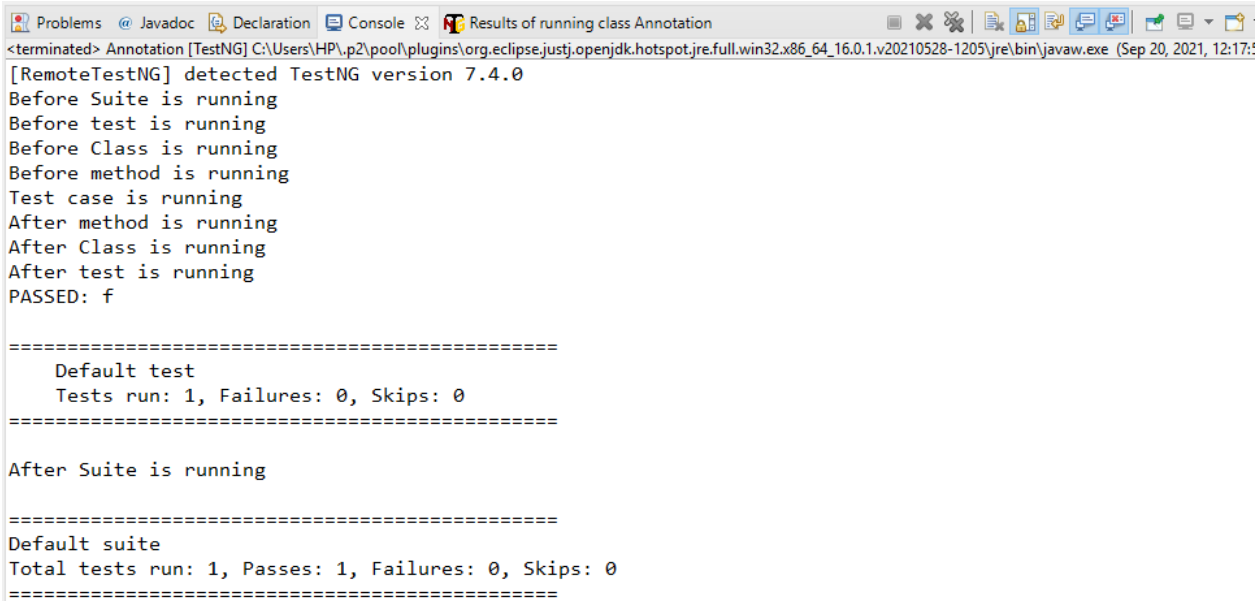
Enter the Class Name

Select annotations Required.

```
package TestNGBasic;

/*Date:19/09/2021
 Program to use different Annotations*/

import org.testng.annotations.Test;

import org.testng.annotations.BeforeMethod;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
```

```java
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.AfterSuite;

public class Annotation {
  @Test
  public void f() {
        System.out.println("Test case is running");
  }
  @BeforeMethod
  public void beforeMethod() {
        System.out.println("Before method is running");
  }

  @AfterMethod
  public void afterMethod() {
        System.out.println("After method is running");
  }

  @BeforeClass
  public void beforeClass() {
        System.out.println("Before Class is running");
  }

  @AfterClass
  public void afterClass() {
        System.out.println("After Class is running");
  }

  @BeforeTest
  public void beforeTest() {
        System.out.println("Before test is running");
  }

  @AfterTest
  public void afterTest() {
        System.out.println("After test is running");
  }

  @BeforeSuite
  public void beforeSuite() {
        System.out.println("Before Suite is running");
  }

  @AfterSuite
  public void afterSuite() {
        System.out.println("After Suite is running");
```

```
  }

}
```



When we have more than one classes in that situation we are to execute only test suite & all classes within Test suite will automatically going to be execute.

2) How to create testNg suite(.xml file)

Right Click on project or Package

Got to testNg option

Select Convert to TestNg

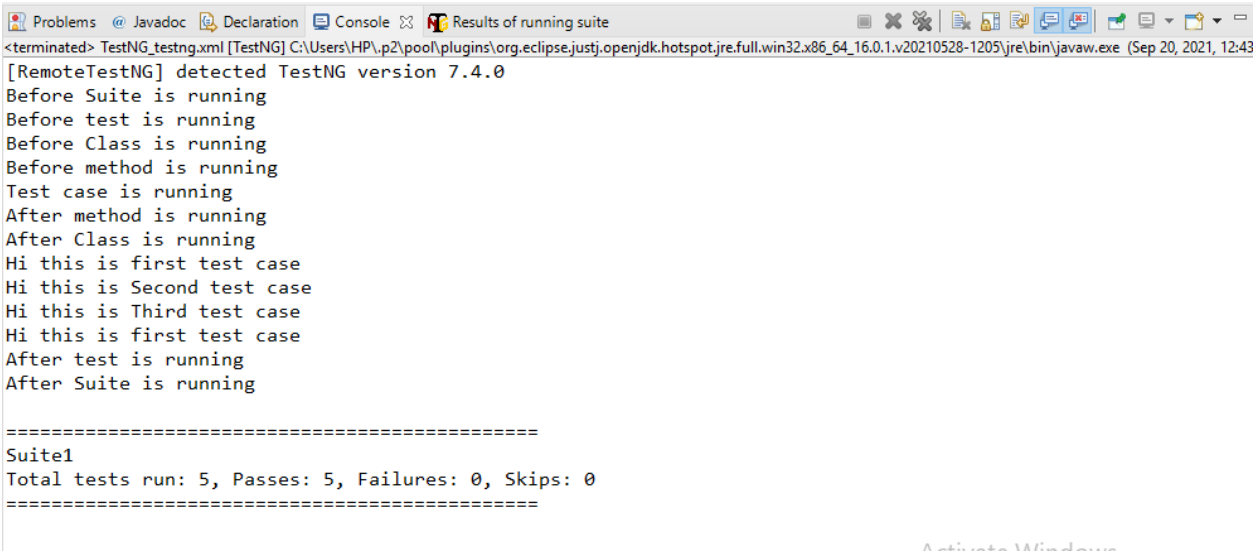
Enter the suite Name require

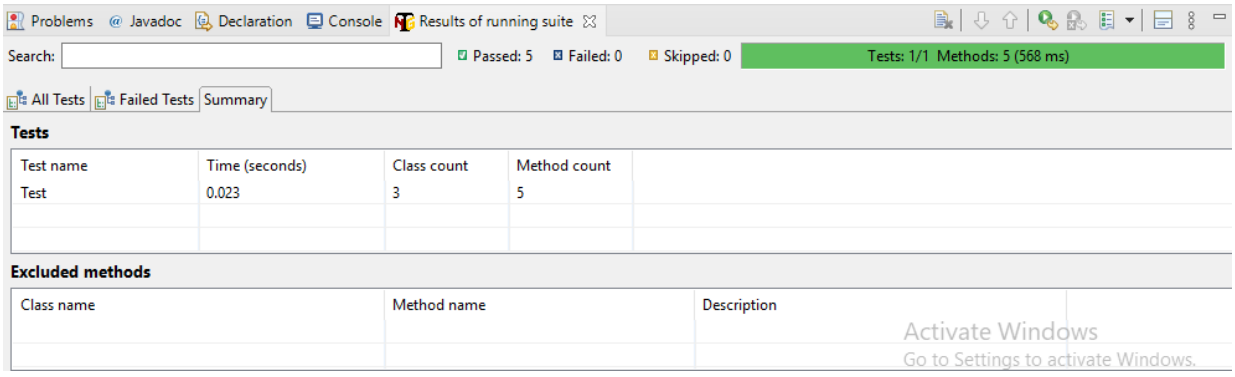
Enter the test name require

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite1">
  <test thread-count="5" name="Test">
    <classes>
```

```
      <class name="TestNGBasic.Annotation"/>
      <class name="TestNGBasic.TestNG2"/>
      <class name="TestNGBasic.TestNGFirstProgram"/>
    </classes>
  </test> <!-- Test -->

</suite> <!-- Suite1 -->
```

Default Thread count in Test suite xml file is 5

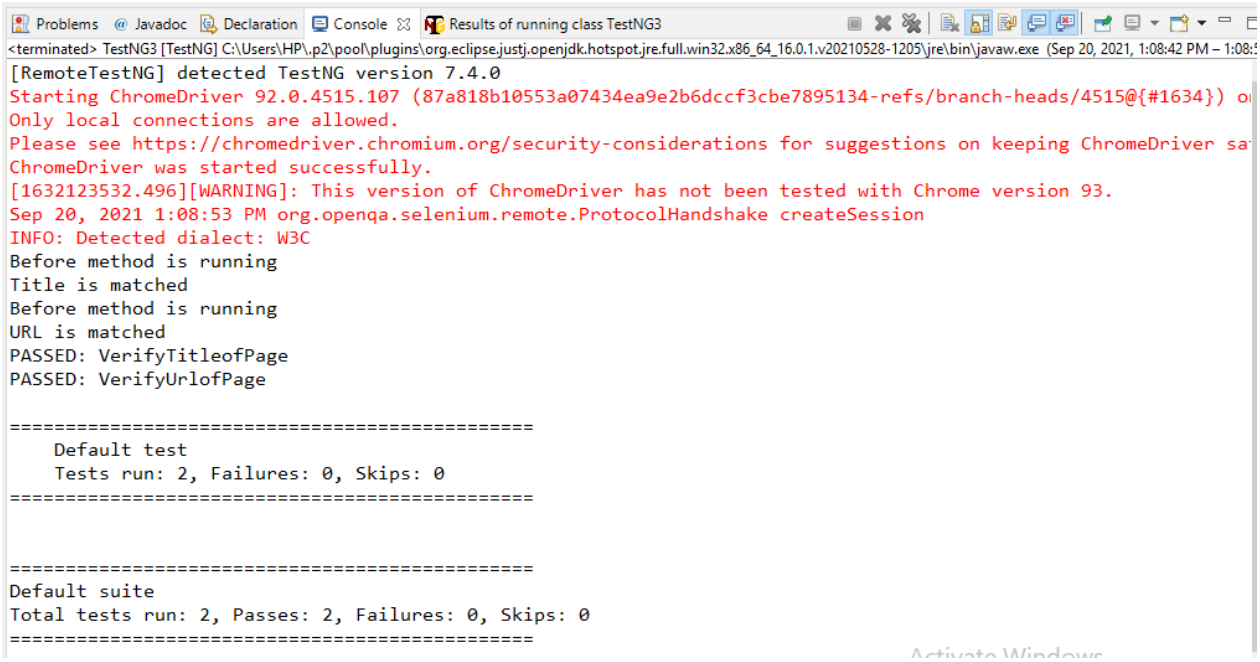


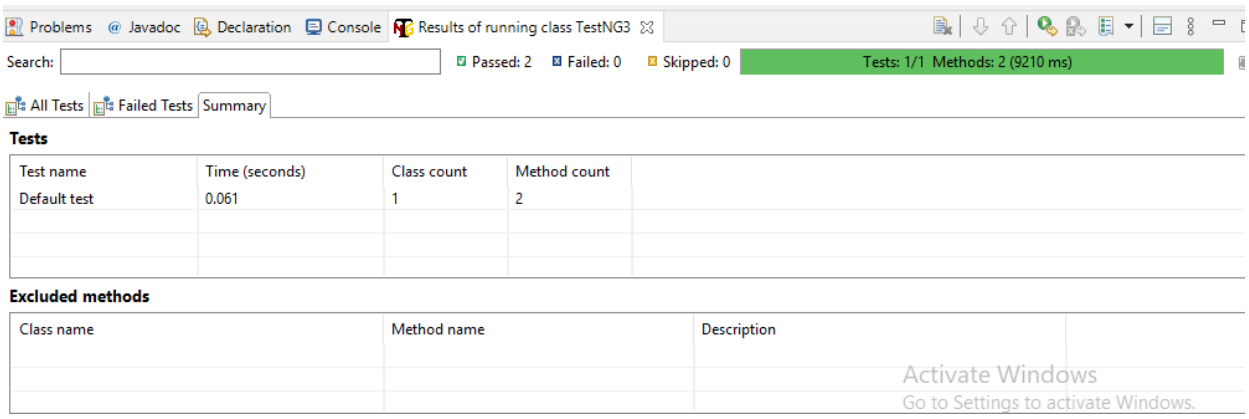


Program: Verify Title & URL of web page using TestNG class.

```
package TestNGBasic;

/*Date:19/09/2021
 Program to Verify title & url of web page using TestNG class.*/

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.Test;

public class TestNG3 {

	WebDriver driver;

	@BeforeSuite
	public void initBrowser()
	{
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

	  driver=new ChromeDriver();
	  driver.manage().window().maximize();
	}

	@BeforeClass
	public void navigatepage()
	{
		driver.get("https://www.google.com/");
	}

	@BeforeMethod
	public void beforemethod()
	{
		System.out.println("Before method is running");
	}

  @Test
  public void VerifyTitleofPage() {
		String title=driver.getTitle();

		if(title.equalsIgnoreCase("Google"))
		{
```

```
                System.out.println("Title is matched");
        }
        else
        {
                System.out.println("Title is not matched");
        }
    }

    @Test
    public void VerifyUrlofPage()
    {
        String URL=driver.getCurrentUrl();
        if(URL.equalsIgnoreCase("https://www.google.com/"))
        {
                System.out.println("URL is matched");
        }
        else
        {
                System.out.println("URL is not matched");
        }

    }

}
```





Interview Questions:

1) How to create testNg classes.
2) How to create testNg suite(.xml file)
3) What are the different testNg annotation
4) How to write the testNg annotation
5) What is sequence for different annotations
6) How to write sample test script.

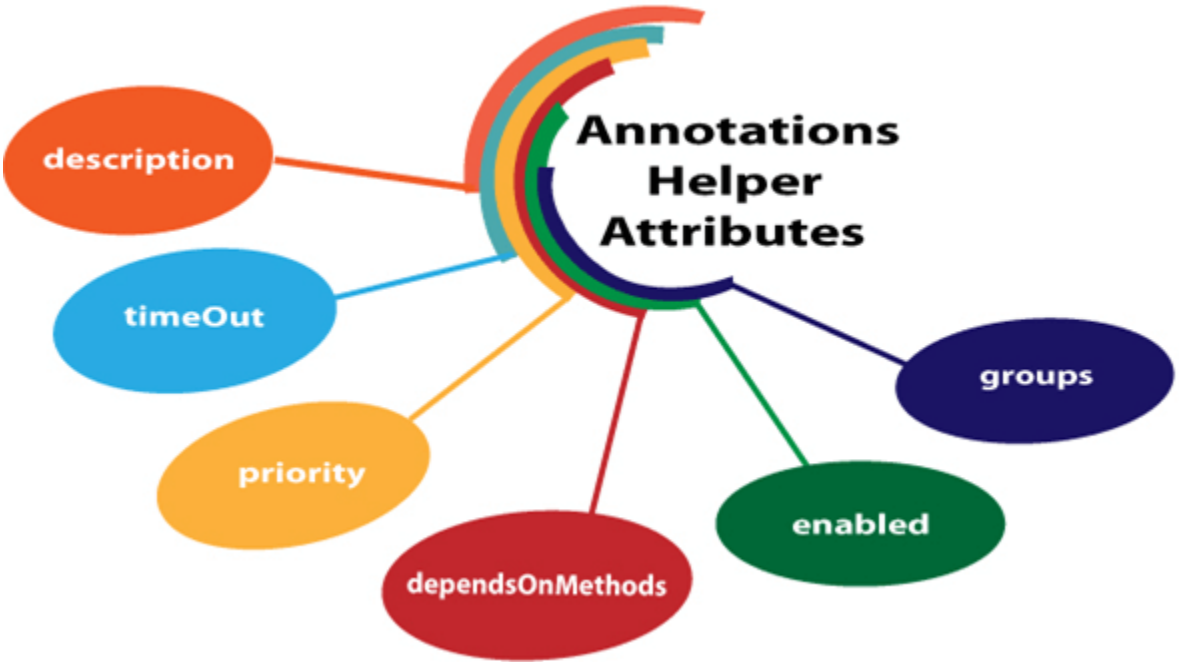
20) Selenium 20 Lecture

If multiple tests are executed then they execute in alphabetical order of their names.

```
package TestNGBasic;

import org.testng.annotations.Test;

/*Date:20/09/2021
 Program for Annotation Multiple tests*/

public class AnnotationMultipleTests {

	@Test
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test
	public void pineapple()
	{
		System.out.println("Fruit is Pineapple");
	}

	@Test
	public void custardapple()
	{
		System.out.println("Fruit is Custardapple");
	}

}
```

```
Problems  Javadoc  Declaration  Console  Results of running class AnnotationAttribute1
<terminated> AnnotationAttribute1 [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (Sep 20, 2021, 1:4
[RemoteTestNG] detected TestNG version 7.4.0
Fruit is Apple
Fruit is Custardapple
Fruit is Pineapple
PASSED: custardapple
PASSED: pineapple
PASSED: apple

===============================================
    Default test
    Tests run: 3, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 3, Passes: 3, Failures: 0, Skips: 0
===============================================
```

TestNG Annotation Attributes



1) Priority:

We can provide positive as well as negative no but precedence will be negative to positive.

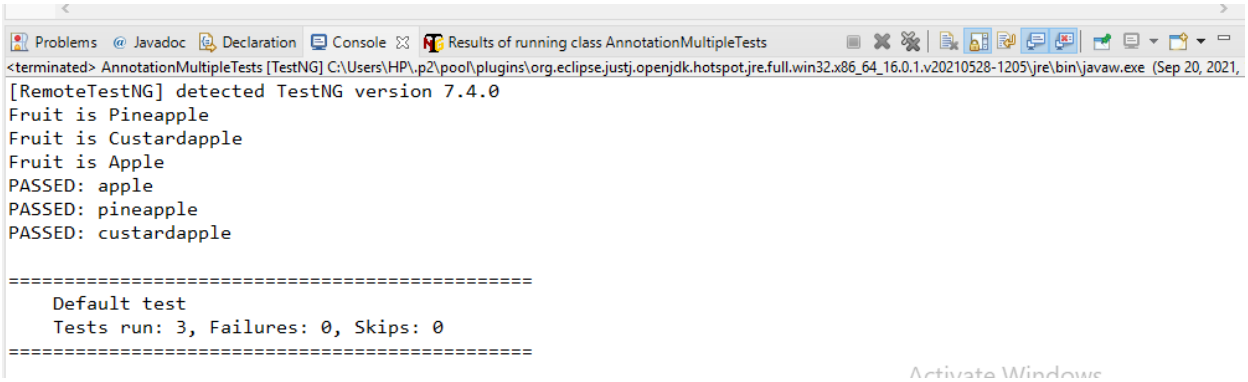
When no 'priority' attribute is specified then the TestNG will run the test cases in alphabetical order.

Priority determines the sequence of the execution of the test cases. The priority can hold the integer values between -5000 and 5000.

When the priority is set, the lowest priority test case will run first and the highest priority test case will be executed last. Suppose we have three test cases and their priority values are -5000, 0, 15, then the order of the execution will be 0,15,5000.

If priority is not specified, then the default priority will be 0.

```java
package TestNGBasic;

import org.testng.annotations.Test;

/*Date:20/09/2021
 Program for Annotation Multiple tests*/

public class AnnotationMultipleTests {

	@Test(priority=2)
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test(priority=-1)
	public void pineapple()
	{
		System.out.println("Fruit is Pineapple");
	}

	@Test(priority=0)
	public void custardapple()
	{
		System.out.println("Fruit is Custardapple");
	}

}
```

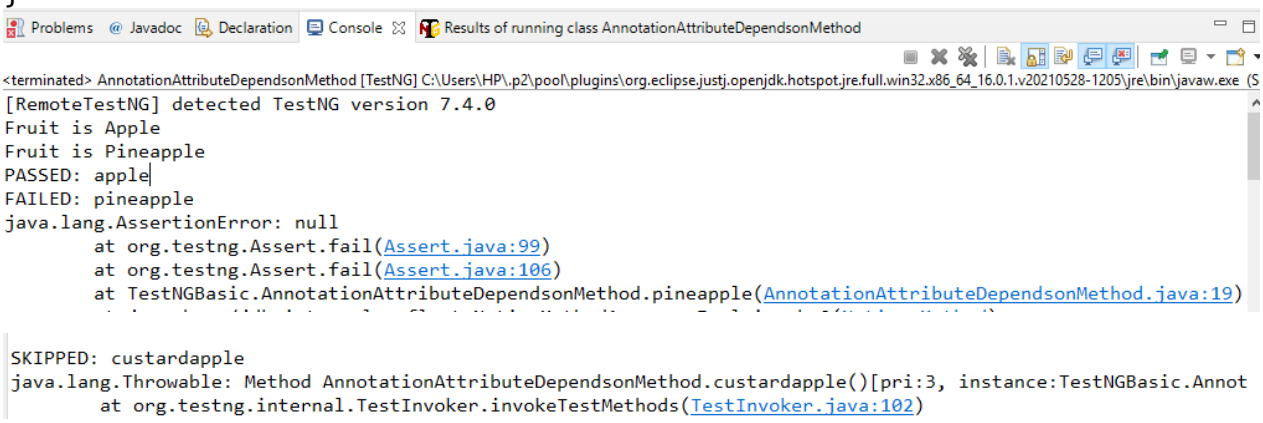


2) DependsonMethods:

This attribute genrate the relation between the test cases. This provide compulsion for depends methods to be excuted only when its first will pass.

If First method gets fail dependednt method will be skip.

