BY: Kanchana

Table of Contents

Before we start with the Selenium make sure you are perfect with the following java concepts:

Oops concepts –

1. Encapsulation: Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.
2. Polymorphism: **Polymorphism in Java** is a concept by which we can perform a single action in different ways
   There are two types polymorphism
   1. compile-time polymorphism
   2. Runtime polymorphism.

   We can perform polymorphism in java by method overloading and method overriding.

3. Inheritance : It is a mechanism in which one object acquires all the properties and behaviors of a parent object
   The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class
4. Abstract class: **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

Selenium: It is a free and open source web application tool that automates the web browsers. It also provides a interface to write test scripts in different programming languages like Java, Ruby, PHP, Perl, Python etc.

Download Selenium: Always download the latest version of Selenium

URL: http://docs.seleniumhq.org/download/

File: selenium-server-standalone-latest version of jar file

Flavors of Selenium:

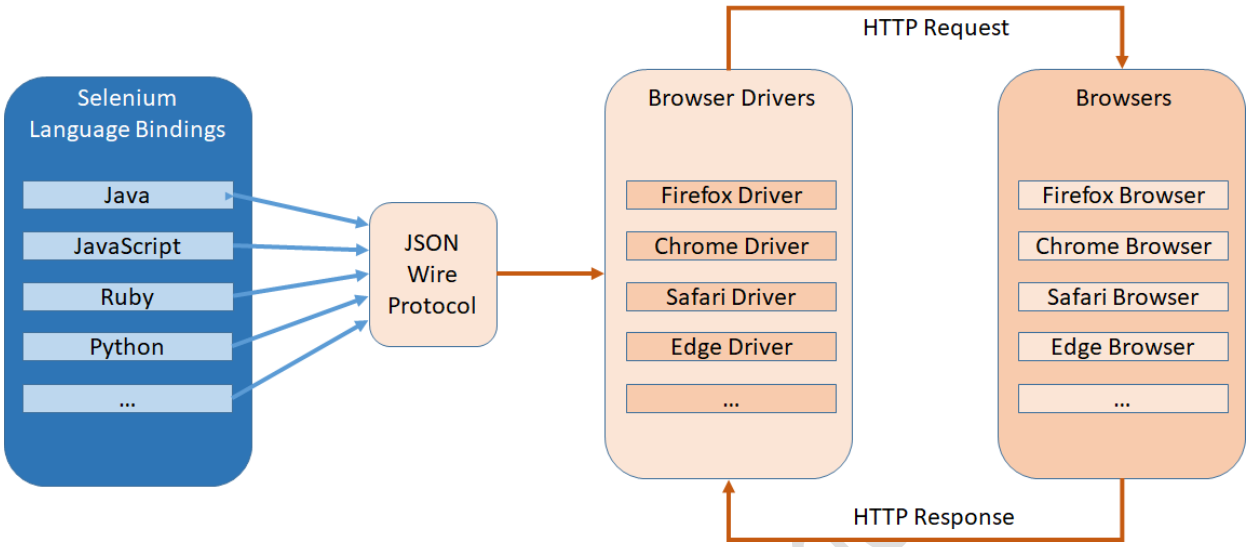
1. Selenium Core
2. Selenium RC
3. Selenium Webdriver
4. Selenium IDE
   a. Selendroid (For Android applications)
   b. Appium (For IOS and Android applications)

Required Software's:

1. JDK (Java Development Kit)
2. Eclipse
3. Browsers (Chrome ,Firefox,IE etc)

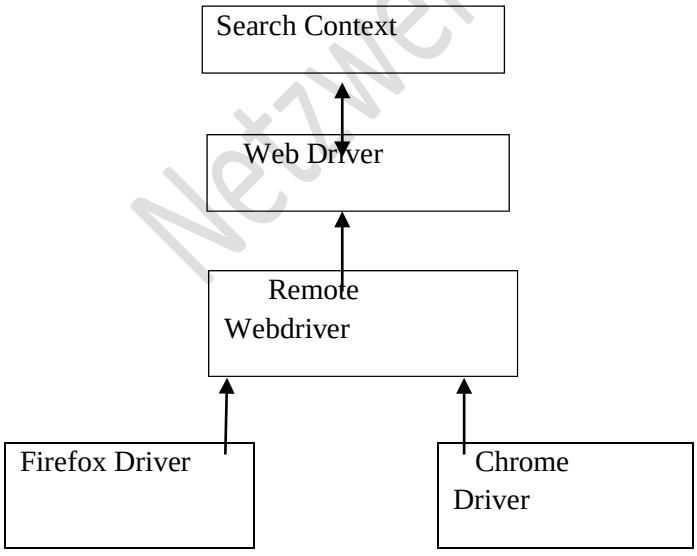
4. Selenium Jar File

Architecture of Selenium Webdriver:



Selenium supports different coding languages which are called as language bindings. The language bindings communicate with the selenium server which performs the action on the browser with the help of browser specific driver executable files (ex: chrome driver for Google chrome browser). Selenium uses Json wire protocol (Java script object notation).