```
package TestNGBasic;

import static org.testng.Assert.fail;

import org.testng.Assert;
import org.testng.annotations.Test;

public class AnnotationAttributeDependsonMethod {
	@Test(priority=1)
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test(priority=2)
	public void pineapple()
	{
		System.out.println("Fruit is Pineapple");
		Assert.fail();
	}

	@Test(dependsOnMethods="pineapple",priority=3 )
	public void custardapple()
	{
		System.out.println("Fruit is Custardapple");
	}

}
```



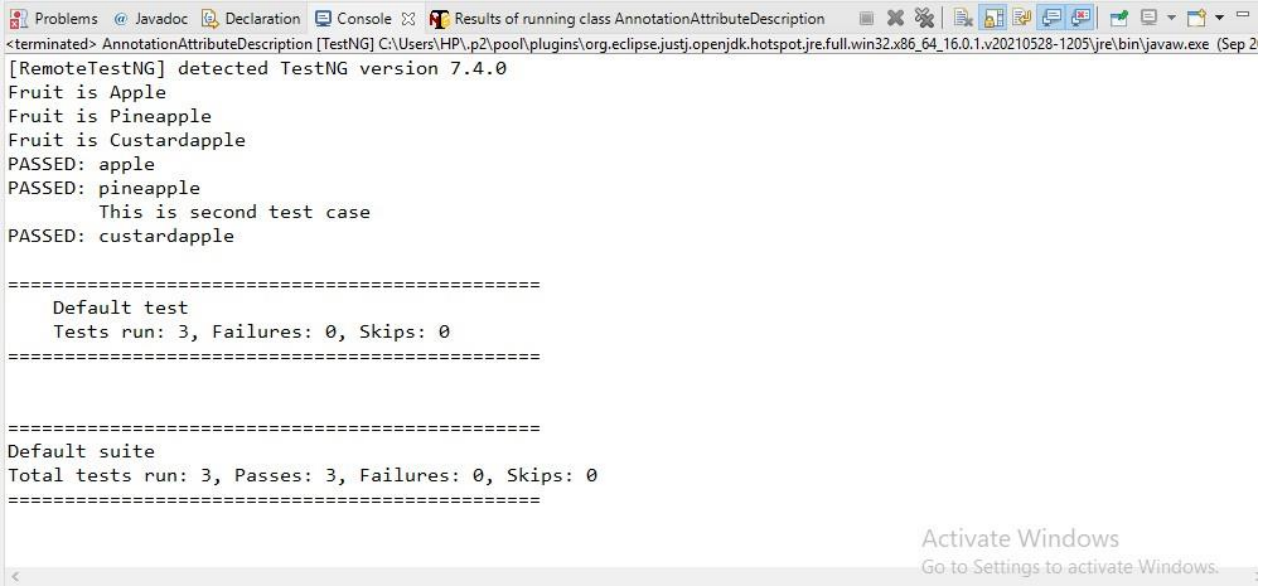
```
===============================================
    Default test
    Tests run: 3, Failures: 1, Skips: 1
===============================================


===============================================
Default suite
Total tests run: 3, Passes: 1, Failures: 1, Skips: 1
===============================================
```


Assert.fail() is used to delibarately i.e. Forcefully fail the teat case

3) Description

Using this we can add description of the Test Case

```
package TestNGBasic;

import org.testng.Assert;
import org.testng.annotations.Test;

public class AnnotationAttributeDescription {

	@Test(priority=1)
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test(priority=2,description="This is second test case")
	public void pineapple()
	{
		System.out.println("Fruit is Pineapple");
	}

	@Test(priority=3 )
	public void custardapple()
	{
		System.out.println("Fruit is Custardapple");
	}


}
```



4) Enabled:

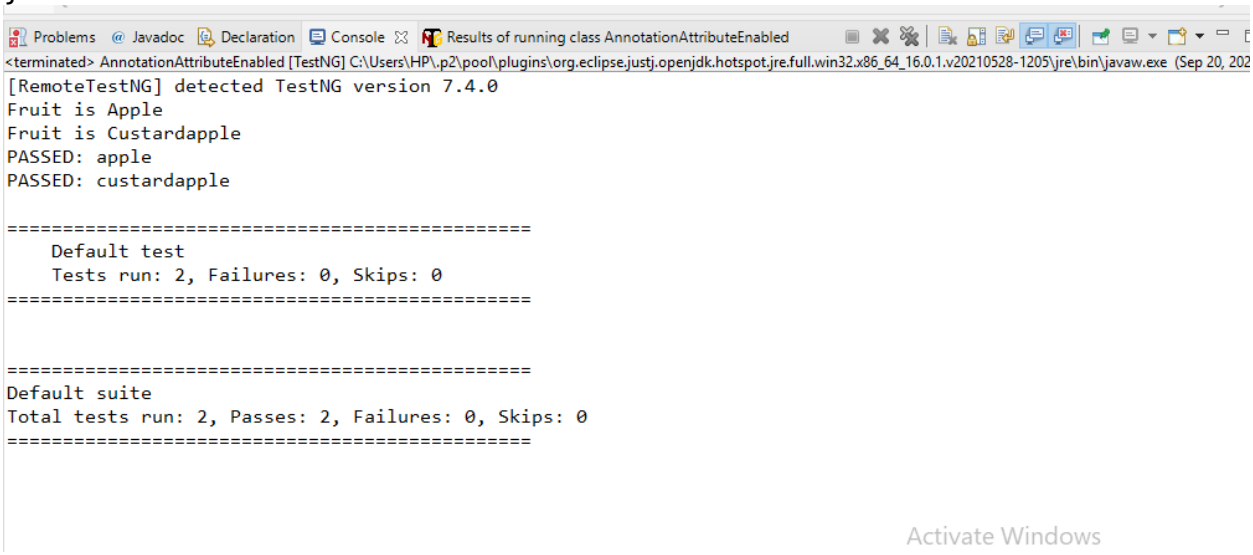
By using this attribute we can skip the Test case from execution.

The 'enabled' attribute contains the boolean value. By **default**, its value is true. If you want to skip some test method, then you need to explicitly specify '**false**' value.

That test case does not show into Test run i.e. in Test count which shows as output on console.

```
package TestNGBasic;

import org.testng.annotations.Test;

public class AnnotationAttributeEnabled {
	@Test(priority=1)
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test(priority=2, enabled=false)
	public void pineapple()
	{
		System.out.println("Fruit is Pineapple");
	}

	@Test(priority=3 )
	public void custardapple()
	{
```

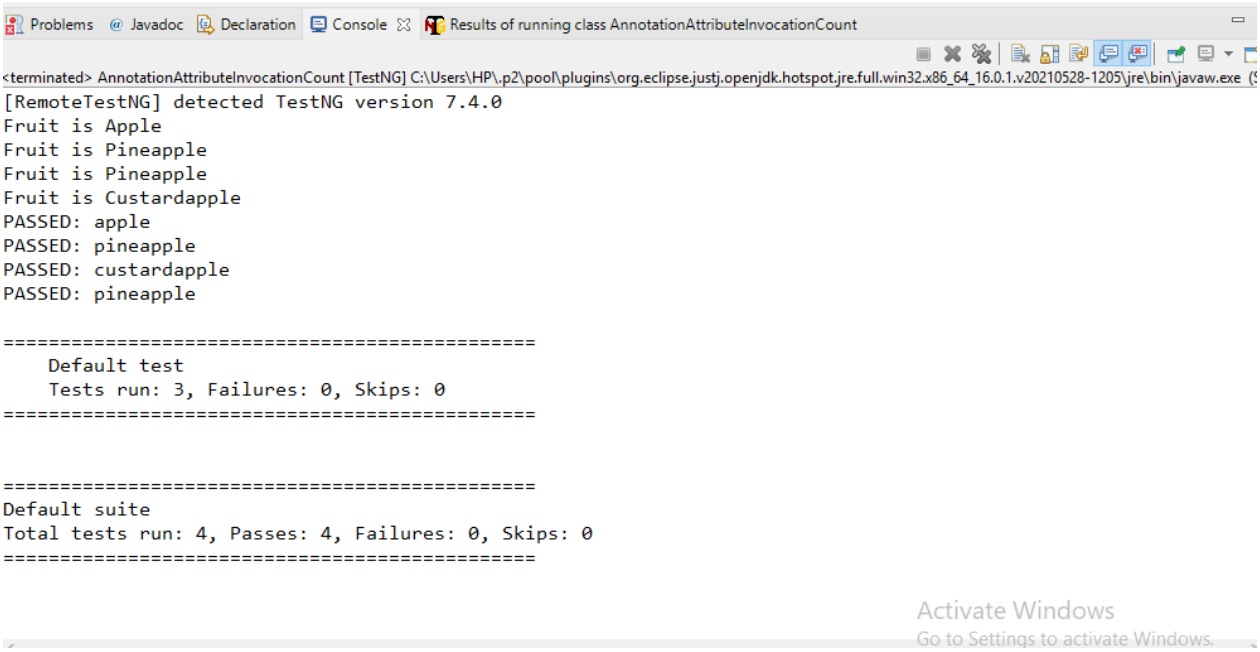
```
		System.out.println("Fruit is Custardapple");
	}

}
```



5) Invocation Count:

By providing count we can run same Test case again & again as no specified.

```
package TestNGBasic;

import org.testng.annotations.Test;

public class AnnotationAttributeInvocationCount {
	@Test(priority=1)
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test(priority=2, invocationCount=2)
	public void pineapple()
	{
		System.out.println("Fruit is Pineapple");
	}

	@Test(priority=3 )
	public void custardapple()
	{
		System.out.println("Fruit is Custardapple");
	}
}
```



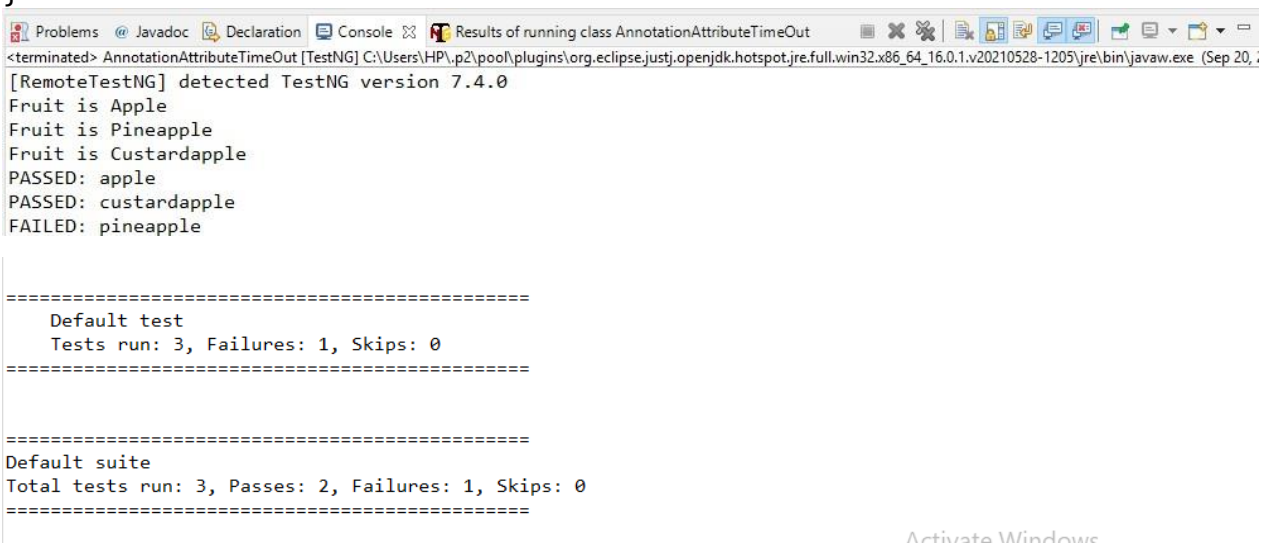
6) Timeout:

We can provide the timeout to Test case but in case Test case take more time as specified in timeout then Test case will be fail.

If one of the test cases is taking a long time due to which other test cases are failing. To overcome such situation, you need to mark the test case as fail to avoid the failure of other test cases. The timeOut is a time period provided to the test case to completely execute its test case.

```
package TestNGBasic;

import org.testng.annotations.Test;

public class AnnotationAttributeTimeOut {
	@Test(priority=1)
	public void apple()
	{
		System.out.println("Fruit is Apple");
	}

	@Test(priority=2, timeOut=3000)
	public void pineapple() throws InterruptedException
	{
		System.out.println("Fruit is Pineapple");
		Thread.sleep(4000);
```

```
	}

	@Test(priority=3 )
	public void custardapple()
	{
		System.out.println("Fruit is Custardapple");
	}

}
```



21) Selenium 21 Lecture

7) Groups

We can categorize test cases in groups, by declaring in as attribute.

The 'groups' attribute is used to group the different test cases that belong to the same functionality.

```
package TestNGBasic;

import org.testng.annotations.Test;

public class AnnotationAttributeGroup1 {

	@Test(groups="Sanity",priority=3)
  public void apple()
  {
		System.out.println("Fruit is apple");
  }

	@Test(groups="Regression",priority=1)
	public void pineapple()
	{
		System.out.println("Fruit is pineapple");
	}

	@Test(groups="Sanity",priority=1)
	public void custardapple()
	{
		System.out.println("Fruit is custardapple");
	}

	@Test(groups="Regression",priority=2)
	public void mango()
	{
		System.out.println("Fruit is Mango");
	}

	@Test(groups="Sanity",priority=2)
	public void banana()
	{
		System.out.println("Fruit is banana");
	}
}
```

```java
package TestNGBasic;

import org.testng.annotations.Test;

public class AnnotationAttributeGroup2 {

	@Test(groups="Sanity",dependsOnMethods="TestNGBasic.AnnotationAttributeGroup1.custardapple")
  public void m1()
	{
		System.out.println("m1 is running");
  }

	@Test(groups="Regression",dependsOnMethods="TestNGBasic.AnnotationAttributeGroup1.mango")
	public void m2()
	{
		System.out.println("m2 is running");
	}

	@Test(groups="Sanity",dependsOnMethods="m1")
	public void m3()
	{
		System.out.println("m3 is running");
	}
}
```

1) We can execute Test Suit by using <Group> tag within it & by using <run> ,<include> tag within the <groups> tag.

We can use **<!-- at the starting & --> at the ending for commenting** multiple lines in XML files.

AnnotationAttributeGroup1.java | AnnotationAttributeGroup2.java | testng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite1">
 <groups>
  <run>
     <include name="Sanity" />
     <include name="Regression" />
  </run>
 </groups>
  <test thread-count="5" name="Test">
    <classes>
    <!--  <class name="TestNGBasic.AnnotationAttributeInvocationCount"/>
      <class name="TestNGBasic.AnnotationAttributePrority"/>
      <class name="TestNGBasic.AnnotationAttributeDependsonMethod"/>
      <class name="TestNGBasic.TestNG2"/>
      <class name="TestNGBasic.TestNG3"/>
      <class name="TestNGBasic.AnnotationAttributeEnabled"/>
      <class name="TestNGBasic.AnnotationAttributeTimeOut"/>
      <class name="TestNGBasic.AnnotationAttributeDescription"/> -->
      <class name="TestNGBasic.AnnotationAttributeGroup1"/>
      <class name="TestNGBasic.AnnotationAttributeGroup2"/>
     <!-- <class name="TestNGBasic.NewTest"/>
      <class name="TestNGBasic.Annotation"/>
      <class name="TestNGBasic.AnnotationMultipleTests"/>
      <class name="TestNGBasic.TestNGFirstProgram"/> -->
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite1 -->
```

Output:

Problems | Javadoc | Declaration | Console | Results of running suite

```
<terminated> TestNG_testng.xml [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (Sep 21, 2021, 12:13
[RemoteTestNG] detected TestNG version 7.4.0
Fruit is custardapple
Fruit is pineapple
Fruit is banana
Fruit is Mango
Fruit is apple
m1 is running
m2 is running
m3 is running

===============================================
Suite1
Total tests run: 8, Passes: 8, Failures: 0, Skips: 0
===============================================
```

2)When we have large number of groups in that situation we can create Main group of all groups & execute specific groups & we can also Exclude specific group from execution by using <define tag>.

AnnotationAttributeGroup1.java | AnnotationAttributeGroup2.java | testng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite1">
 <groups>

  <define name="all">
     <include name="all" />
  </define>

     <run>
       <exclude name="Regression" />
      </run>

 </groups>
  <test thread-count="5" name="Test">
    <classes>
```

Output:

Problems | Javadoc | Declaration | Console | Results of running suite

<terminated> TestNG_testng.xml [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 21, 2021, 12:09:

```
[RemoteTestNG] detected TestNG version 7.4.0
Fruit is custardapple
Fruit is banana
Fruit is apple
m1 is running
m3 is running

===============================================
Suite1
Total tests run: 5, Passes: 5, Failures: 0, Skips: 0
===============================================
```

22) Selenium 22 Lecture

8) Parameter:

Parameterization in Selenium

Parameterization in Selenium is a process to parameterize the test scripts in order to pass multiple data to the application at runtime. It is a strategy of execution which automatically runs test cases multiple times using different values. The concept achieved by parameterizing the test scripts is called **Data Driven Testing**.

Type of Parameterization in TestNG-

To make parameterization more clear, we will go through the parameterization options in one the most popular framework for Selenium Webdriver – **TestNG**.

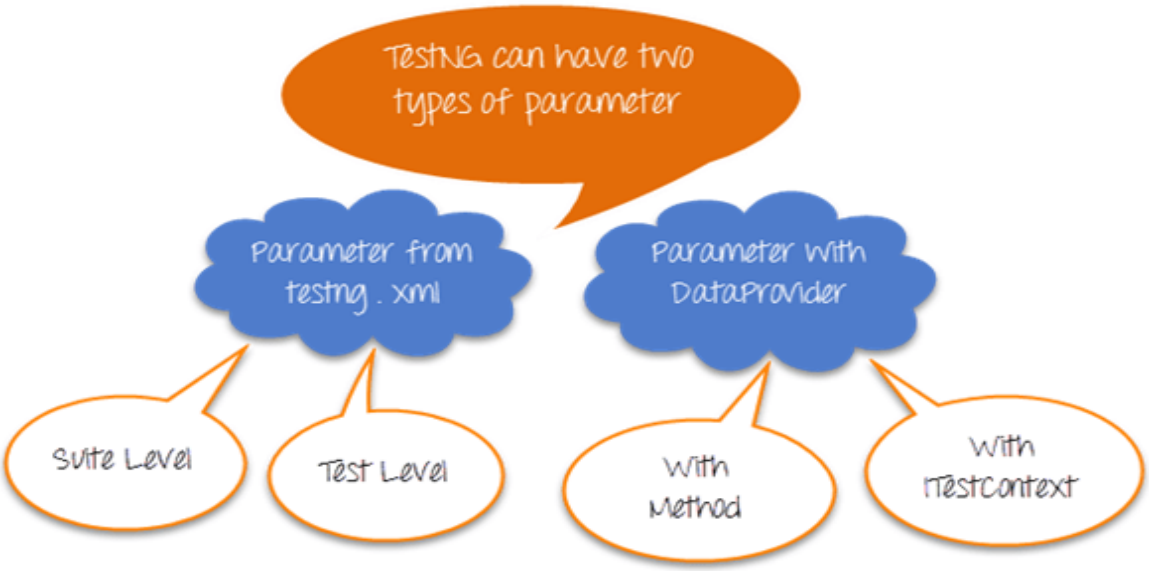
There are **two ways** by which we can achieve parameterization in TestNG

1) With the help of **Parameters annotation** and **TestNG XML** file.

```
@Parameters({"name","searchKey"})
```

2) With the help of **DataProvider** annotation.

```
@DataProvider(name="SearchProvider")
```



1) Parameters annotation with Testng.xml

Select parameterization using annotations when you do want to deal with complexity & the number of input combinations are less.

Program1: Passing Parameter to method within test case through TestNG.xml file.

```
package TestNGBasic;

import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class Parameter {
    @Test
     @Parameters("Smoke")
public void f(String s)
     {
           System.out.println("Test running of :"+s);
    }
}
```

*Parameter.java | testng.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
3 <suite name="Suite">
4  <test thread-count="5" name="Test">
5  <parameter name="Smoke" value="Today"/>
6
7     <classes>
8       <class name="TestNGBasic.Parameter"/>
```

Problems | Javadoc | Declaration | Console | Results of running suite

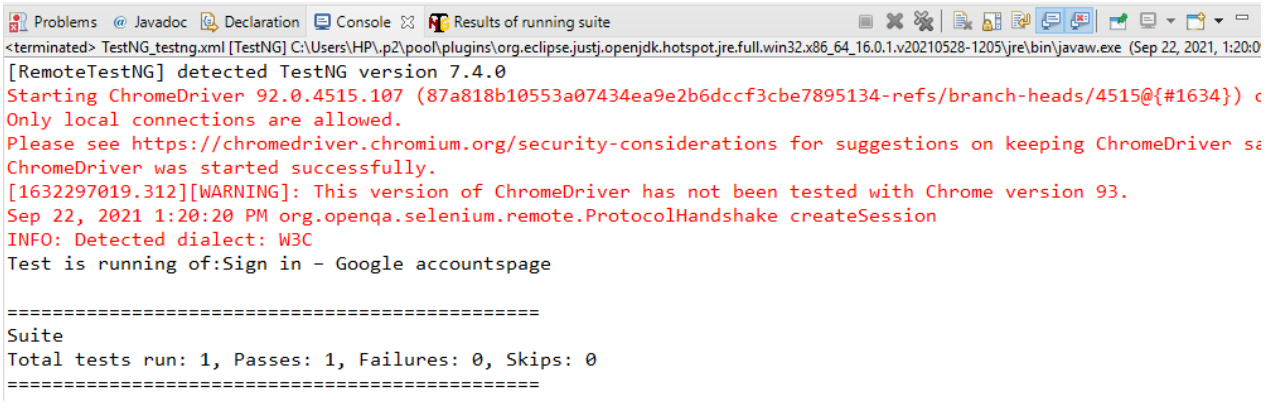
```
<terminated> TestNG_testng.xml [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v202105
[RemoteTestNG] detected TestNG version 7.4.0
Test running of :Today

===============================================
Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

Program2: Launch a Gmail Web page by Passing Parameter to Test method through the TestNg.xml file.

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

/*Date:22/09/2021
Program2:Launch a Gmail Web page by Passing Parameter to Test method
through the TestNg.xml file.*/

public class Parameter2 {
	WebDriver driver;
	@BeforeTest
	@Parameters("browser")
	public void BrowserTest(String s)
	{
		if(s.equalsIgnoreCase("Chrome"))
		{

	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");
			driver=new ChromeDriver();
			driver.get("https://accounts.google.com/");
			driver.manage().window().maximize();
		}
	}

	@Test
	public void f1()
	{
		String title=driver.getTitle();
		System.out.println("Test is running of:" +title +"page");
	}

}
```

testng.xml Parameter2.java

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
3 <suite name="Suite">
4  <test thread-count="5" name="Test">
5  <parameter name="browser" value="Chrome"/>
6
7     <classes>
8       <class name="TestNGBasic.Parameter2"/>
```



Program3: Perform Parallel testing open a same web page in same browser two times means launch same browser two times by passing Parameter through TestNg.xml file.

While performing Parallel testing(cross browser testing)

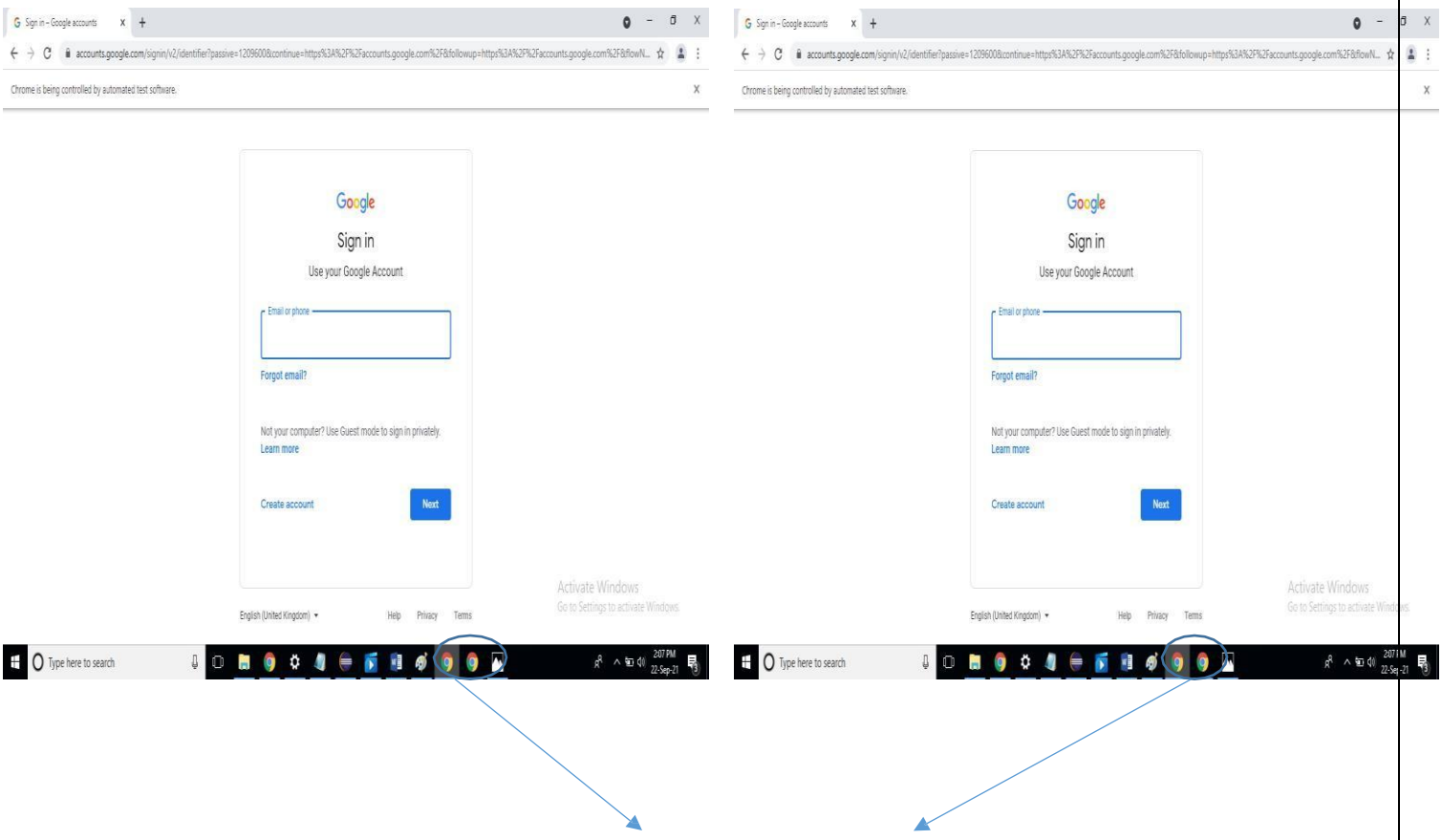
i) Select Parallel=Tests while creating Test Suite

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;

public class Parameter3 {
      WebDriver driver;
      @BeforeTest
      @Parameters("browser")
      public void BrowserTest(String s)
      {
```

```
            if(s.equalsIgnoreCase("Chrome"))
            {

      System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");
                  driver=new ChromeDriver();
                  driver.get("https://accounts.google.com/");
                  driver.manage().window().maximize();
            }
            else if(s.equalsIgnoreCase("Chrome"))
            {

      System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");
                  driver=new ChromeDriver();
                  driver.get("https://accounts.google.com/");
                  driver.manage().window().maximize();
      }

 }
}
```

Parameter3.java | Parameter2.java | testng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite parallel="tests" name="Suite">

<test thread-count="1" parallel="tests" name="Test">
 <parameter name="browser" value="Chrome" />
    <classes>
     <class name="TestNGBasic.Parameter3"/>
    </classes>
</test> <!-- Test -->

<test thread-count="1" parallel="tests" name="Test2">
 <parameter name="browser" value="chrome" />
   <classes>
     <class name="TestNGBasic.Parameter3"/>
   </classes>
</test> <!-- Test2 -->


</suite> <!-- Suite -->

```



As we can see that Gmail webpage is opened in two Chrome browser at a same time.

Program4: Perform Parallel testing i.e. Cross browser testing by using Parameter which is being pass through TestNg.xml file.

Means Open one Gmail web page in Google Chrome browser & One Gmail page in Mozila Firefox browser.

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class Parameter4 {
	WebDriver driver;

	@BeforeTest
	@Parameters("browser")
	public void BrowserTest(String s)
	{
		if(s.equalsIgnoreCase("Chrome"))
		{

	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");
			driver=new ChromeDriver();
			driver.get("https://accounts.google.com/");
			driver.manage().window().maximize();
		}
		else if(s.equalsIgnoreCase("Firefox"))
		{

	System.setProperty("webdriver.gecko.driver","C:\\Users\\HP\\Deskt
op\\Dhananjay Deshmukh\\Software Testing\\Selenium\\geckodriver.exe");
			driver=new FirefoxDriver();
			driver.get("https://accounts.google.com/");
			driver.manage().window().maximize();
		}
	}
	 @Test
	 public void WebpageTitle()
	 {
```

```
            String title=driver.getTitle();
            System.out.println("Title of Web Page is:" +title);
        }
    }
```

testng.xml | Parameter4.java

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite parallel="tests" name="Suite">

<test thread-count="1" parallel="tests" name="ChromeTest">
 <parameter name="browser" value="Chrome" />
    <classes>
     <class name="TestNGBasic.Parameter4"/>
    </classes>
</test> <!-- Test -->

<test thread-count="1" parallel="tests" name="FirefoxTest">
 <parameter name="browser" value="Firefox" />
   <classes>
     <class name="TestNGBasic.Parameter4"/>
   </classes>
</test> <!-- Test2 -->


</suite> <!-- Suite -->
```

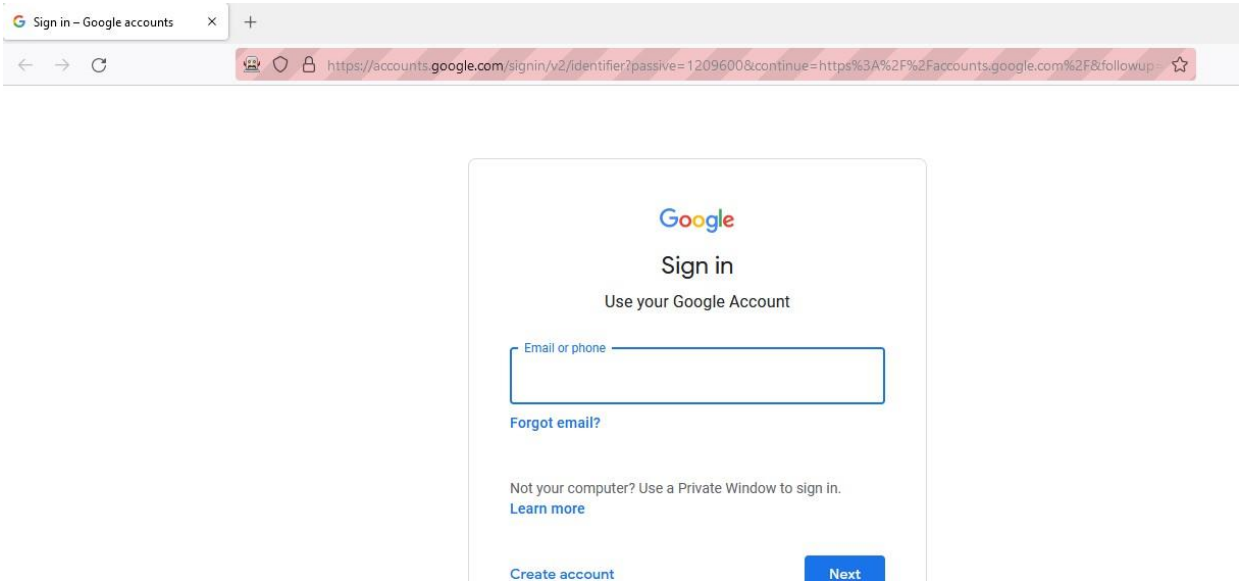
Console Output:

Problems | Javadoc | Declaration | Console | Results of running suite

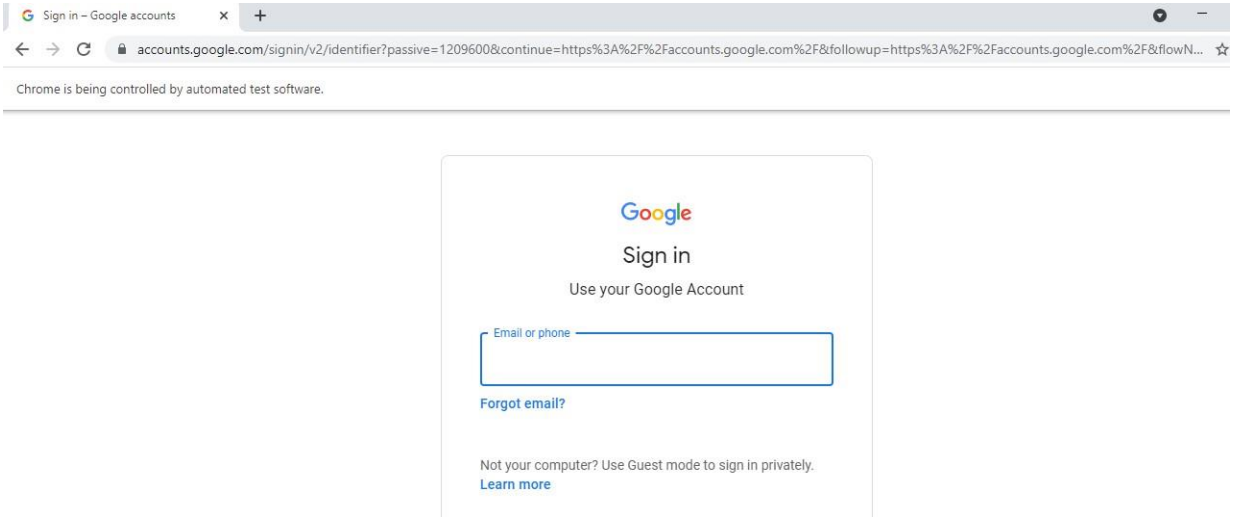
<terminated> TestNG_testng.xml [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 22, 2021, 2:56:54 PM – 2:58:04 PM)

```
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634}) on port 4233
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
1632302825372   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "-foregrou
[1632302829.994][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
1632302832535   Marionette      INFO    Marionette enabled
Sep 22, 2021 2:57:13 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
console.warn: SearchSettings: "get: No settings file exists, new profile?" (new NotFoundError("Could not open the file at C:\\Users\\HP\\App
console.error: Region.jsm: "Error fetching region" (new Error("TIMEOUT", "resource://gre/modules/Region.jsm", 772))
console.error: Region.jsm: "Failed to fetch region" (new Error("TIMEOUT", "resource://gre/modules/Region.jsm", 419))
1632302845883   Marionette      INFO    Listening on port 13568
1632302850046   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
Sep 22, 2021 2:57:30 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Title of Web Page is:Sign in - Google accounts
Title of Web Page is:Sign in - Google accounts

===============================================
Suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```

Mozila Browser:



Google Chrome Browser:



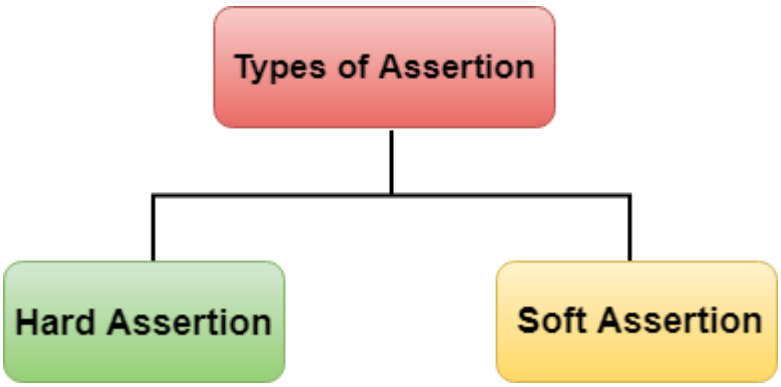
23) Selenium 23 Lecture

Assertion in Selenium WebDriver using TestNg

Assertion in testNg are way to verify that expected result or value and actual result or value matched or not.

Syntax- Assert.method(actual, Expected)

Actual result is compared with expected result with the help of Assertion. It means verification is done to check if state of the application is same to what we are expecting or not. For creating assertion we are going to use Assert class provided by TestNG.



There are two types of Assertion:-

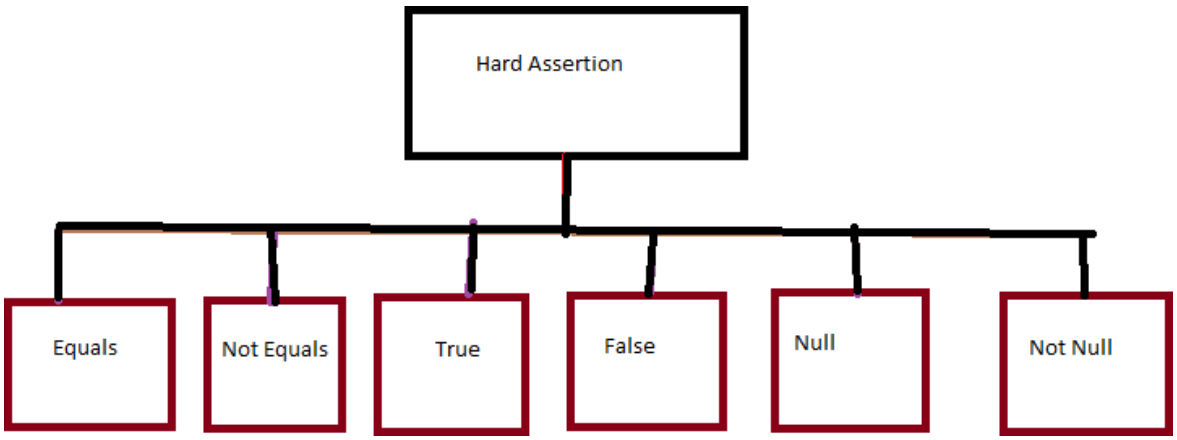
1. Hard Assertions.
2. Soft assertions.

1. Hard Assertions : Hard Validation

If test assertion failed immediately test case mark as failed & immediately terminated.

When any assert statement fails this type of assertion throws an exception immediately and continues with the next test in the test suite.

Hard Assertion can be of following types:-



i) **assertEquals**

This is used to compare expected and actual values in selenium webdriver. The assertion passes with no exception whenever the expected and actual values are same. But, if the actual and expected values are not same then assert fails with an exception and the test is marked as failed.

Syntax : Assert.assertEquals(actual, expected);

```
package TestNGBasic;

/*Date:22/09/2021*/

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

public class AssertionHardEquals {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
```

```
        WebDriver driver=new ChromeDriver();

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

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

         String title=driver.getTitle();

         Assert.assertEquals(title,"Google");

         System.out.println("Tile of web page is :"+title);
    }
}
```

```
Problems  Javadoc  Declaration  Console  Results of running class AssertionHardEquals
<terminated> AssertionHardEquals [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping C
ChromeDriver was started successfully.
[1632376241.141][WARNING]: This version of ChromeDriver has not been tested with Chrome version 9
Sep 23, 2021 11:20:42 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Tile of web page is :Google
PASSED: f

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

ii) assertNotEquals –

This is just opposite to the assertEquals. The assertion passes with no exception whenever the expected and actual values are not same. But, if the actual and expected values are same then assert fails with an exception and the test is marked as failed.

Syntax : Assert.assertNotEquals(actual, expected, message);

Program1:When NotEuals() true

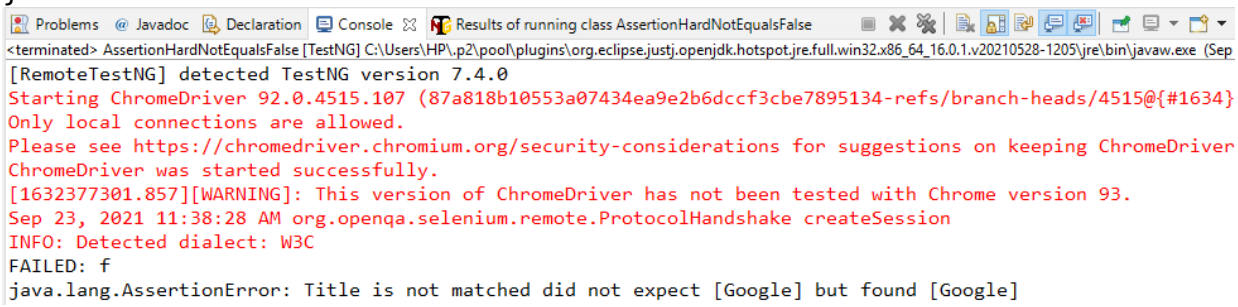
```
package TestNGBasic;

/*Date:23/09/2021*/
```

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class AssertionHardNotEquals {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       Assert.assertNotEquals(title,"Googlee","Title is not matched");

       System.out.println("Tile of web page is :"+title);
  }
}
```

```
Problems  Javadoc  Declaration  Console  Results of running class AssertionHardNotEquals
<terminated> AssertionHardNotEquals [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 23
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDrive
ChromeDriver was started successfully.
[1632376903.382][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 23, 2021 11:32:03 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Tile of web page is :Google
PASSED: f

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

Program2:When NotEquals is Flase.