Architecture of Webdriver API



Search context is super most interfaces which are extended by Webdriver interface. Abstract methods of these two interfaces are implemented in Remote Webdriver class and overridden in specific browser classes such as Firefox Driver, Chrome Driver, IE Driver etc.

Configure Selenium with the following steps:

1. Go to required location (C or D) and create a folder with any name Ex:demo
2. In Eclipse go to File>Switch Workspace>Other.
3. Browse and select newly created folder demo and click ok .It will restart the Eclipse.
4. Go to File>New>Project (Java project).Specify the name as Automation and click Finish.
5. Right click on the New Automation project and select New>Folder, give name as Jar files and click finish.
6. Copy the selenium jar file, right click on jar file folder and select Paste.
7. Expand the jar file folder, right click on copied Selenium jar file, go to Build Path and select Add to build path.
8. Right click on src, go to New > Package and give name as demoproject (Everything should be written in small case and click Finish)
9. Right click on demoproject and go to New >Class. Give class name as Demo. Select Public Static void Main, click Finish.

Sample code:

```
 Package demoproject;

import org.openqa.selenium.firefox.FirefoxDriver;

Public class demo {

        Public static void main (String [] args) {

System.setProperty ("webdriver.gecko.driver"," path of the driver executable\\geckodriver.exe");

            FirefoxDriver f=new FirefoxDriver ();
    }

}
```

Notes: If browser is not opening it could be because of the following reasons:

1. Version is not correct: Make sure the version of selenium jar file and driver executables are up to date.
2. Browser path is different: If the driver executables are installed in the different locations then specify the path of browsers in environment variable path.

Example code: script to open google.com and verify that title is Google and also verify that it is redirected to google.com.in

Package demoproject

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

```
public class Demo{

public static void main(String[] args) {

        System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

                WebDriver driver = new FirefoxDriver();

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

                //WebDriver driver = new ChromeDriver();

        String baseUrl = "https://www.netzwerkconsultants.com";

    String expectedTitle = "Welcome: Are you looking for Job";

    String actualTitle = " ";

    driver.get(baseUrl);

    actualTitle = driver.getTitle();

   if (actualTitle.contentEquals(expectedTitle)){

       System.out.println("Test Passed!");

    } else {

       System.out.println("Test Failed");

    }

   driver.close();

    }

}
```


WEB ELEMENT:

Anything present on the webpage is called as web element such as textbox, link, button etc.

The Elements are created using HTML (Hyper Text Markup Language)

Example: Open the notepad and write the following code

```
<html>

<body>

<a href=http://localhost id=”al” name =”nl” class=”cl”>demo </a>

</body>
```

</html>

Go to file and select save, save it in required location .Save the name with .html (Demo.html) and click save.

Double click on the newly created file which opens the file in default browser.

Components of HTML Element:

1. Tag
2. Attribute
3. Text

Selenium code to open the above web page:

WebDriver driver=new FirefoxDriver ();

driver.get ("file:///D:/Demo.html");

Notes: To see the source code of the web element on the webpage right click on the element and select "Inspect Element".

Locators:

Locators are defined as the static methods of By class which is a abstract class. These are used to identify the elements, before performing any action on the web elements we should find the element.

Types of Locators: There are 8 types of locators

1. By.id
2. By.name
3. By.classname
4. By.tagname
5. By.LinkText
6. By.partialLinkText
7. By.Xpath
8. By.CSSselector [Cascading style sheets]

Sample code:

Package demoproject;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

```
import org.openqa.selenium.firefox.FirefoxDriver;
 public class Demo7
{ public static void main(String[] args)
 {
WebDriver driver=new FirefoxDriver();
 driver.get("file:///D:///Demo.html");
By b=By.tagName("a");
WebElement e = driver.findElement(b);
e.click();
}
}
```

Using id

Driver.findElement (By.id (“a1”)).click ();

Using name

Driver.findElement (By.name (“n1”)).click ();

Using class name

Driver.findElement (By.classname (“c1”)).click ();

Using tag name

Driver.findElement (By.tagname (“a”)).click ();

Using Link text

Driver.findElement (By.LinkText ("demo")).click ();

Using Partial Link Text

Driver.findElement (By.partialLinkText ("de")).click ();

Sample code to use CSS selector

<html>

```
<body>
UN<input type="text">
PW<input type="password">
</body>
</html>
```