```
package TestNGBasic;

/*Date:23/09/2021*/

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

public class AssertionHardNotEqualsFalse {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       Assert.assertNotEquals(title,"Google","Title is not matched");

       System.out.println("Tile of web page is :"+title);
  }
}
```



```
===============================================
    Default test
    Tests run: 1, Failures: 1, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 0, Failures: 1, Skips: 0
===============================================
```


iii) assertTrue:

Assertion verifies the boolean value returned by a condition. If the boolean value is true, then assertion passes the test case, and if the boolean value is false, then assertion aborts the test case by an exception. Syntax of AssertTrue() method is given below:

Syantax: Assert.AssertTrue(condition);

```
package TestNGBasic;

import org.openqa.selenium.By;

/*Date:23/09/2021*/

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class AssetionHardTrue {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

```
        String title=driver.getTitle();

        Boolean b=title.equalsIgnoreCase("Google");

        Assert.assertTrue(b);

        System.out.println("Tile of web page is :"+title);
    }
}
```

```
<terminated> AssetionHardTrue [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (Sep 23, 2021, 12:31
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634})
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver s
ChromeDriver was started successfully.
[1632380523.060][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 23, 2021 12:32:05 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Tile of web page is :Google
PASSED: f

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

iv) assertFalse:

Assertion verifies the boolean value returned by a condition. If the boolean value is false, then assertion passes the test case, and if the boolean value is true, then assertion aborts the test case by an exception. Syntax of AssertFalse() method is given below:

Syntax: Assert.AssertFalse(condition);

```
package TestNGBasic;

import org.openqa.selenium.By;

/*Date:23/09/2021*/

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
```

```java
public class AssertionHardFalse {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       Boolean b=title.equalsIgnoreCase("Google");

       Assert.assertFalse(b);

       System.out.println("Tile of web page is :"+title);
  }
}
```

Problems Javadoc Declaration Console Results of running class AssertionHardFalse

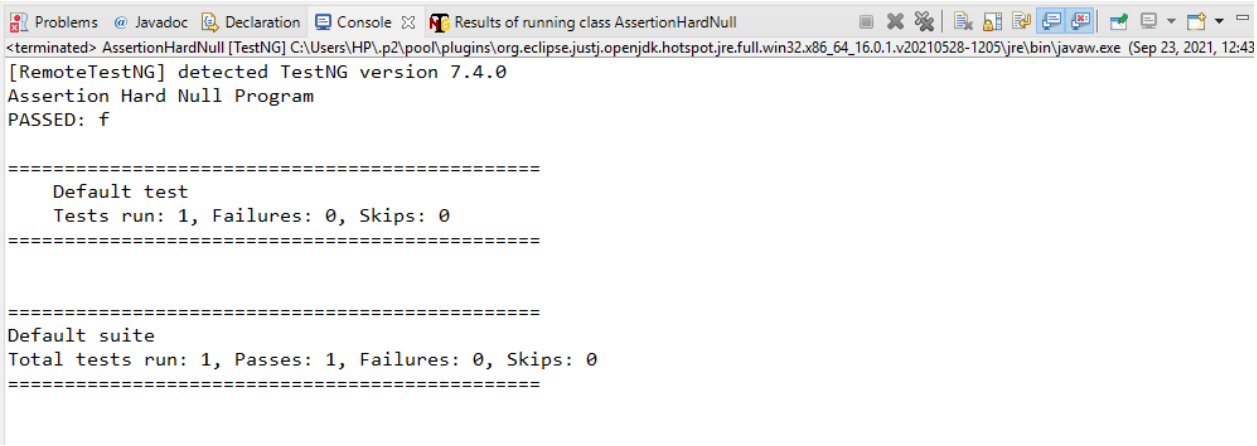
```
<terminated> AssertionHardFalse [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 23, 2021, 12:36
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634}) c
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver sa
ChromeDriver was started successfully.
[1632380791.468][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 23, 2021 12:36:33 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
FAILED: f
java.lang.AssertionError: expected [false] but found [true]

===============================================
    Default test
    Tests run: 1, Failures: 1, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 0, Failures: 1, Skips: 0
===============================================
```

v) assertNull:

AssertNull() is a method that verifies whether the object is null or not. If an object is null, then assertion passes the test case, and the test case is marked as "passed", and if an object is not null, then assertion aborts the test case and the test case is marked as "failed". Syntax of AssertNull() method is given below:

Syntax: Assert.assertNull(object);

```
package TestNGBasic;

import org.testng.Assert;
import org.testng.annotations.Test;

public class AssertionHardNull {
  @Test
  public void f() {

        Assert.assertNull(null);
        System.out.println("Assertion Hard Null Program");

  }
}
```



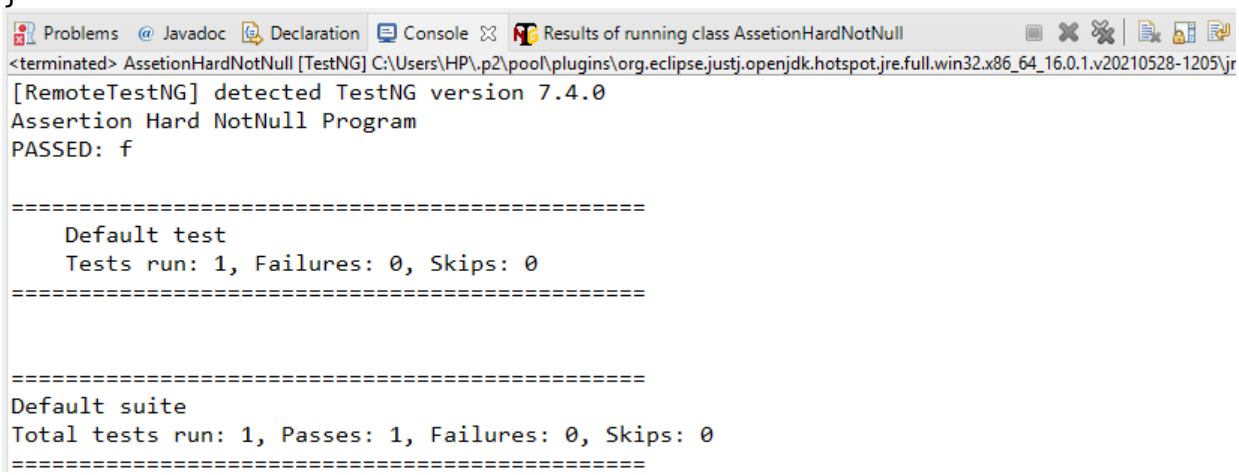
vi) assertNotNull:

AssertNotNull() is a method that verifies whether the object is null or not. If an object is not null, then assertion passes the test case and test case is marked as "passed", and if an object is null,

then assertion aborts the test case and test case is marked as "failed". Syntax of AssertNotNull() method is given below:

Syantax: Assert.assertNotNull(object);

```
package TestNGBasic;

import org.testng.Assert;
import org.testng.annotations.Test;

public class AssetionHardNotNull {
  @Test
  public void f() {

        Assert.assertNotNull(9);
        System.out.println("Assertion Hard NotNull Program");

  }
}
```



1. Soft Assertions : Soft Validation

To continue the execution even if test case failed and later on mark the result as fail.
If Test case failed next line will be executed .

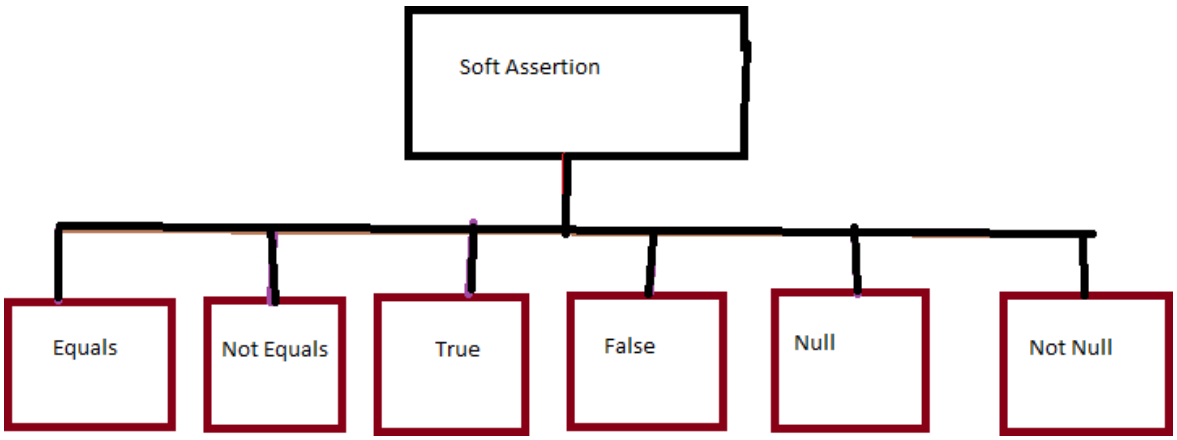
Syntax: create the object of soft assertion
SoftAssert sa=new SoftAssert();
sa.AssertyAll();
We need to provide the last statement of Test Case to mark it as a failed.

SoftAssert in TestNG helps to collect all the assertions throughout the @Test method. And to see assertions result at the end of the test, we have to invoke assertAll().

SoftAssert don't throw an exception when an assert fails, but it records the failure. The test execution will continue with the next step after the assert statement. Calling assertAll() will cause an exception to be thrown if at least one assertion failed.

If we use normal asserts like Assert.assertTrue() or Assert.assertEquals() in TestNG, @Test Method will immediately fail after any of the Asserts fails.

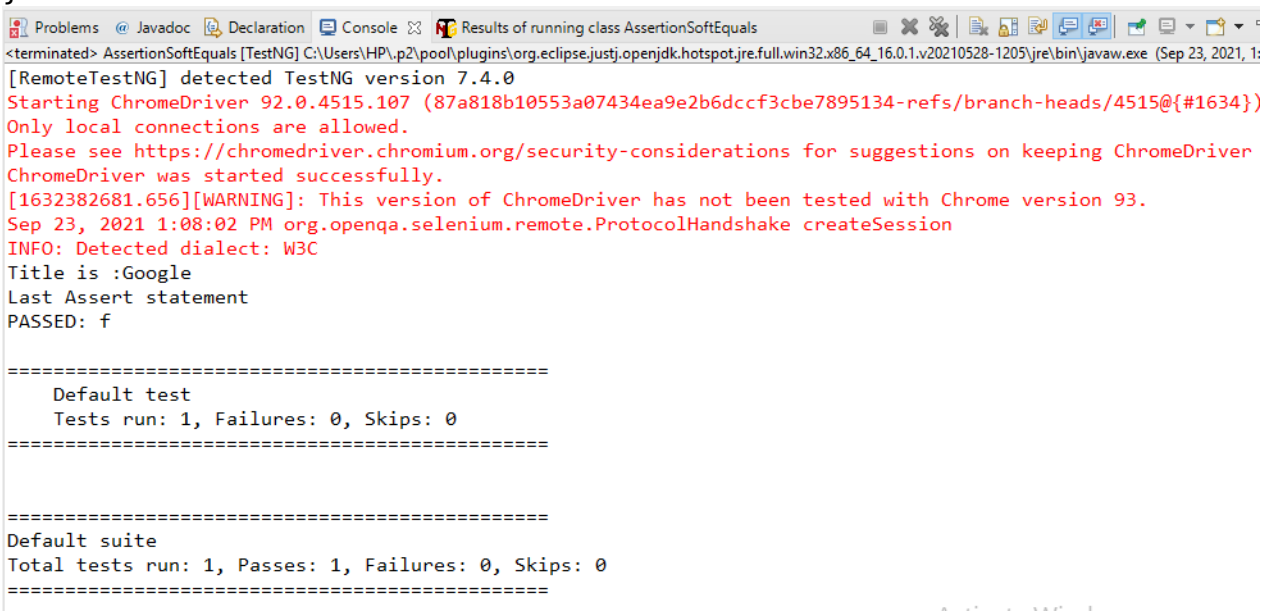
There are multiple scenarios where you want to continue the execution even if some assert fails and see the result at the end of the test. For instance, after landing to a page where you want to validate multiple cases we can use softAsserts and throw error at the end of the test execution / before user navigates to next page.



i) assertEquals:

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftEquals {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();
```

```
        SoftAssert as=new SoftAssert();

        as.assertEquals(title,"Google");

        System.out.println("Title is :" +title);

        as.assertAll();

        System.out.println("Last Assert statement");

    }
}
```



ii) assertNotEquals:

Program1: When NotEquals True

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftNotEqualsTrue {
  @Test
  public void f() {
```

```
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       SoftAssert as=new SoftAssert();

       as.assertNotEquals(title,"Googlee");

       System.out.println("Title is :" +title);

       as.assertAll();

       System.out.println("Last Assert statement");

  }
}
```

Program2: When NotEquals is False

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
```

```java
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftNotEqualsFalse {
  @Test
  public void f() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       SoftAssert as=new SoftAssert();

       as.assertNotEquals(title,"Google");

       System.out.println("Title is :" +title);

       as.assertAll();

       System.out.println("Last Assert statement");

  }
}
```

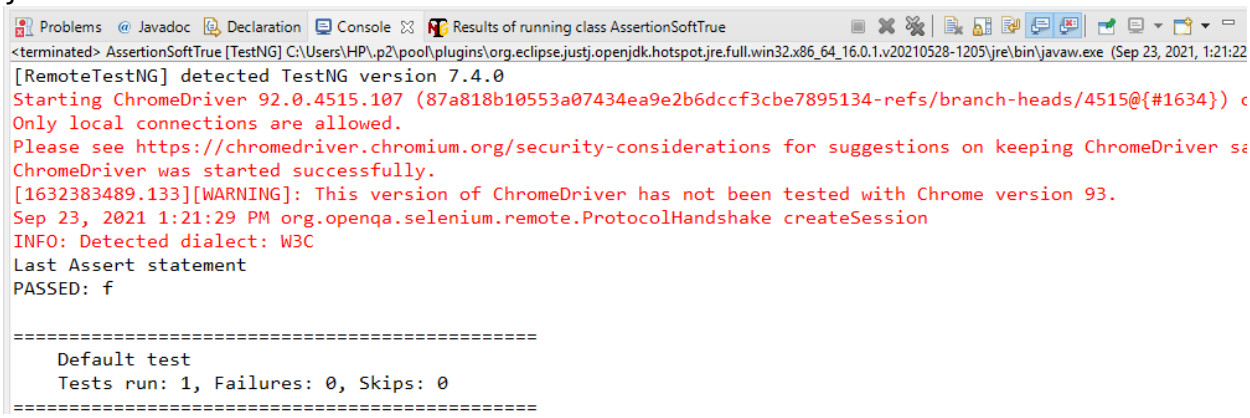
```
Problems  Javadoc  Declaration  Console  Results of running class AssertionSoftNotEqualsFalse
<terminated> AssertionSoftNotEqualsFalse [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 23, 2
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634}) c
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver sa
ChromeDriver was started successfully.
[1632383163.280][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 23, 2021 1:16:03 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Title is :Google
FAILED: f
java.lang.AssertionError: The following asserts failed:
        did not expect [Google] but found [Google]
```

```
===============================================
    Default test
    Tests run: 1, Failures: 1, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 0, Failures: 1, Skips: 0
===============================================
```


iii) assertTrue:

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftTrue {
  @Test
  public void f() {


System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       Boolean b=title.equalsIgnoreCase("Google");

       SoftAssert as=new SoftAssert();

       as.assertTrue(b);


       as.assertAll();

       System.out.println("Last Assert statement");
```

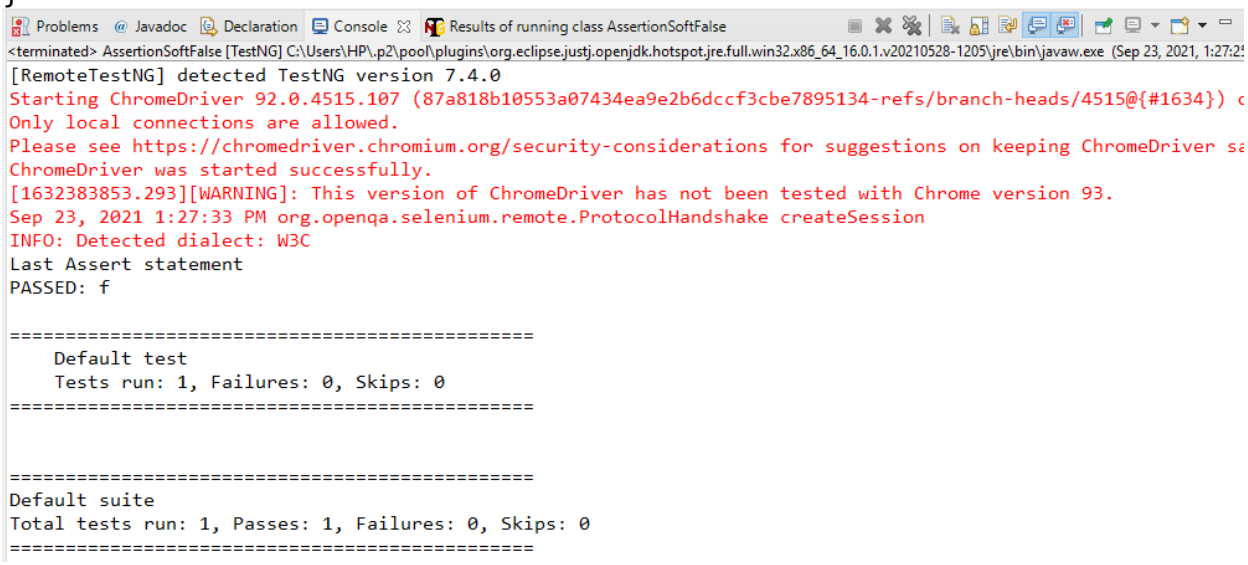
```
  }
}
```



iv) assertFalse:

```
package TestNGBasic;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftFalse {
  @Test
  public void f() {
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

      WebDriver driver=new ChromeDriver();

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

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

       String title=driver.getTitle();

       Boolean b=title.equalsIgnoreCase("Googlee");

       SoftAssert as=new SoftAssert();

       as.assertFalse(b);
```

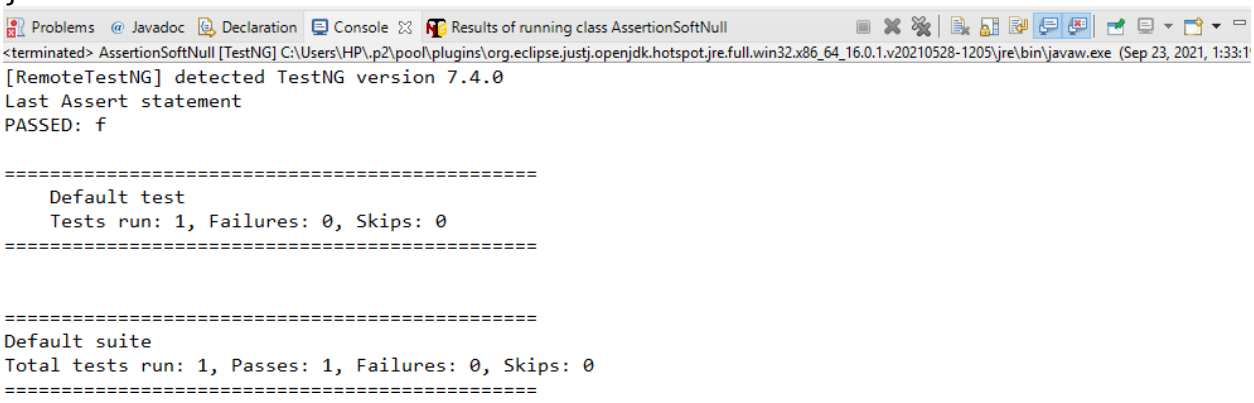
```
        as.assertAll();

        System.out.println("Last Assert statement");