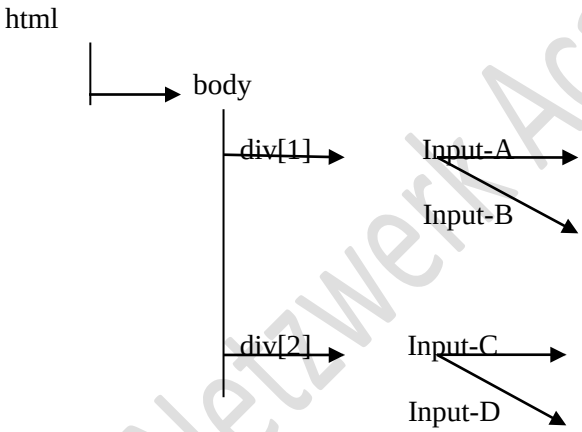
Syntax: Tag [AttributeName=’AttributeValue’]

driver.findElement (By.cssSelector ("input [type='text']")).sendKeys ("user");

XPath : XPath is defined as the XML path ,it is the path of the element in HTML tree .xpath is used to find the location of any element on a webpage using the html structure

Syntax: xpath=//tagname[@attribute=’value’]

Consider the following html tree to derive the Xpath



Types of XPath:

1. **Absolute xpath**: It is the direct way to find the element; it begins with the single forward slash (/) which represents the root node of the element.
   Disadvantage: If there are any changes in the path of the element then that Xpath gets failed.

Ex:

Xpath	Matching element
/html/body/div[1]/input[1]	A
/html/body/div[1]/input[2]	B
/html/body/div[2]/input[1]	C
/html/body/div[2]/input[2]	D
/html/body/div[1]/input	AB
/html/body/div[2]/input	CD
/html/body/div/input[1]	AC
/html/body/div/input[2]	BD
/html/body/div/input	ABCD
/html/body/div[1]/input[1]| /html/body/div[2]/input[2]	AD
/html/body/div[1]/input[2]| /html/body/div[2]/input[1]	BC
/html/body/div[1]/input[1]| /html/body/div[1]/input[2]| /html/body/div[2]/input[1]	ABC

2. **Relative xpath**: For Relative Xpath the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.

Ex:

//div[1]/input[2]	B
//div[2]/input[1]	C
//div[2]/input[2]	D
//div[1]/input	AB
//div[2]/input	CD
//input[1]	AC
//input[2]	BD
//input	ABCD
//div[1]/input[1]| //div[2]/input[2]	AD
//div[1]/input[2]| //div[2]/input[1]	BC
//div[1]/input[1]| //div[1]/input[2]| //div[2]/input[1]	ABC

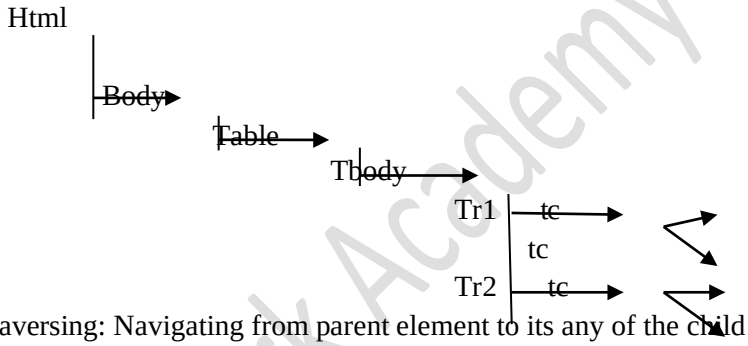
3. **Xpath by Attribute** : XPath expression select nodes or list of nodes on the basis of attributes like **ID , Name, Class name**,
   Syntax: //tag[@Attributename='AttributeValue']
   Ex: //input [@id='username']

4. **Xpath by Text() function** : If Attribute is matching with more than one element or if the attribute is not present then we can identify the element using its text.
   Syntax: //tag[text()='textvalue']
   Ex: //div[text()='login']
   Notes to remember: If there is a 'Non Breakable Space' in attribute value or in text value then 'xpath' will not identify the element.

5. **Contains () function**: Contains () is a method used in XPath expression. It is used when the value of any attribute changes dynamically. And also we can use contains function when there is a 'non breakable space' to identify the element.
   Syntax :
   a. //tag[contains(@AttributeName, 'Attribute Value')]
   b. //tag[contains(text(),'textvalue')]

   Ex:

   a. //button[contains(@type, 'submit')]
   b. //button[contains(@text, 'Sign in')]
6. **Traversing in Xpath** : we can write xpath expression which navigates from one element to another element which is called as traversing .There are 2 types of traversing
   1. Forward Traversing
   2. Backward Traversing



   1. Forward Traversing: Navigating from parent element to its any of the child element is called as forward traversing.
      Ex: Navigating from table node to second cell (//table/tbody/tr[1]/tc[2])

   2. Backward Traversing: Navigating from child element to any of its parent element is called as backward traversing.
      Ex: Navigating from second cell to table node
      Considering second cell has "selenium" word
      (//tc[text()='selenium']/../../..)

7. **Independent-Dependent Xpath**: If the element is completely dynamic or if the element is duplicate, then we can identify that element using some other element by applying a technique called independent, Dependent xpath.
   Ex:

1	Java	100
2	Selenium	200

   Write Xpath to identify the cost of Java book:
   //td[text()='100'] - In the first 'xpath' expression we are identifying the cost directly. If the cost of the Java book changes then this 'xpath' will not identify the element.

//td[text()='Java']/../td[3] - In the 2nd 'xpath' expression we are identifying the cost using the name of the subject. In this example Java is called as independent element and cost is called as dependent element. This will identify the cost even if it completely changes.

Easy steps to derive the Independent –Dependent
1. Inspect the independent element and note the source code.
2. Place the mouse pointer on source code of independent element and move the mouse pointer on upward direction step by step till it highlights both independent and dependent element. It will be the common parent. Add it to HTML tree.
3. Navigates from common parent to dependent element using arrow key and add it to HTML tree.

8. **Xpath by Group Index**: Retrieving one element from a group of matching elements by using index is called group index. When the xpath expression matches with multiple elements, then we go for group index. In Group index, we write xpath expression within the braces and then we write the index outside the braces. Internally, it executes the xpath expression first and stores the result in an xpath array whose index starts with 1.
   a. Last () is a function that is used to retrieve the last element present in the xpath array.
   b.position() is a function that is used to retrieve the position of element in the xpath array.

Consider the following html code

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

The below Table represents the different Xpath examples using group index .It also represents the single xpath expression may match multiples elements, but selenium always return the first matching element.

Xpath using group index	Matching element
//input	ABCD
(//input)[1]	A
(//input)[2]	B
(//input)[3]	C
(//input)[4]	D
(//input)[last()]	D
(//input)[last()-1]	C
//input[1]	AC
(//input[1])[1]	A
(//input[1])[2]	C
(//input[1])[last()]]	C
//input[2]	BD
(//input[2])[1]	B
(//input[2])[2]	D
(//input[2])[last()]]	D
(//input)[position()=1]	A
(//input)[position()=2]	B
(//input)[position()>=3]	CD
(//input)[position()=1 OR position()=last()]	AD

SYNCHRONIZATION: Process of matching Selenium speed with application is called as Synchronization. On real time applications when Selenium try to find the element it may through 'NoSuchElementException' even though specified locator is correct. To handle this we can use 'Sleep' method of thread class.

There are two types

1. Implicit Wait
2. Explicit Wait

1. **Implicit Wait:** The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception". The default setting is 0. Once we set the time, web driver will wait for that time before throwing an exception.
   Syntax: driver.manage().timeouts().implicitlyWait(Timeout,TimeUnit.SECONDS);
   Sample Code:

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.

If we use implicitlyWait then if the element is not located the findElement()method will keep searching for the element after every 500 MILLISECONDS. This is duration is called as "Poling Period". This is

specified in a class called FluentWait. If the element is not located even after the duration then we get NoSuchElementException.

Explicit Wait: The explicit wait is used to tell the Web Driver to wait for certain conditions (**Expected Conditions**) or the maximum time exceeded before throwing an "**ElementNotVisibleException**" exception.

2. **Explicit Wait:** The explicit wait is an intelligent kind of wait, but it can be applied only for specified elements. Explicit wait gives better options than that of an implicit wait as it will wait for dynamically loaded Ajax 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

When the control comes to wait.until statement it will keep checking the condition after every 500 Mili Seconds. If the condition is satisfied it will go to next statement. If the condition is not satisfied even after the duration we get TimeoutException.

All the conditions are present in the class called ExpectedConditions. These conditions are also called as Predicate

Syntax :webDriverWait wait=new WebDriverWait(WebDriverRefrence,Timeout);

Differences between Implicit wait and Explicit wait

Implicit Wait	Explicit Wait
Implicit Wait time is applied to all the elements in the script	Explicit Wait time is applied only to those elements which are intended by user
we need **not** specify "Expected Conditions" on the element to be located	we need to specify "Expected Conditions" on the element to be located
It is recommended to use when the elements are located with the time frame specified in implicit wait	It is recommended to use when the elements are taking long time to load and also for verifying the property of the element like(visibilityOfElementLocated, elementToBeClickable,elementToBeSelected)
Duration can be DAYS, HOURS, MINUTES, SECONDS etc.	Duration will be only Seconds

Action class in Selenium: It s a built-in feature provided by the selenium for handling keyboard and mouse events. It includes various operations such as multiple events clicking by control key, drag and drop events and many more. These operations from the action class are performed using the advanced

Handling Keyboard & Mouse Events

Handling special keyboard and mouse events are done using the **Advanced User Interactions API**. It contains the **Actions** and the **Action** classes that are needed when executing these events. The following are the most commonly used keyboard and mouse events provided by the Actions class.