    }
}
```



v) assertNull:

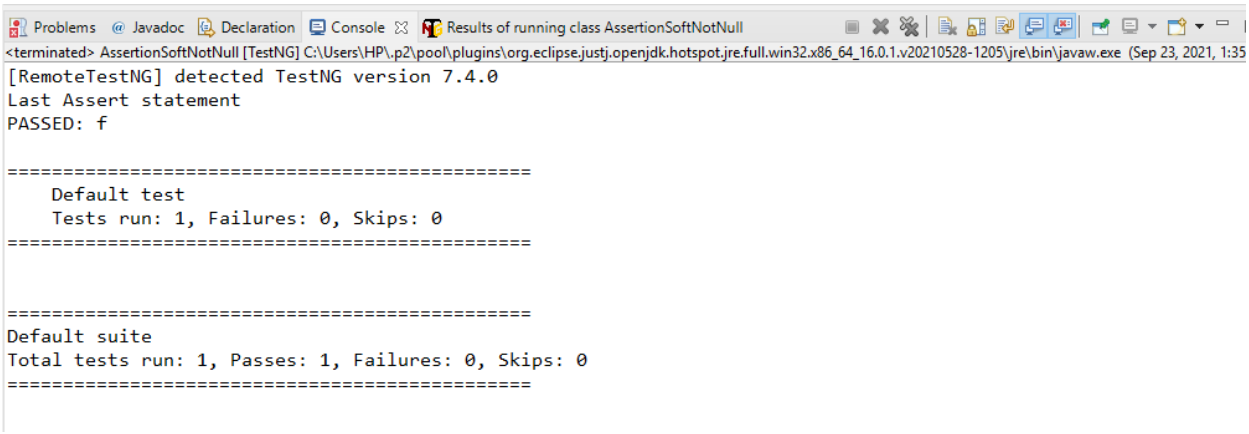
```
package TestNGBasic;

import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftNull {
  @Test
  public void f() {

        SoftAssert as=new SoftAssert();

      as.assertNull(null);

      as.assertAll();

      System.out.println("Last Assert statement");
```

```
  }
}
```



vi) assertNotNull:

```
package TestNGBasic;

import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionSoftNotNull {
  @Test
  public void f() {

SoftAssert as=new SoftAssert();

      as.assertNotNull(10);

      as.assertAll();

      System.out.println("Last Assert statement");

  }

 }
```


Problems
Javadoc
Declaration
Console
Results of running class AssertionSoftNotNull
<terminated> AssertionSoftNotNull [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 23, 2021, 1:35
[RemoteTestNG] detected TestNG version 7.4.0
Last Assert statement
PASSED: f
Default test
Tests run: 1, Failures: 0, Skips: 0
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

24) Selenium 24 Lecture

Listeners in TestNG

Listeners annotation which listens to evry event that occurs in selenium script.

Listeners- are activated either before the test case or after the test case.

For this we need to implement itestListenenrs (interface) in the class.

Listener is defined as interface that modifies the default TestNG's behavior. As the name suggests Listeners "listen" to the event defined in the selenium script and behave accordingly. It is used in selenium by implementing Listeners Interface. It allows customizing TestNG reports or logs. There are many types of TestNG listeners available.

Types of Listeners in TestNG

There are many types of listeners which allows you to change the TestNG's behavior.

Below are the few TestNG listeners:

1. IAnnotationTransformer ,
2. IAnnotationTransformer2 ,
3. IConfigurable ,
4. IConfigurationListener ,
5. IExecutionListener,
6. IHookable ,
7. IInvokedMethodListener ,
8. IInvokedMethodListener2 ,
9. IMethodInterceptor ,
10. IReporter,
11. ISuiteListener,
12. ITestListener .

Above Interface are called TestNG Listeners. These interfaces are used in selenium to generate logs or customize the TestNG reports.

ITestListener has following methods

- **OnStart-** OnStart method is called when any Test starts.
- **onTestSuccess-** onTestSuccess method is called on the success of any Test.

- **onTestFailure-** onTestFailure method is called on the failure of any Test.
- **onTestSkipped-** onTestSkipped method is called on skipped of any Test.
- **onTestFailedButWithinSuccessPercentage-** method is called each time Test fails but is within success percentage.
- **onFinish-** onFinish method is called after all Tests are executed.

Implements ITestListeners interface which is from org.testng.ItetListener package.

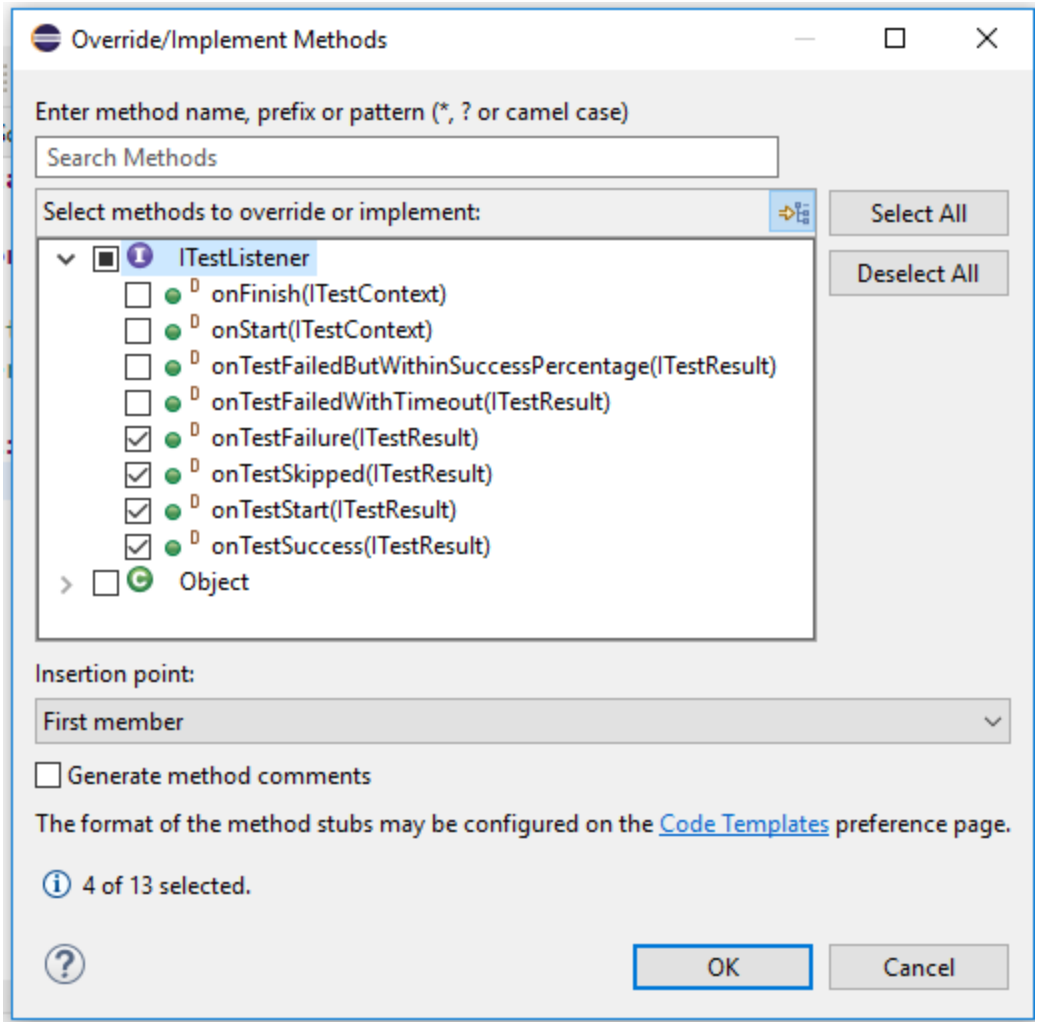
For adding unimplemented methods in a class

i) Right click on class window

ii) Click on Source

iii) Select Overwrite/Unimplemented methods

iv) Select the methods of ITestResult from list & click ok



Then we have to do configuration in Test class .i.e. in First class. With the second class i.e. the class in which interface is implemented.

Means we have to Configure Listeners into our class.

There are two ways for Configuring Listeners into class

1) With the class(.java file)

2) With the Suite(.xml file)

1) Listener With the class(.java file)

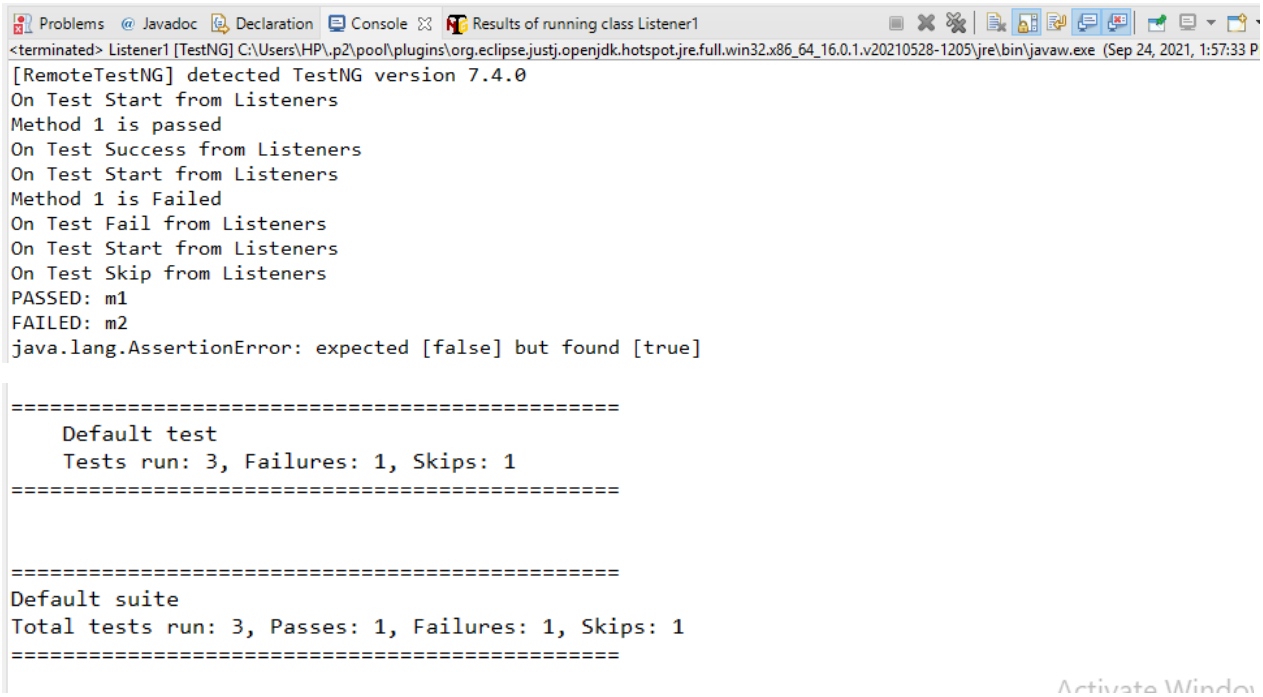
We are passing .class file while configuring Listeners with the class because program is already compiled & .class file is created & we are passing it at the run time.

We are using Listener Annotation above the class.

```
package TestNGBasic;
/*Date:24/09/2021
  Refer Listeners1.java & Listeners1.java*/

import org.testng.Assert;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners(TestNGBasic.Listener2.class)
public class Listener1 {

	@Test(priority=0)
	public void m1() {
		System.out.println("Method 1 is passed");
	}

	@Test(priority=1)
	public void m2() {
		System.out.println("Method 1 is Failed");
		Assert.assertFalse(true);
	}

	@Test(dependsOnMethods="m2")
	public void m3() {
		System.out.println("Method 1 is Skipped");
	}

}
```

```
package TestNGBasic;

import org.testng.ITestListener;
import org.testng.ITestResult;

/*Date:24/09/2021
Refer Listeners1.java & Listeners1.java*/

public class Listener2 implements ITestListener{

	@Override
	public void onTestStart(ITestResult result) {
		System.out.println("On Test Start from Listeners");
	}

	@Override
	public void onTestSuccess(ITestResult result) {
		System.out.println("On Test Success from Listeners");
	}

	@Override
	public void onTestFailure(ITestResult result) {
		System.out.println("On Test Fail from Listeners");
	}

	@Override
	public void onTestSkipped(ITestResult result) {
```

```
        System.out.println("On Test Skip from Listeners");
    }

}
```



2) Listener With the Suite level(.xml file)

Listener1.Java

```
package TestNGBasic;
/*Date:24/09/2021

  Refer Listeners1.java & Listeners1.java
  Configuring Listeners with the Class*/

import org.testng.Assert;


import org.testng.annotations.Listeners;
import org.testng.annotations.Test;
```

```java
@Listeners(TestNGBasic.Listener2.class)
public class Listener1 {

	@Test(priority=0)
	public void m1() {
		System.out.println("Method 1 is passed");
	}

	@Test(priority=1)
	public void m2() {
		System.out.println("Method 1 is Failed");
		Assert.assertFalse(true);
	}

	@Test(dependsOnMethods="m2")
	public void m3() {
		System.out.println("Method 1 is Skipped");
	}

}
```

Listener2.Java file

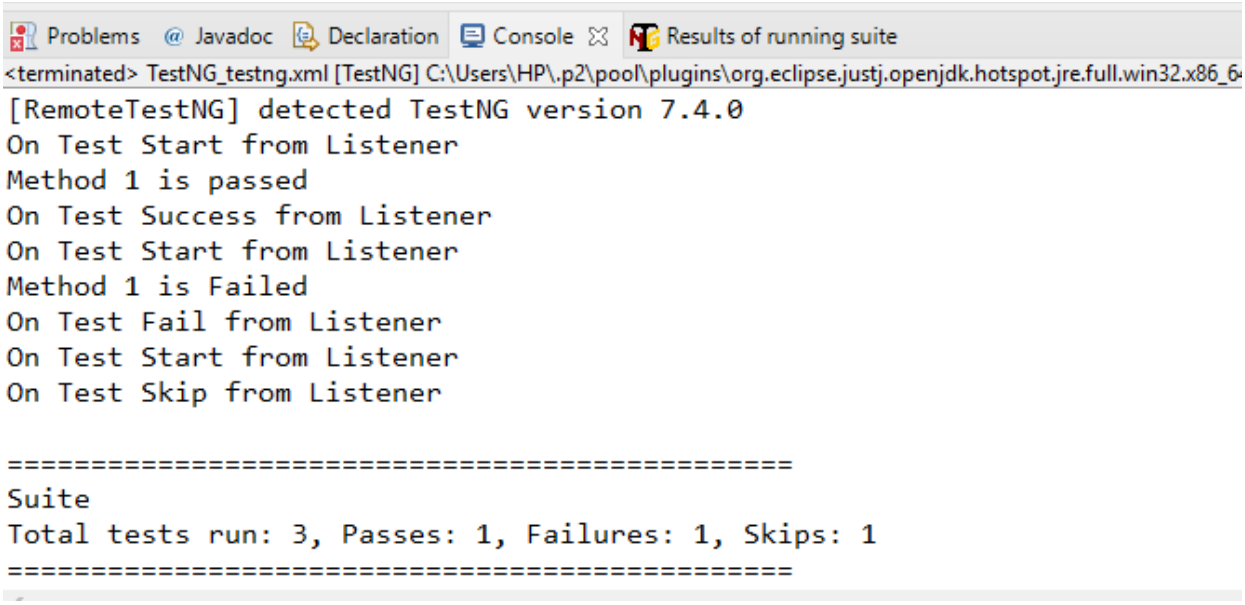
```java
package TestNGBasic;


import org.testng.ITestListener;

import org.testng.ITestResult;

import org.testng.annotations.Test;


public class Listener2 implements ITestListener {


	@Override

	public void onTestStart(ITestResult result) {

		System.out.println("On Test Start from Listener");

	}
```

```
    @Override
    public void onTestSuccess(ITestResult result) {
        System.out.println("On Test Success from Listener");
    }

    @Override
    public void onTestFailure(ITestResult result) {
        System.out.println("On Test Fail from Listener");
    }

    @Override
    public void onTestSkipped(ITestResult result) {
        System.out.println("On Test Skip from Listener");
    }

}
```

TestNG.xml:

testng.xml | Listener1.java | Listener2.java

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">

   <listeners>
    <listener class-name="TestNGBasic.Listener2"/>
    </listeners>
    <test thread-count="5" name="Test">
    <classes>
      <class name="TestNGBasic.Listener1"/>

    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```


Problems
Javadoc
Declaration
Console
Results of running suite
<terminated> TestNG_testng.xml [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
[RemoteTestNG] detected TestNG version 7.4.0
On Test Start from Listener
Method 1 is passed
On Test Success from Listener
On Test Start from Listener
Method 1 is Failed
On Test Fail from Listener
On Test Start from Listener
On Test Skip from Listener
===
Suite
Total tests run: 3, Passes: 1, Failures: 1, Skips: 1
===

25) Selenium 25 Lecture

Capture a Screenshot when Test case will Fail by using Listener.

First class i.e. Listener3.java

i)Create Web Driver reference as a Public Static driver.

```
package TestNGBasic;
/*Date:24/09/2021
Refer Listeners3.java & Listeners4.java & Listener3&4.xml
Configuring Listeners with the Test Suite*/
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners(TestNGBasic.Listener4.class)
public class Listener3 {
	public static WebDriver driver;

	@BeforeTest
	public void initBrowser(){
		System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");
		driver=new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://opensource-demo.orangehrmlive.com/");
	}

  @Test(priority=1)
  public void login() throws InterruptedException {
	  //Enter Username

driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("A
dmin");

	  //Enter Password
```

```
driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("a
dmin1234");

        //Click on Login button

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

        Thread.sleep(4000);

        //We are Forcefully Fail this test case
        Assert.assertTrue(false);

  }

  @Test(priority=2)
  public void homepage() throws InterruptedException
  {
        System.out.println("This is second test case which is for
Homepage");
        Thread.sleep(4000);
        Assert.assertTrue(false);
  }
}
```

Program:Listener4.java

```
package TestNGBasic;
/*Date:25/09/2021
Refer Listeners3.java & Listeners4.java & Listener3&4.xml
Configuring Listeners with the Test Suite*/
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.io.FileHandler;
import org.testng.ITestListener;
import org.testng.ITestResult;
import org.testng.annotations.Test;

public class Listener4 extends Listener3 implements ITestListener {

     @Override
     public void onTestStart(ITestResult result) {
```

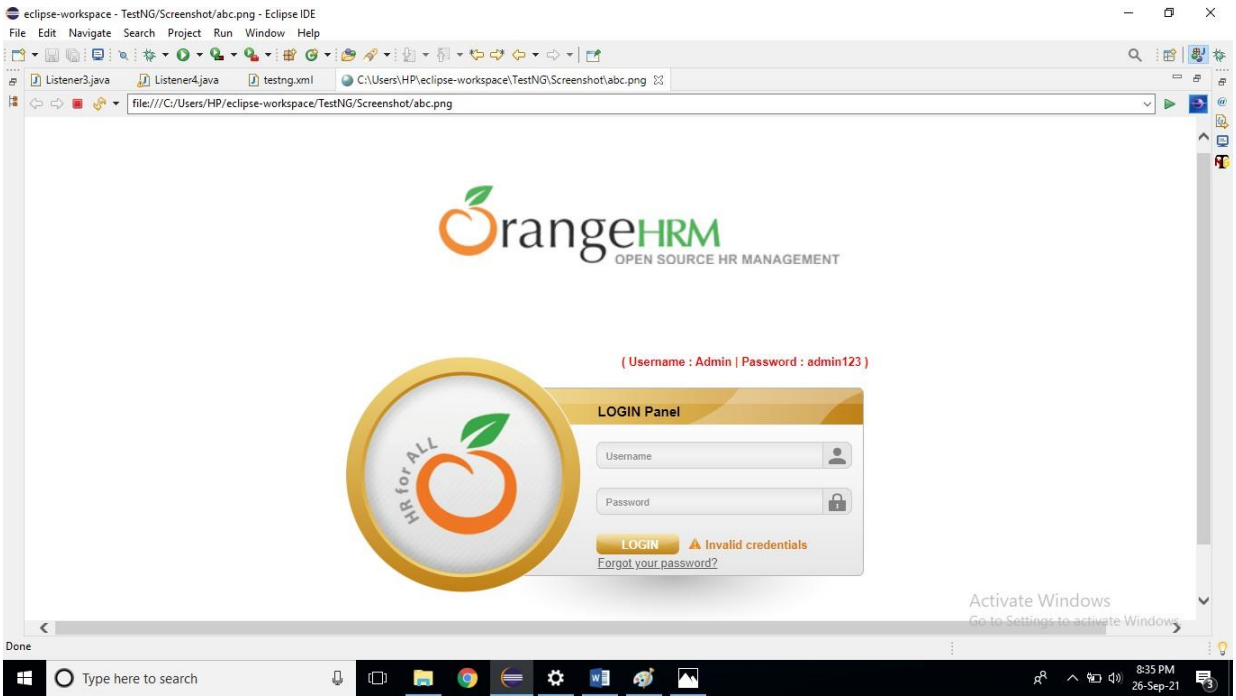
```
		System.out.println("On Test Start from Listener");
	}

	@Override
	public void onTestSuccess(ITestResult result) {
		System.out.println("On Test Success from Listener");
		TakeScreenshot(driver,result.getTestName());
	}

	public static void TakeScreenshot(WebDriver driver,String name)
	{
		TakesScreenshot screen=(TakesScreenshot) driver;
		File Source=screen.getScreenshotAs(OutputType.FILE);

		int no=1;

		//File Destination=new
File(System.getProperty("user.dir")+"Screenshot\\"+name+".png");
		File Destination=new File("C:\\Users\\HP\\eclipse-
workspace\\TestNG\\Screenshot\\abc.png");

		no++;

		try {
			FileHandler.copy(Source,Destination);
		}
		catch(IOException e){
			e.printStackTrace();
		}

		System.out.println("ScreenShot is taken");

	}

	@Override
	public void onTestFailure(ITestResult result) {
		System.out.println("On Test Fail from Listener");
		TakeScreenshot(driver,result.getTestName());
	}

	@Override
	public void onTestSkipped(ITestResult result) {
		System.out.println("on Test Skip from Listener");
	}

}
```

Listener3.java | Listener4.java | testng.xml | C:\Users\HP\eclipse-workspace\TestNG\Screenshot\abc.png

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
3  <suite name="Suite">
4   <listeners>
5    <listener class-name="TestNGBasic.Listener4"/>
6    </listeners>
7    <test thread-count="5" name="Test">
8      <classes>
9
10       <class name="TestNGBasic.Listener3"/>
11
12     </classes>
13   </test> <!-- Test -->
14 </suite> <!-- Suite -->
15
```

Javadoc | Declaration | Console | Results of running class Listener4

```
<terminated> Listener4 [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 26, 2021, 8:31:42 P
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#163
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriv
ChromeDriver was started successfully.
[1632668517.278][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 26, 2021 8:31:57 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
On Test Start from Listener
On Test Fail from Listener
ScreenShot is taken
On Test Start from Listener
This is second test case which is for Homepage
On Test Fail from Listener
ScreenShot is taken
FAILED: login
java.lang.AssertionError: expected [true] but found [false]


===============================================
    Default test
    Tests run: 2, Failures: 2, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Passes: 0, Failures: 2, Skips: 0
===============================================
```


eclipse-workspace - TestNG/Screenshot/abc.png - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Listener3.java
Listener4.java
testng.xml
C:\Users\HP\eclipse-workspace\TestNG\Screenshot\abc.png
file:///C:/Users/HP/eclipse-workspace/TestNG/Screenshot/abc.png
OrangeHRM
OPEN SOURCE HR MANAGEMENT
(Username : Admin | Password : admin123)
LOGIN Panel
Username
Password
LOGIN
Invalid credentials
Forgot your password?
HR for ALL
Activate Windows
Go to Settings to activate Windows.
Done
Type here to search
8:35 PM
26-Sep-21

26) Selenium 26 Lecture

.Properties File:

.properties files are mainly used in Java programs to maintain project configuration data, database config or project settings, etc. You can easily read properties from some file using an object of type Properties. This is a utility provided by Java itself.