	Method	Description
2	clickAndHold()	Clicks (without releasing) at the current mouse location.
3	contextClick()	Performs a context-click at the current mouse location. (Right Click Mouse Action)
4	doubleClick()	Performs a double-click at the current mouse location.
5	dragAndDrop(source, target)	Performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse.
6	dragAndDropBy(source, x-offset, y-offset)	Performs click-and-hold at the location of the source element, moves by a given offset, then releases the mouse.
7	keyDown(modifier_key)	Performs a modifier key press. Does not release the modifier key - subsequent interactions may assume it's kept pressed.
8	keyUp(modifier _key)	Performs a key release.
9	moveByOffset(x-offset, y-offset)	Moves the mouse from its current position (or 0,0) by the given offset.
10	moveToElement(toElement)	Moves the mouse to the middle of the element.
11	release()	Releases the depressed left mouse button at the current mouse location
12	sendKeys(onElement, charsequence)	Sends a series of keystrokes onto the element.

Few Sample Codes:

Take any application or web page of your choice and try to code all the following scenarios

1. Code to print the value present in the Textbox

```
package demo;
    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.Test;
   import io.github.bonigarcia.wdm.WebDriverManager;

        public class PrintValueInTxtBox {
          WebDriver driver;

          @Test
       public void printTextBoxValue() {
```

```
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("https://gmail.com/");
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

        WebElement userNameTxt = driver.findElement(By.id("identifierId"));
        userNameTxt.sendKeys("abc@gmail.com");
        String textBoxValue = userNameTxt.getAttribute("value");
        System.out.println(textBoxValue);
        driver.quit();
    }
}
```

2. Code to change the value present in the Text Box

```
package demo;
   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.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class ChangeTextBoxVale {
   WebDriver driver;

@Test
public void changeTextBoxValue() {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("https://gmail.com/");
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

        WebElement userNameTxt = driver.findElement(By.id("identifierId"));
        userNameTxt.sendKeys("abc@gmail.com");
        String presentTextBoxValue = userNameTxt.getAttribute("value");
        System.out.println(presentTextBoxValue);
        userNameTxt.clear();
        userNameTxt.sendKeys("xyz@gmail.com");
        String modifiedTextBoxValue = userNameTxt.getAttribute("value");
        System.out.println(modifiedTextBoxValue);
```

```
                driver.quit();
        }
}
```

3. Code to remove text present in the text box without using clear method

```
package demo;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;
    public class RemoveTextWithoutClearFunction {
            WebDriver driver;
    @Test
public void removeTextWithoutClearFunction() throws InterruptedException {
            WebDriverManager.chromedriver().setup();
            driver = new ChromeDriver();
            driver.get("https://gmail.com/");
            driver.manage().window().maximize();
            driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

            WebElement userNameTxt = driver.findElement(By.id("identifierId"));
            userNameTxt.sendKeys("abc@gmail.com");
            userNameTxt.sendKeys(Keys.CONTROL + "a");
            userNameTxt.sendKeys(Keys.DELETE);
            Thread.sleep(2000);
            driver.quit();
        }
}
```

4. Code to clear the text present in the text box by pressing back space key package demo;

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
 import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;
```

```
public class RemoveTextWithBackspaceFunction {
    WebDriver driver;

@Test
public void removeTextWithBackspaceFunction() throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("https://gmail.com/");
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

        WebElement userNameTxt = driver.findElement(By.id("identifierId"));
        userNameTxt.sendKeys("abc@gmail.com");
        userNameTxt.sendKeys(Keys.CONTROL + "a");
        userNameTxt.sendKeys(Keys.BACK_SPACE);
        Thread.sleep(2000);
        driver.quit();
}
}
```

5. Code to copy and paste the value present in one text box into another text box

```
package demo;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
 import org.testng.annotations.Test;

    public class CopyPasteText {
    WebDriver driver;

@Test
public void copyPasteText() throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("https://www.facebook.com/login/");
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

        WebElement userNameTxt = driver.findElement(By.id("email"));
        WebElement passwordTxt = driver.findElement(By.id("pass"));
```

```
                userNameTxt.sendKeys("abc@gmail.com");
                userNameTxt.sendKeys(Keys.CONTROL + "a" + "c");
                passwordTxt.sendKeys(Keys.CONTROL + "v");
                Thread.sleep(2000);
                driver.quit();
        }
}
```

6. Code to print the text of the link.