In simple words, a Properties file is a file with **.properties** an extension. Each parameter in the properties file is stored as a pair of strings, in key-value pair format(key = value), where each key is on one line. They can also be used for storing strings for Internationalization and localization.

Since the **.properties** file is a java library from **.util** the package we do not need any dependencies. but because I'm using TestNG, Selenium and doing browser action let's start with a maven project.

Create a file with a .properties extension:

i) Right click on Project >> New >> File >> Give file name as config.properties

The value in .properties file is in the String format.

Steps:

1) User property prop variable

2) Declare the method for read Data from Property file

3) Give the path of config.properties file by Right click on it >> Properties >> Select the Locations & Copy it.

4) For reading the properties of Config file. Create object of properties class.

5) Declare File Input Stream class for reding data from file.

6) Load the Properties file by using prop.load() method.

7) Read the properties from file by using prop.getproperty() method.

8) Print the value of key on the console.

9) Call redProperty() method from main () method.

10) Locate the Username element by using getproperty method & passing key of email in config.peoperties file as input.

11) Locate the Password element by using getproperty method & passing key of pass in config.peoperties file as input.

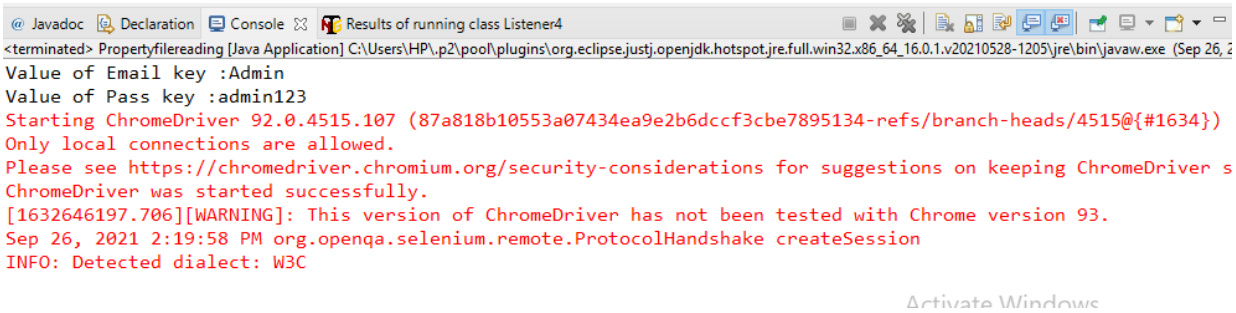
Program:Reading data from .properties file & pass that dat to web page.

```
package TestNGBasic;

/*Date:26/09/2021
 Program for reading data from .properties file*/
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;

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

public class Propertyfilereading {
	public static Properties prop;

	public static void readProperty() throws IOException
	{
		String path="C:\\Users\\HP\\eclipse-
workspace\\TestNG\\config.properties";

		prop=new Properties();

		FileInputStream fis=new FileInputStream(path);

		prop.load(fis);

		String AdminEmail=prop.getProperty("email");

		String AdminPass=prop.getProperty("pass");

		System.out.println("Value of Email key :"+AdminEmail);

		System.out.println("Value of Pass key :"+AdminPass);
	}
```

```
	public static void main(String[] args) throws IOException,
InterruptedException {

		readProperty();

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

		WebDriver driver=new ChromeDriver();

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

		driver.get("https://opensource-demo.orangehrmlive.com/");


driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys(pr
op.getProperty("email"));
		Thread.sleep(4000);


driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys(pr
op.getProperty("pass"));
		Thread.sleep(4000);


driver.findElement(By.xpath("//input[@id='btnLogin']")).click();
		Thread.sleep(4000);

		driver.findElement(By.id("welcome")).click();

		Thread.sleep(4000);

		driver.close();



	}

}
```

config.properties

```
email=Admin
pass=admin123
```



OrangeHRM
OPEN SOURCE HR MANAGEMENT

Activ

Waits in Selenium:

Synchronization:

Matching of selenium Test script running with applicaton speed on webpage is known as "Synchronization".

To achieve synchronization following methods we use.

1) Thread.sleep(15000);//**Hard Waits or static waits**

2) There are 3 types of Waits in Selenium which is known as **Soft waits or dynamic waits**

1) Implicit wait

2) Explicit wait

3) Fluent wait

Wait -->Interface

^

Implements

FluentWait Class

extends

Webdriver Wait Class

Why Do We Need Waits In Selenium?

Most of the web applications are developed using Ajax and Javascript. When a page is loaded by the browser the elements which we want to interact with may load at different time intervals.

Not only it makes this difficult to identify the element but also if the element is not located it will throw an "**ElementNotVisibleException**" exception. Using Selenium Waits, we can resolve this problem.

Let's consider a scenario where we have to use both implicit and explicit waits in our test. Assume that implicit wait time is set to 20 seconds and explicit wait time is set to 10 seconds.

Suppose we are trying to find an element which has some **"ExpectedConditions** "(Explicit Wait), If the element is not located within the time frame defined by the Explicit wait(10 Seconds), It

will use the time frame defined by implicit wait(20 seconds) before throwing an "**ElementNotVisibleException**".

1) 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". **The default setting is 0.** Once we set the time, the web driver will wait for the element for that time before throwing an exception.
syntax:

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

Implicit wait will accept 2 parameters, the first parameter will accept the time as an integer value and the second parameter will accept the time measurement in terms of SECONDS, MINUTES, MILISECOND, MICROSECONDS, NANOSECONDS, DAYS, HOURS, etc.

Note: Implicit Wait is globally applied. It is readily available for driver instance. It also means that if the driver is having interaction with One thousand elements in the meantime, the implicit Wait will be applicable only for those 1000 elements. It cannot go beyond that.

Program:Implicit waits

```
package TestNGBasic;

/*Date:26/09/2021
 Program for Implicit wait*/

import org.testng.annotations.Test;

import java.util.concurrent.TimeUnit;

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

public class WaitsImplicit {

	@Test
	public void login() {
```

```
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

	WebDriver driver=new ChromeDriver();

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

	driver.get("https://opensource-demo.orangehrmlive.com/");

	driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);


driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("A
dmin");


driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("a
dmin123");

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

	driver.findElement(By.id("welcome")).click();
	  }


}
```

As in above example there are 4 Elements which need to be locate i.e.

Username, Password, Login, Logout.

Implicit wait is 5 seconds i.e. applicable for all elements.

If Username Element finds in 3 seconds then 2 seconds save & imeediately it will find second element i.e. Password this is advantage of Implicit wait.

2) Explicit Waits:

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 elements.
Once we declare explicit wait we have to use "**ExpectedConditions**" or we can configure how frequently we want to check the condition using **Fluent Wait**. These days while implementing we are using **Thread.Sleep()** generally it is not recommended to use

Condition 1:

Suppose you have a web page consisting of a login form that takes input and loads the Home or Main page content. This page is dynamic because of the time constraints and network frequency, sometimes taking 10 seconds or maybe 15 seconds to load completely. Explicit Wait comes in handy in such cases and allows you to wait until the page is not present to display.

Condition 2:

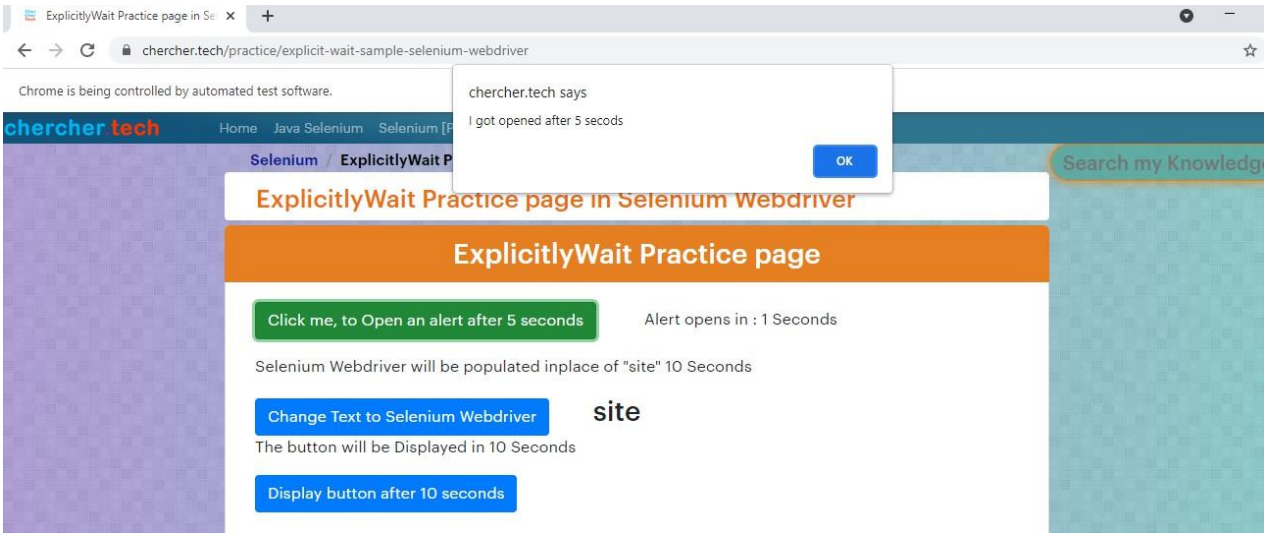
Consider that you are working on an application that is travel themed and users fill the web form and submit it using submit button. Now, you might need to wait until and unless the specific data is not displayed. In such a case, Explicit Wait becomes helpful by waiting until a specific period for the set of elements that are not displayed yet.

Program:Explicit wait for Alert

```
package TestNGBasic;
/*Date:26/09/2021
 Program of Explicit wait for Alert*/
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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class WaitsExplicit {
```

```
	public static WebDriver driver;

	@BeforeTest
	public void BeforeAlert()
	{

	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

			driver=new ChromeDriver();

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

			driver.get("https://chercher.tech/practice/explicit-wait-sample-selenium-webdriver");

	}
   @Test(priority=1)
   public void Alert() {

		driver.findElement(By.id("alert")).click();

		WebDriverWait wb=new WebDriverWait(driver,12);

		wb.until(ExpectedConditions.alertIsPresent());

		driver.switchTo().alert();
   }
}
```

```
Javadoc  Declaration  Console  Results of running class WaitsExplicit
<terminated> WaitsExplicit [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 26, 2021, 3:26:44 PM
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634})
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver s
ChromeDriver was started successfully.
[1632650214.489][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 26, 2021 3:26:55 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: Alert

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```



As shown in above example, by clicking on Alert button Alert pop up will going to open in 5 seconds after clicking.

But by using Explicit waits it will checks the condition after 1 second i.e. IsAlertPersent() which returns true or False. If It finds element in 2 seconds then it will perform next action immediately not wait for total declared time.

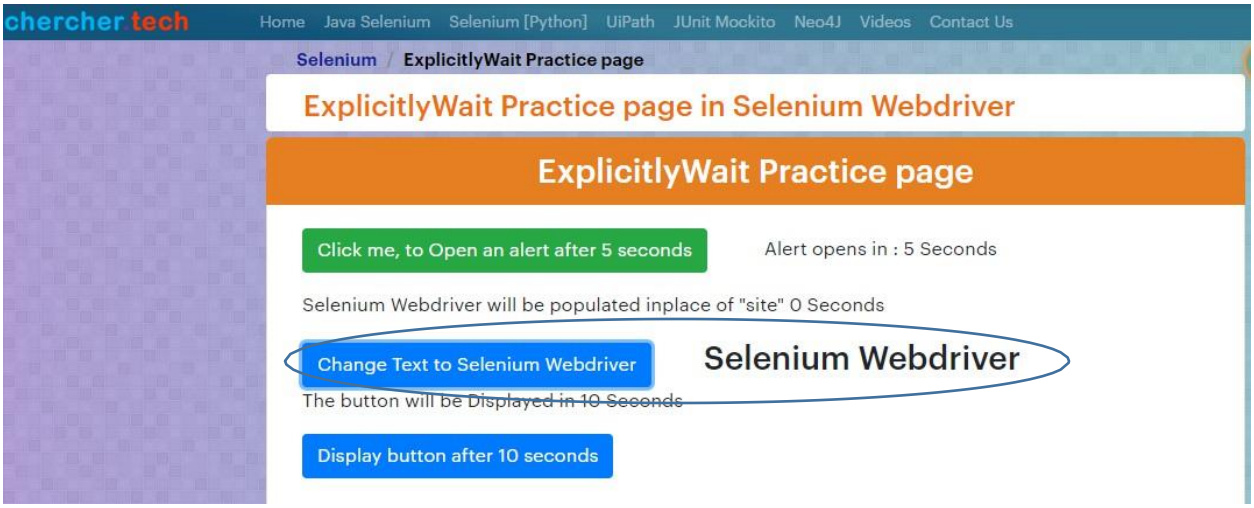
We can use Implicit wait & Explicit wait together.

Program:Explicit wait for Hidden Text.

```
package TestNGBasic;
/*Date:26/09/2021
Program of Explicit wait for Display of Hidden Text*/
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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class WaitsExplicit2 {
     public static WebDriver driver;
```

```
    @BeforeTest
    public void BeforeText()
    {

      System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

            driver=new ChromeDriver();

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

            driver.get("https://chercher.tech/practice/explicit-wait-
sample-selenium-webdriver");

    }
      @Test
  public void Text() {
            driver.findElement(By.id("populate-text")).click();

            WebElement text=driver.findElement(By.id("populate-text"));

            WebDriverWait wb=new WebDriverWait(driver,12);

            wb.until(ExpectedConditions.textToBePresentInElement(text,
"Selenium Webdriver"));


  }
}
```

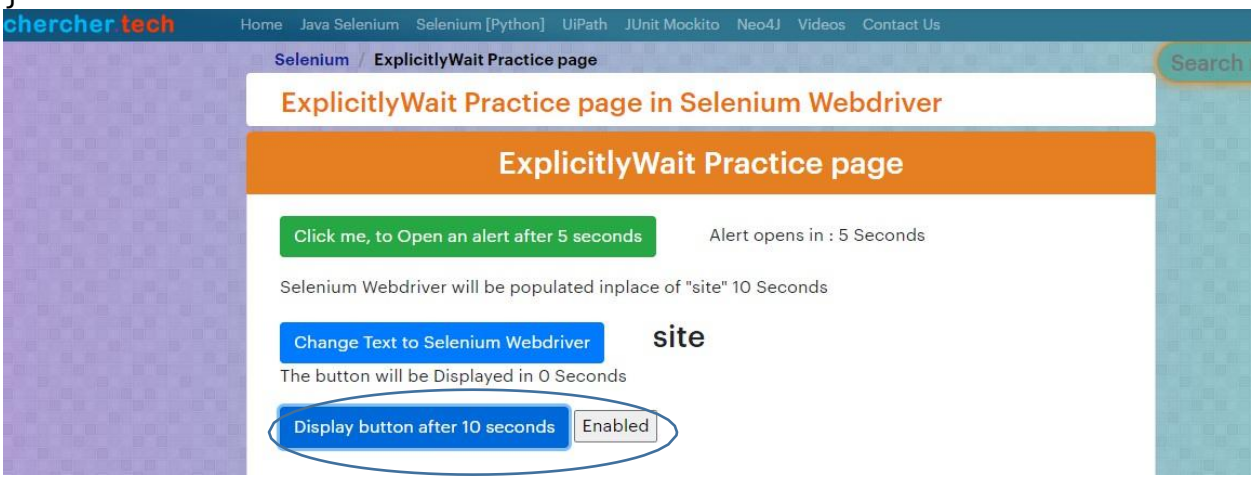
```
Javadoc  Declaration  Console  Results of running class WaitsExplicit2
<terminated> WaitsExplicit2 [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 27, 2021, 2:08:44 PM
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 93.0.4577.63 (ff5c0da2ec0adeaed5550e6c7e98417dac77d98a-refs/branch-heads/4577@{#1135}) o
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver s
ChromeDriver was started successfully.
Sep 27, 2021 2:08:49 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: Text

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

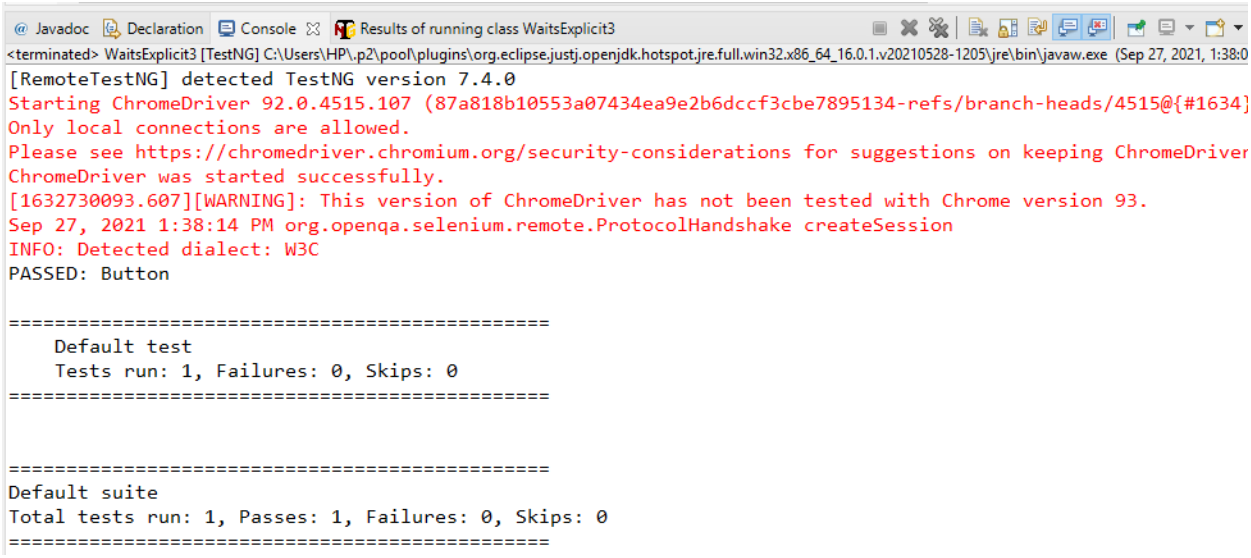


Program:Explicit wait for hidden button

```
package TestNGBasic;
/*Date:26/09/2021

 Program of Explicit wait for Display of Hidden button*/
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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class WaitsExplicit3 {

	public static WebDriver driver;

	@BeforeTest
	public void BeforeButton()
	{

	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
```

```
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

            driver=new ChromeDriver();

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

            driver.get("https://chercher.tech/practice/explicit-wait-
sample-selenium-webdriver");

      }
   @Test(priority=1)
   public void Button() {
         driver.findElement(By.id("display-other-button")).click();

         WebElement button=driver.findElement(By.id("hidden"));

         WebDriverWait wb=new WebDriverWait(driver,12);

         wb.until(ExpectedConditions.elementToBeClickable(button));

         //wb.until(ExpectedConditions.visibilityOf(button));

         }
}
```




Javadoc
Declaration
Console
Results of running class WaitsExplicit3
<terminated> WaitsExplicit3 [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 27, 2021, 1:38:0
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634}
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver
ChromeDriver was started successfully.
[1632730093.607][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 27, 2021 1:38:14 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: Button
===
Default test
Tests run: 1, Failures: 0, Skips: 0
===
===
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===

27) Selenium 27 Lecture

3) 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.
Polling Time/Frequency: Setting up a repeat cycle with the time frame to verify/check the condition at the regular interval of time

Let's consider a scenario where an element is loaded at different intervals of time. The element might load within 10 seconds, 20 seconds or even more then that if we declare an explicit wait of 20 seconds. It will wait till the specified time before throwing an exception. In such scenarios, the fluent wait is the ideal wait to use as this will try to find the element at different frequency until it finds it or the final timer runs out.