```
package demo;
 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.Test;
 import io.github.bonigarcia.wdm.WebDriverManager;

 public class PrintTextOfALink {
  WebDriver driver;

 @Test
 public void printTextOfLink() throws InterruptedException {
                WebDriverManager.chromedriver().setup();
                driver = new ChromeDriver();
                driver.get("https://www.facebook.com/login/");
                driver.manage().window().maximize();
                driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

                WebElement forgotPwd = driver.findElement(By.xpath("//a[text()='Forgotten
account?']"));
                String linkText = forgotPwd.getAttribute("href");
                System.out.println("Reset password link: "+linkText);
                Thread.sleep(2000);
                driver.quit();
        }
}
```

7. Code to print x and y coordinates of an element

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

```
import org.openqa.selenium.By;
import org.openqa.selenium.Point;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class XYCoordinatesOfElement {
WebDriver driver;

	@Test
	public void printTextOfLink() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement userNameTxt = driver.findElement(By.id("email"));
		Point point = userNameTxt.getLocation();
		int xcord = point.getX();
		System.out.println("Position of the webelement from left side is "+xcord +" pixels");
		int ycord = point.getY();
		System.out.println("Position of the webelement from top side is "+ycord +" pixels");
		Thread.sleep(2000);
		driver.quit();
	}
}
```

8. **Code to check that email text box and Next button present in Gmail login page are aligned horizontally? (x value should be same)**

```
	import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Point;
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;
import io.github.bonigarcia.wdm.WebDriverManager;

public class AlignementTest {
WebDriver driver;
```

```
	@Test
	public void alignementTest() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement userNameTxt = driver.findElement(By.id("email"));
		WebElement passwordTxt = driver.findElement(By.id("pass"));
		Point unpoint = userNameTxt.getLocation();
		Point pwdpoint = passwordTxt.getLocation();
		int userNameXcord = unpoint.getX();
		int passwordXcord = pwdpoint.getX();
		Assert.assertEquals(userNameXcord, passwordXcord);
		System.out.println(userNameXcord + " and " + passwordXcord + " are aligned
horizontally.");
		Thread.sleep(2000);
		driver.quit();
	}
}
```

9. Code to print the width and height of a text box

```
package demo;
	import java.util.concurrent.TimeUnit;
	import org.openqa.selenium.By;
	import org.openqa.selenium.Point;
	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;
	import io.github.bonigarcia.wdm.WebDriverManager;

	public class WidthAndHeightOfElement {
	 WebDriver driver;

	  @Test
```

```
		public void widthAndHeightOfElement() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement userNameTxt = driver.findElement(By.id("email"));
		int userNameWidth = userNameTxt.getSize().width;
		int userNameHeight = userNameTxt.getSize().height;
		System.out.println("Width of the element is: " + userNameWidth);
		System.out.println("Height of the element is: " + userNameHeight);
		Thread.sleep(2000);
		driver.quit();
	}
}
```

10. **Script to verify that heights of email password and login button which are present in FB login page are same?**

```
 package demo;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Point;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class VerifyTxtFieldHight {
	WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
```

```
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement userNameTxt = driver.findElement(By.xpath("//input[@id='email']"));
		WebElement passwordTxt = driver.findElement(By.xpath("//input[@id='pass']"));
		WebElement loginBtn = driver.findElement(By.xpath("//button[@id='loginbutton']"));
		int unpoint = userNameTxt.getSize().getHeight();
		int pwdpoint = passwordTxt.getSize().getHeight();
		int loginpoint = loginBtn.getSize().getHeight();
		System.out.println(unpoint);
		System.out.println(pwdpoint);
		System.out.println(loginpoint);
		if (unpoint == pwdpoint && unpoint == loginpoint) {
			System.out.println("Height of email password and login button which are present
			in FB login page are same.");
		}else {
			System.out.println("Height of email password and login button which are present
			in FB login page are not same.");
		}
		Thread.sleep(2000);
		driver.quit();
	}
}
```

11. Script to verify that email text box present in Facebook login page is empty? Hint: get the value and check the length of it. Length should be 0.

```
	package demo;
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.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class TestBoxIsEmpty {
WebDriver driver;

	@Test
	public void verifyTextBoxIsEmpty() throws InterruptedException {
```

```
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement userNameTxt = driver.findElement(By.xpath("//input[@id='email']"));
		String value = userNameTxt.getAttribute("value");
		int count =value.length();
		if(count == 0 && value.isEmpty()==true) {
			System.out.println("Facebook login page email text box is empty.");
		}else {
			System.out.println("Facebook login page email text box is empty.");
		}
		Thread.sleep(2000);
		driver.quit();
	}
}
```

12. Script to verify whether login button is enabled or not which is present in the FB page?

```
	package demo;
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.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class VerifyLoginBtnIsEnebled {
WebDriver driver;

	@Test
	public void verifyBtnIsEnabled() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement loginBtn = driver.findElement(By.xpath("//button[@id='loginbutton']"));
		if (loginBtn.isEnabled()) {
			System.out.println("Login button is enabled in FB login page.");
```

```
		}else {
			System.out.println("Login button is disabled in FB login page.");
		}
		Thread.sleep(2000);
		driver.quit();
	}
}
```

13. Code to delete all the cookies present in the browser

```
	package demo;
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.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class DeleteCookies {
WebDriver driver;

	@Test
	public void deleteCookies() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		driver.manage().deleteAllCookies();
		Thread.sleep(2000);
		driver.quit();
	}
}
```

14. Code to print background color of a textbox

```
	package demo;
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.Test;
```

```
import io.github.bonigarcia.wdm.WebDriverManager;

public class VerifyBackgroundColor {
WebDriver driver;

	@Test
	public void verifyBackgroundColor() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement loginBtn = driver.findElement(By.xpath("//button[@id='loginbutton']"));
		String backgroundColor = loginBtn.getCssValue("background-color");
		String color = loginBtn.getCssValue("color");
		System.out.println(backgroundColor);
		System.out.println(color);
		Thread.sleep(2000);
		driver.quit();
	}
}
```

15. Code to click on the button using java scripts.

```
	package demo;
import java.util.concurrent.TimeUnit;
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.testng.annotations.Test;

import io.github.bonigarcia.wdm.WebDriverManager;

public class ClickUsingJavaScript {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
```

```
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		driver.findElement(By.xpath("//input[@id='email']")).sendKeys("abc@gmail.com");
		driver.findElement(By.xpath("//input[@id='pass']")).sendKeys("123456");
		WebElement loginBtn = driver.findElement(By.xpath("//button[@id='loginbutton']"));
		JavascriptExecutor executor = (JavascriptExecutor)driver;
		executor.executeScript("arguments[0].click();", loginBtn);
		Thread.sleep(5000);
		driver.quit();
	}
}
```

16. Code to enter the text into the textbox without using 'sendkeys'

```
	package demo;
import java.util.concurrent.TimeUnit;
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.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class EnterTextUsingJavaScript {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		JavascriptExecutor jse = (JavascriptExecutor) driver;
		jse.executeScript("document.getElementById('email').value='xyz@gmail.com'");
		Thread.sleep(3000);
		driver.quit();
	}
}
```

17. Code o scroll to the bottom of the web page

```
	package demo;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class ScrollToEnd {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://stackoverflow.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		JavascriptExecutor jse = (JavascriptExecutor) driver;
		jse.executeScript("window.scrollTo(0, document.body.scrollHeight)");
		Thread.sleep(3000);
		driver.quit();
	}
}
```

18. Code to scroll to the specific element? Hint: get the Y coordinate of the element using get location method and pass it as argument for 'scrollTo' method.

```
	package demo;
import java.util.concurrent.TimeUnit;
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.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class ScrollToSpecificElement {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
```

```
		driver.get("https://stackoverflow.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement element = driver.findElement(By.xpath("//a[contains(text(),'Stack Overflow
Advertising')]"));
		JavascriptExecutor jse = (JavascriptExecutor) driver;
		jse.executeScript("arguments[0].scrollIntoView(true);", element);
		Thread.sleep(3000);
		driver.quit();
	}
}
```

19. Code to print text of links present on the page.

```
	package demo;
import java.util.List;
import java.util.concurrent.TimeUnit;
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.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class PrintTextOfAllLinks {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://stackoverflow.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		List<WebElement> elements = driver.findElements(By.xpath("//a"));
		for(WebElement element:elements) {
			System.out.println(element.getAttribute("href"));
		}
		Thread.sleep(3000);
		driver.quit();
	}
```

```
}
```

20. Code to select all the check boxes present on the page from top to bottom

```
package demo;
import java.util.List;
import java.util.concurrent.TimeUnit;
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.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class HandleCheckBox {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.ironspider.ca/forms/checkradio.htm");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement element = driver.findElement(By.xpath("//form[contains(text(),'My
favourite colors are:')]"));
		JavascriptExecutor jse = (JavascriptExecutor) driver;
		jse.executeScript("arguments[0].scrollIntoView(true);", element);
		Thread.sleep(2000);