Fluent Wait syntax:

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

It provides the flexibility to the user to define the polling frequency in the time units.

Steps:

i) We will declare Fluent wait class which is of Web Driver type & Declare driver as a argument within it.

```
FluentWait <WebDriver> fwait=new FluentWait(driver);
```

ii) We will modify Polling time by

a) Give default timeout by

```
fwait.withTimeout(15, TimeUnit.SECONDS)
```

b) Give Polling time in Miliseconds means at that time of miliseconds it will check every time where element is found or not.

```
pollingEvery(10, TimeUnit.MILLISECONDS)
```

c) Ignore the exception of NoSuchElement Exception by using ignoring method as

```
ignoring(NoSuchElementException.class)
```

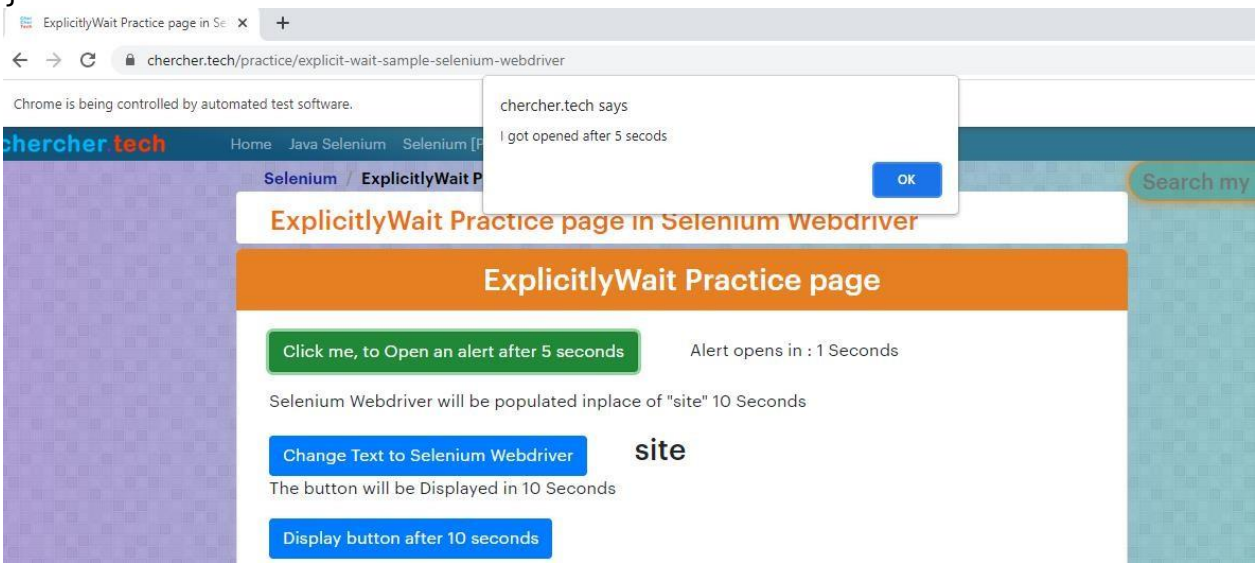
d) Then use until method & Expected condition as an argument within it.

```
fwait.until(ExpectedConditions.alertIsPresent());
```

e) Accept the alert.

```
package TestNGBasic;
/*Date:27/09/2021
 Program of Fluent wait for Alert*/

import java.util.NoSuchElementException;
import java.util.concurrent.TimeUnit;

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

public class WaitFluent {
public static WebDriver driver;
//public Fluent fwait;

      @BeforeTest
      public void BeforeAlert()
      {

      System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desk
top\\Dhananjay Deshmukh\\Software
Testing\\Selenium\\chromedriver.exe");

            driver=new ChromeDriver();

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

            driver.get("https://chercher.tech/practice/explicit-wait-
sample-selenium-webdriver");

      }
   @Test(priority=1)
   public void Alert() {
```

```
        driver.findElement(By.id("alert")).click();

        FluentWait <WebDriver> fwait=new FluentWait(driver);

        fwait.withTimeout(15, TimeUnit.SECONDS).pollingEvery(10,
TimeUnit.MILLISECONDS).ignoring(NoSuchElementException.class);

        fwait.until(ExpectedConditions.alertIsPresent());

        driver.switchTo().alert().accept();
    }

}
```




```
Javadoc  Declaration  Console  Results of running class WaitFluent
<terminated> WaitFluent [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (Sep 27, 2021, 1:03:05 PM – 1:03:35 PM)
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 92.0.4515.107 (87a818b10553a07434ea9e2b6dccf3cbe7895134-refs/branch-heads/4515@{#1634}) on port 25924
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1632727995.216][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
Sep 27, 2021 1:03:18 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: Alert

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

Page Object Model:

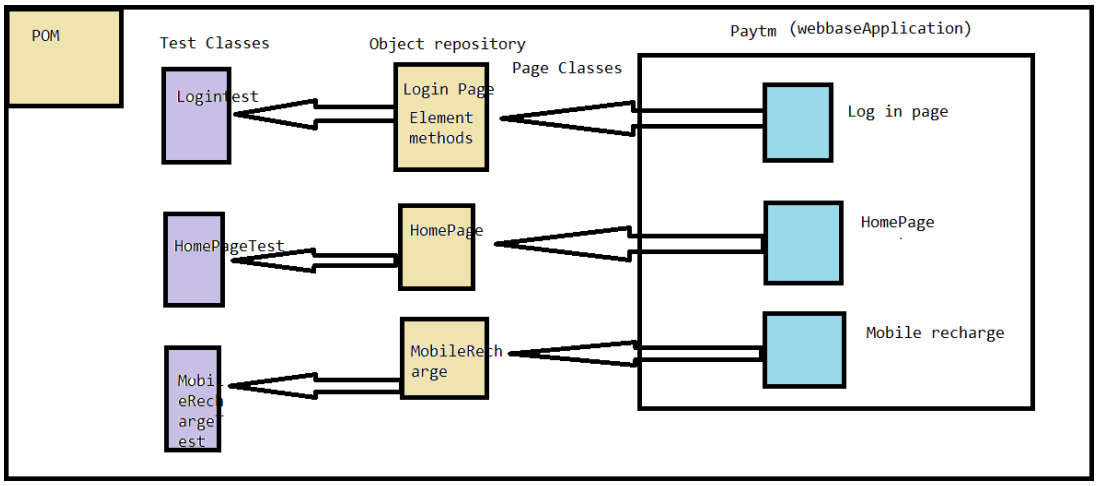
It IS A design pattern while writing the automation script which creates seperate object repository for web element.

This design pattern helps to reduce the code complication and improves test mainatanability.

As per this for each webpage there is corressponding

1) Page class

2) Test class.



Page Object Model (POM) is a design pattern, popularly used in test automation that creates Object Repository for web UI elements. The advantage of the model is that it reduces code duplication and improves test maintenance.
Under this model, for each web page in the application, there should be a corresponding Page Class. This Page class will identify the WebElements of that web page and also contains Page methods which perform operations on those WebElements. Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay().

Advantages of POM

1. Page Object Design Pattern says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. The Second benefit is the object repository is independent of test cases, so we can use the same object repository for a different purpose with different tools. For example, we can integrate Page Object Model in Selenium with TestNG/JUnit for functional Testing and at the same time with JBehave/Cucumber for acceptance testing.

3. Code becomes less and optimized because of the reusable page methods in the POM classes.
4. Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like ‘gotoHomePage()’.

28) Selenium 28 Lecture

Program:Login Page & Login Test Program by using POM(Page object model)

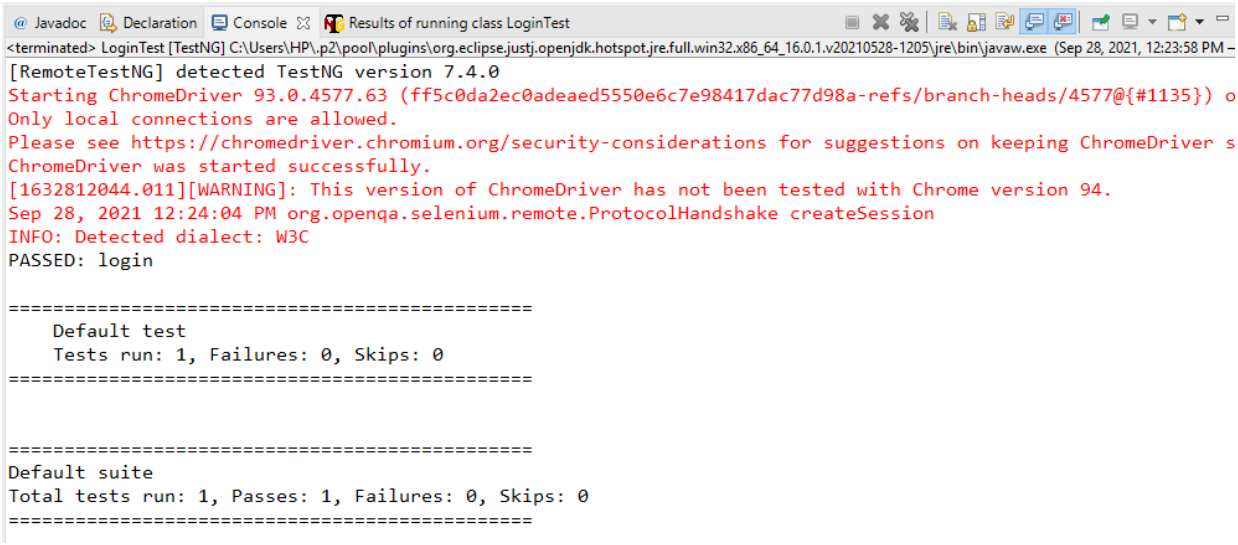
LoginPage.java

```
package POM;
/*Date:28/09/2021
Program Of Login page using POM in which Encapsulation is performed
Refer LoginPage.java & LoginTest.java*/
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class LoginPage {

      private WebElement username;
      private WebElement password;
      private WebElement button1;

      public LoginPage(WebDriver driver)
    {
      username=driver.findElement(By.id("txtUsername"));
      password=driver.findElement(By.id("txtPassword"));
      button1=driver.findElement(By.id("btnLogin"));

    }

    public void Login()
    {
      username.sendKeys("Admin");
      password.sendKeys("admin123");
      button1.click();

    }
}
```

LoginTest.java

```
package POM;
/*Date:28/09/2021
Program Of Login Test using POM in which Encapsulation is performed
Refer LoginPage.java & LoginTest.java*/
import org.testng.annotations.Test;
import org.testng.annotations.BeforeClass;
import java.util.concurrent.TimeUnit;
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;

public class LoginTest {

	public WebDriver driver;

  @BeforeClass
  public void InitBrowser() {


System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

	driver=new ChromeDriver();

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

	driver.get("https://opensource-demo.orangehrmlive.com/");

	driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

  }
  @Test
  public void login() {
		LoginPage lp=new LoginPage(driver);
		lp.Login();
  }

  @AfterClass
  public void afterClass() {
		driver.close();
  }

}
```

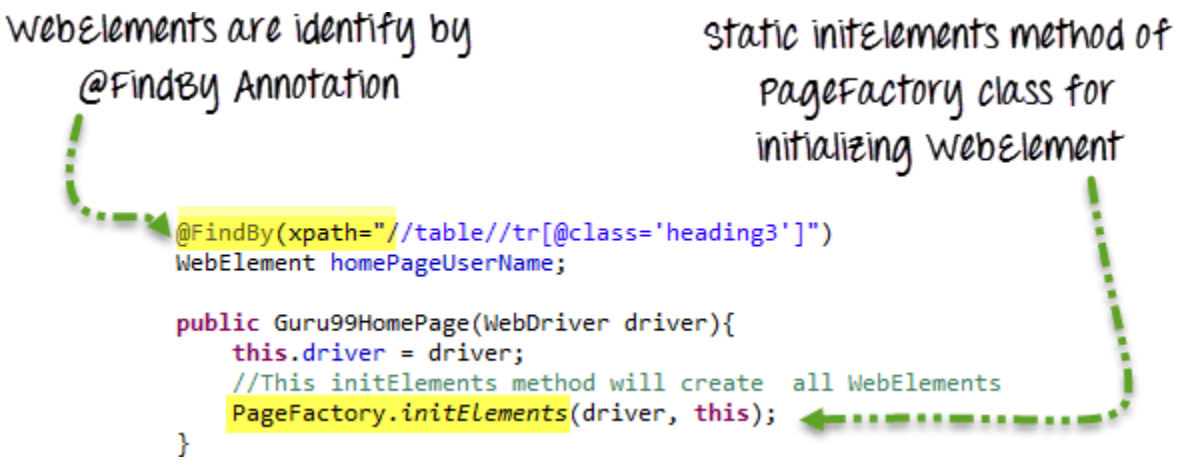

Javadoc
Declaration
Console
Results of running class LoginTest
<terminated> LoginTest [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 28, 2021, 12:23:58 PM –
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 93.0.4577.63 (ff5c0da2ec0adeaed5550e6c7e98417dac77d98a-refs/branch-heads/4577@{#1135}) o
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver s
ChromeDriver was started successfully.
[1632812044.011][WARNING]: This version of ChromeDriver has not been tested with Chrome version 94.
Sep 28, 2021 12:24:04 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: login
===
Default test
Tests run: 1, Failures: 0, Skips: 0
===
===
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===

Page Factory in Selenium:

Whenever we use POM there are some elements which are in hidden format on a web page. So to find those elements & initialize them early we require Page Factory.

@FindBy Annotation is used to find web elements early by using PageFacory.initelements() method.

Page Factory in Selenium is an inbuilt Page Object Model framework concept for Selenium WebDriver but it is very optimized. **It is used for Early initialization of Page objects or to instantiate the Page object itself.** It is also used to initialize Page class elements without using "FindElement/s."
Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of class PageFactory in Selenium, we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements



@FindBy **can accept** tagName, partialLinkText, name, linkText, id, css, className, xpath **as attributes.**

PageFactory.InitElements() method initialize all data members of @FindBy Element i.e. Early initialization.

Program:Login Page & Login Test Program by using POM Page Factory.

LoginPage2.java

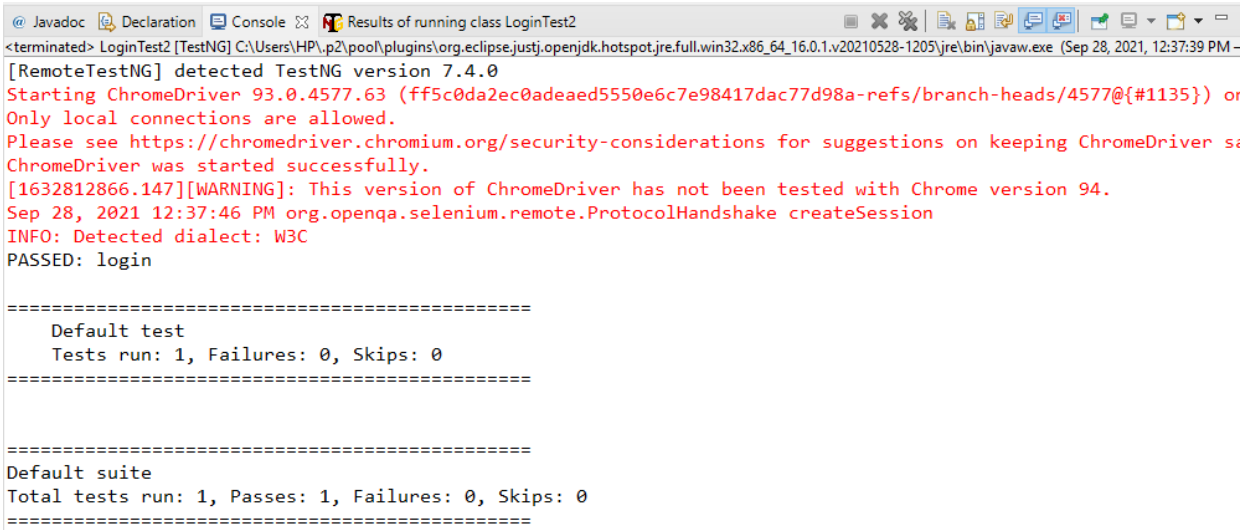
```
package POM;
/*Date:28/09/2021
Program Of Login page using POM Page Factory in which Encapsulation is
performed
```

```java
Refer LoginPage2.java & LoginTest2.java
Generally we perform POM by using Page Factory*/
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class LoginPage2 {

      @FindBy(id="txtUsername")
      private WebElement username;

      @FindBy(id="txtPassword")
      private WebElement password;

      @FindBy(id="btnLogin")
      private WebElement button1;

      public LoginPage2(WebDriver driver)
    {
      PageFactory.initElements(driver, this);
    }

    public void Login()
    {
      username.sendKeys("Admin");
      password.sendKeys("admin123");
      button1.click();

    }
}
```

LoginTest2.java

```java
package POM;
/*Date:28/09/2021
Program Of Login Test using POM Page Factory in which Encapsulation is
performed
Refer LoginPage.java & LoginTest.java*/
import org.testng.annotations.Test;
import org.testng.annotations.BeforeClass;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;

public class LoginTest2 {

	public WebDriver driver;

  @BeforeClass
  public void InitBrowser() {


System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");

	driver=new ChromeDriver();

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

	driver.get("https://opensource-demo.orangehrmlive.com/");

	driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

  }
  @Test
  public void login() {
	LoginPage2 lp=new LoginPage2(driver);
	lp.Login();
  }

  @AfterClass
  public void afterClass() {
	driver.close();
  }

}
```


Javadoc
Declaration
Console
Results of running class LoginTest2
<terminated> LoginTest2 [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Sep 28, 2021, 12:37:39 PM –
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 93.0.4577.63 (ff5c0da2ec0adeaed5550e6c7e98417dac77d98a-refs/branch-heads/4577@{#1135}) or
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver sa
ChromeDriver was started successfully.
[1632812866.147][WARNING]: This version of ChromeDriver has not been tested with Chrome version 94.
Sep 28, 2021 12:37:46 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: login
===
Default test
Tests run: 1, Failures: 0, Skips: 0
===
===
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===

29) Selenium 29 Lecture

Data Provider in TestNG

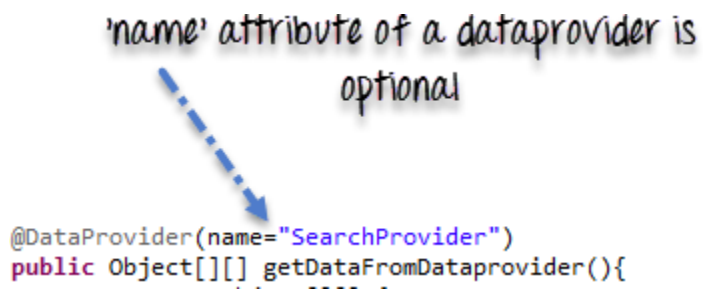
Data Provider in TestNG is a method used when a user needs to pass complex parameters. Complex Parameters need to be created from Java such as complex objects, objects from property files or from a database can be passed by the data provider method. The method is annotated by @DataProvider and it returns an array of objects.

Parameters using Dataprovider

@Parameters annotation is easy but to test with multiple sets of data we need to use Data Provider.

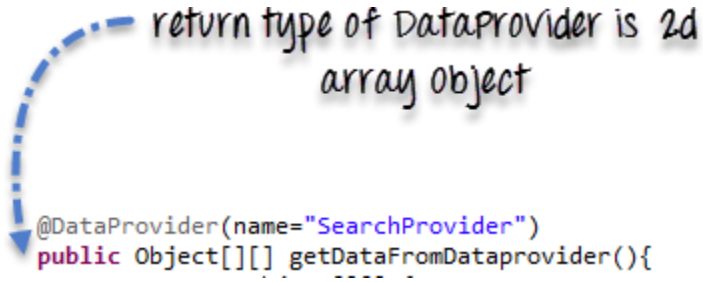
To fill thousand's of web forms using our testing framework we need a different methodology which can give us a very large dataset in a single execution flow.

This data driven concept is achieved by **@DataProvider** annotation in TestNG.



It has only one **attribute 'name'**. If you do not specify the name attribute then the DataProvider's name will be same as the corresponding method name.

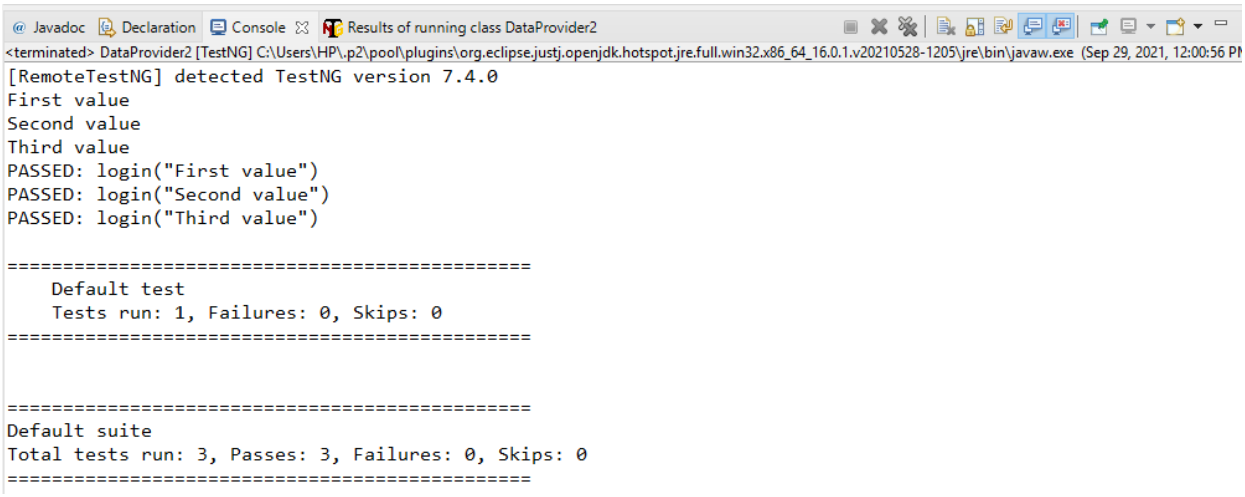
Data provider returns **a two-dimensional JAVA object** to the test method and the test method, will invoke M times in a M*N type of object array. For example, if the DataProvider returns an array of 2*3 objects, the corresponding testcase will be invoked 2 times with 3 parameters each time.



Data provider is used to pass multiple sets of data.

Program1:Simple use of Data provider to provide String type of data to same test case.

```
package POM;
/*Date:29/09/2021
Program Of Login Test using Data Provider.Refer DataProvider2.java*/
import org.testng.annotations.Test;
import org.testng.annotations.DataProvider;

public class DataProvider2 {
  @Test(dataProvider="TestData")
  public void login(String value) {

        System.out.println(value);
  }

  @DataProvider(name="TestData")
  public String[] getData() {

        String[] data=new String[3];
        data[0]="First value";
        data[1]="Second value";
        data[2]="Third value";
        return data;
        }
}
```



As Shown in above program we have written only one test case but As Shown in above output Number of Passed test cases are 3.

It executes the test case based on the data provided.

For every separated data it is marking or considering as a separate test case.

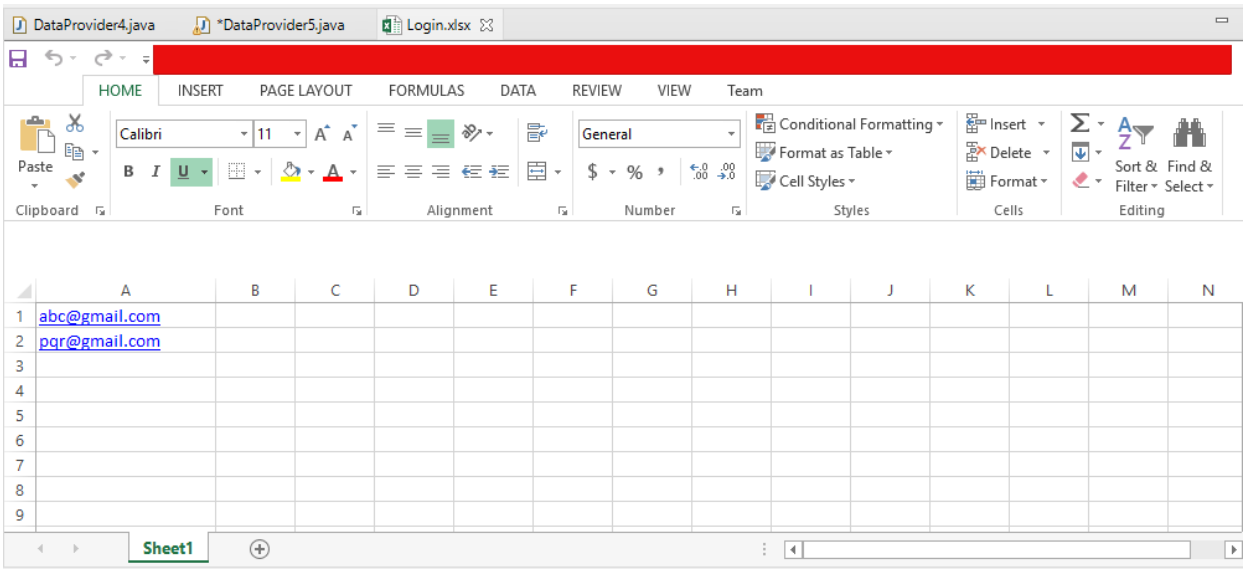
We will use Data provide when we have different data & same test case then we will provide data using Data Provider.

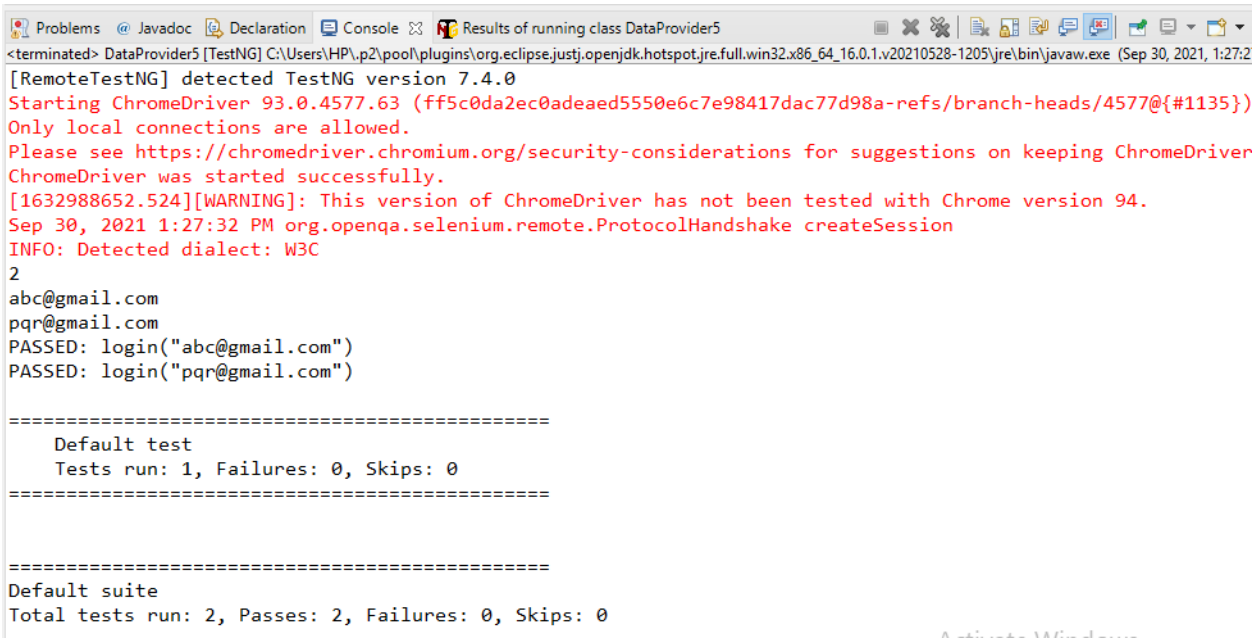
30) Selenium 30 Lecture

Program1: Read data from Excel file & Print it on Console by using Data Provider.

```
package POM;
/*Date:30/09/2021
 Program to read data from Excel file & print it on Console by using
Data Provider*/
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DataProvider5 {
	WebDriver driver;

	@BeforeClass
	  public void initbrowser()
	{
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
	    driver = new ChromeDriver();
	    driver.manage().window().maximize();
	    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
	    driver.get("https://opensource-demo.orangehrmlive.com/");
	   }

	@Test(dataProvider = "TestData")
	public void login(String value)
	{

		System.out.println(value);
	}


	 @DataProvider(name="TestData")
	  public String[] getData() throws IOException
```

```
	{

		String path="C:\\Users\\HP\\eclipse-
workspace\\TestNG\\Screenshot\\Login.xlsx";
		File file=new File(path);
		FileInputStream fs=new FileInputStream(file);
		XSSFWorkbook wb=new XSSFWorkbook(fs);
		XSSFSheet sheet =wb.getSheetAt(0);

		int row=sheet.getPhysicalNumberOfRows();
		System.out.println(row);
	      String  [] data=new String  [row];

	      for(int i=0; i<row;i++)
		{
		String
data1=sheet.getRow(i).getCell(0).getStringCellValue();
		    data [i]=data1;

		}
		return data;

  }
}
```



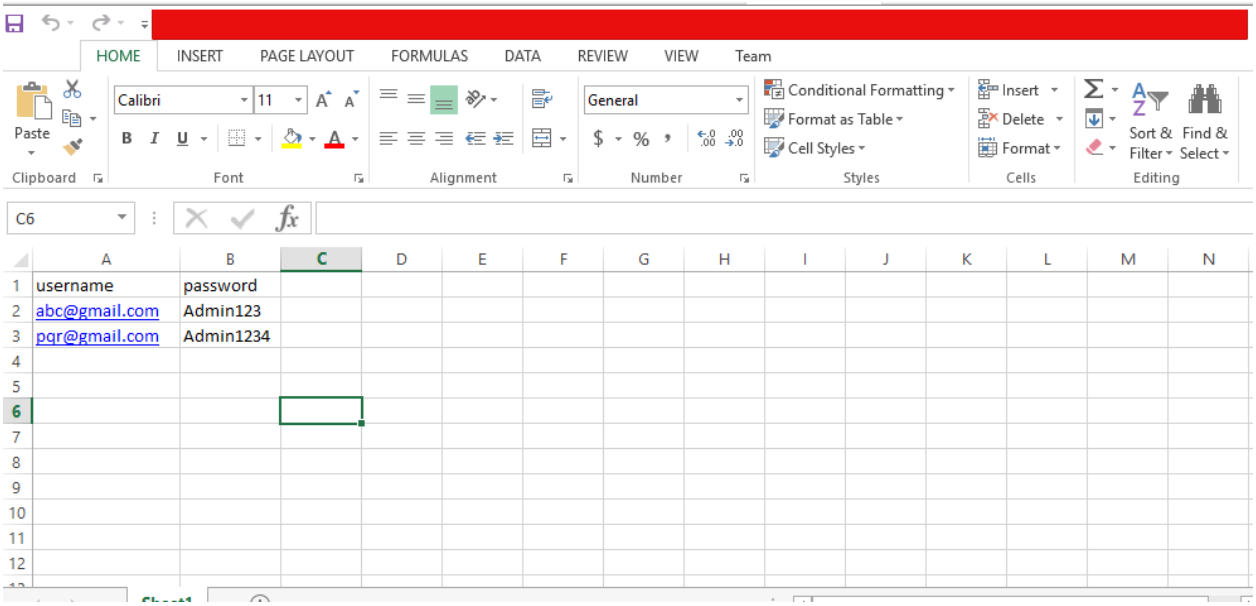


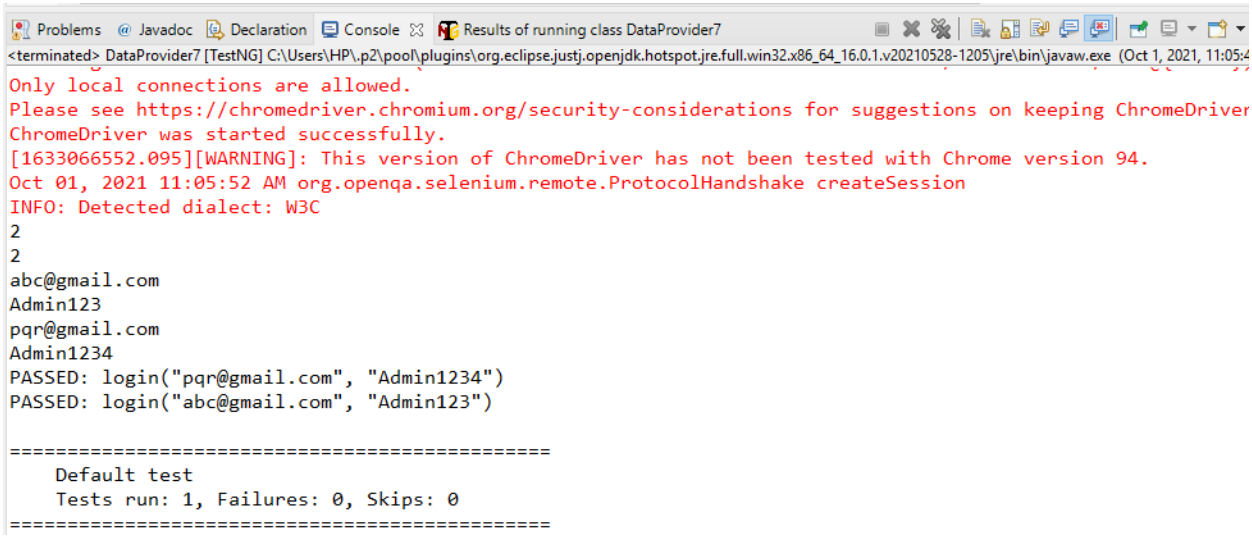
Program2: Read the multiple data from Excel file like Username & Password & Print it on Console using Data Provider.

```
package POM;
/*Date:30/09/2021
 Program to read data from Excel file & print it on Console by using Data Provider
 Refer DataProvider7.java & Login.xlsx*/
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DataProvider7 {
	WebDriver driver;
```

```
      @BeforeClass
        public void initbrowser()
      {
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
         driver = new ChromeDriver();
         driver.manage().window().maximize();
         driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
         driver.get("https://opensource-demo.orangehrmlive.com/");
        }

      @Test(dataProvider = "TestData")
      public void login(String username,String password)
      {

            System.out.println(username);
            System.out.println(password);
      }


       @DataProvider(name="TestData")
        public String[][] getData() throws IOException
       {

              String path="C:\\Users\\HP\\eclipse-
workspace\\TestNG\\Screenshot\\Login2.xlsx";
              File file=new File(path);
              FileInputStream fs=new FileInputStream(file);
              XSSFWorkbook wb=new XSSFWorkbook(fs);
              XSSFSheet sheet =wb.getSheetAt(0);

              int row=sheet.getLastRowNum();
              int col=sheet.getRow(0).getLastCellNum();
              System.out.println(row);
              System.out.println(col);
             String  [][] data=new String  [row][col];

             for(int i=0; i<row;i++)
              {
              for(int j=0;j<col;j++)
              {

              data
[i][j]=sheet.getRow(i+1).getCell(j).getStringCellValue();
```

```
            }
            }
            return data;

    }
        @AfterClass
        public void browserClose()
        {
              driver.close();
        }
}
```





Program3: Read the multiple data from Excel file & print that data into console & pass it to the web page by using Data provider.

DataProvider8.java

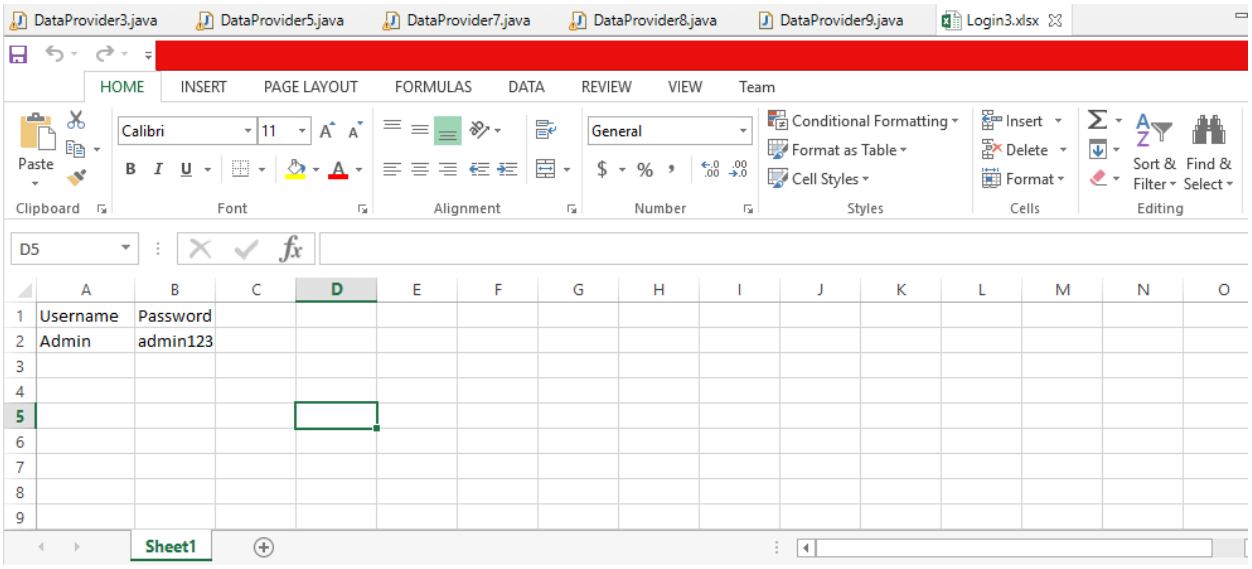
```
package POM;
/*Date:30/09/2021
Program to read data from Excel file & print it on console & pass that
data to web Page.
Refer DataProvider8.java & DataProvider9.java*/
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DataProvider8 {
            static String path="C:\\Users\\HP\\eclipse-
workspace\\TestNG\\Screenshot\\Login3.xlsx";

              public static String[][] getData() throws IOException
      {

              File file=new File(path);
              FileInputStream fs=new FileInputStream(file);
              XSSFWorkbook wb=new XSSFWorkbook(fs);
              XSSFSheet sheet =wb.getSheetAt(0);

              int row=sheet.getLastRowNum();
              int col=sheet.getRow(0).getLastCellNum();
              System.out.println(row);
              System.out.println(col);
             String  [][] data=new String  [row][col];

             for(int i=0; i<row;i++)
              {
              for(int j=0;j<col;j++)
              {
```

```
                data
[i][j]=sheet.getRow(i+1).getCell(j).getStringCellValue();

                }
                }
                return data;

  }

}
```

DataProvider9.java

```
package POM;
/*Date:30/09/2021
 Program to read data from Excel file & print it on console & pass
that data to web Page.
 Refer DataProvider8.java & DataProvider9.java*/
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DataProvider9 {
  WebDriver driver;

  @BeforeClass
  public void initbrowser() {

System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Desktop\\
Dhananjay Deshmukh\\Software Testing\\Selenium\\chromedriver.exe");
      driver = new ChromeDriver();
      driver.manage().window().maximize();
      driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      driver.get("https://opensource-demo.orangehrmlive.com/");
  }

  @Test(dataProvider="orange")
  public void testLogin(String username,String password)
```

```
    {
          System.out.println(username);
          System.out.println(password);
          driver.findElement(By.id("txtUsername")).sendKeys(username);
          driver.findElement(By.id("txtPassword")).sendKeys(password);

          WebElement button=driver.findElement(By.id("btnLogin"));
          button.click();
    }

    @DataProvider(name="orange")
    public String[][] hrm() throws IOException
    {
          String data [][]=DataProvider8.getData();
          try {
               data=DataProvider8.getData();
        } catch (IOException e) {

              e.printStackTrace();
        }
          return data;
    }
}
```



Problems Javadoc Declaration Console Results of running class DataProvider9

<terminated> DataProvider9 [TestNG] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (Oct 1, 2021, 11:36:44

```
[RemoteTestNG] detected TestNG version 7.4.0
Starting ChromeDriver 93.0.4577.63 (ff5c0da2ec0adeaed5550e6c7e98417dac77d98a-refs/branch-heads/4577@{#1135})
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver
ChromeDriver was started successfully.
[1633068410.714][WARNING]: This version of ChromeDriver has not been tested with Chrome version 94.
Oct 01, 2021 11:36:50 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
1
2
1
2
Admin
admin123
PASSED: testLogin("Admin", "admin123")

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```