		List<WebElement> checkboxes =
driver.findElements(By.xpath("//input[@type='checkbox']"));
		for(WebElement checkbox:checkboxes) {
			checkbox.click();
			Thread.sleep(1500);
		}
		Thread.sleep(2000);
		driver.quit();
	}
```

```
}
```

21. Code to print sum of all the numbers present in the table

```
package demo;
import java.util.List;
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.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class SumOfTableContent {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.techlistic.com/p/demo-selenium-practice.html");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
		int sum =0;
		List<WebElement> elements = driver.findElements(By.xpath("//td[4]"));
		for(WebElement element:elements) {
			int tableValue = Integer.parseInt(element.getText());
			sum = sum + tableValue;
		}
		System.out.println("Sum of table content is: " + sum);
		Thread.sleep(3000);
		driver.quit();
	}
}
```

22. Code to print all the contents of the listbox.

```
package demo;
import java.util.List;
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.openqa.selenium.support.ui.Select;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class PrintListBoxContents {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		driver.findElement(By.xpath("//a[@id='u_0_2']")).click();
		WebElement monthListBox = driver.findElement(By.xpath("//select[@id='month']"));
		Select select = new Select(monthListBox);
		List<WebElement> allElements = select.getOptions();
		for(WebElement element:allElements) {
			System.out.println(element.getText());
			Thread.sleep(1500);
		}
		Thread.sleep(3000);
		driver.quit();
	}
}
```

23. Code to print the content of list in sorted order

```
	package demo;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.TreeSet;
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.openqa.selenium.support.ui.Select;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;
```

```
public class ContentListInSortedOrder {
WebDriver driver;

	@Test
	public void verifyHeight() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		driver.findElement(By.xpath("//a[@id='u_0_2']")).click();
		WebElement monthListBox = driver.findElement(By.xpath("//select[@id='month']"));
		Select select = new Select(monthListBox);
		List<WebElement> allElements = select.getOptions();
		List<String> contentList = new ArrayList<String>();
		for(WebElement element:allElements) {
			contentList.add(element.getText());
		}
		TreeSet<String> set = new TreeSet<String>(contentList);
		Iterator itr = set.iterator();
		while(itr.hasNext()) {
			System.out.println(itr.next());
		}
		Thread.sleep(3000);
		driver.quit();
	}
}
```

24. Code to take screenshot of the application.

```
package demo;
import java.io.File;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class TS_002 {
WebDriver driver;
```

```
	@Test
	public void takeScreenShot() throws Exception {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://gmail.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		// Take screenshot and store as a file format
		File src= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
		// now copy the screenshot to desired location using copyFile //method
		FileUtils.copyFile(src, new File("C:/selenium/error.png"));
		Thread.sleep(2500);
		driver.quit();
	}
}
```

25. Write a script to login and logout from the application without specifying the waiting period or without using any of the Synchronization methods

```
	package demo;
import org.openqa.selenium.By;
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;
import io.github.bonigarcia.wdm.WebDriverManager;

public class LoginWithoutWait {
WebDriver driver;

	@Test
	public void loginWithoutSync() throws Exception {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();

		WebElement userNameTxt = driver.findElement(By.id("email"));
		WebElement passwordTxt = driver.findElement(By.id("pass"));
		userNameTxt.sendKeys("abc@gmail.com");
		passwordTxt.sendKeys("**********");

		driver.findElement(By.xpath("//i[@class='hu5pjgll op6gxeva sp_LYAc5k-1MU4_1_5x
sx_6da5ac']")).click();
		driver.findElement(By.xpath("//span[contains(text(),'Log Out')]")).click();

		Assert.assertEquals(driver.getTitle(), "Facebook – log in or sign up");
```

```
		Thread.sleep(2500);
		driver.quit();
	}

}
```

26. a script to verify that width of email textbox and next button is same which are present in Gmail login page?

```
	package demo;
import java.io.File;
import java.util.concurrent.TimeUnit;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class TS_014 {
WebDriver driver;

	@Test
	public void verifyWidth() throws Exception {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://gmail.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

		WebElement textbox = driver.findElement(By.id("identifierId"));
		WebElement nextBtn = driver.findElement(By.xpath("//div[@class='VfPpkd-
RLmnJb']"));

		if(textbox.getSize().getWidth() == nextBtn.getSize().getWidth()) {
			System.out.println("Width of email textbox and next button is same which are
present in Gmail login page");
		}else {
			System.out.println("Width of email textbox and next button is not same which
are present in Gmail login page");
		}
		Thread.sleep(2500);
		driver.quit();
```

```
}
}
```

27. code to remove the value present in the textbox using Java Script

```
  package demo;
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.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class TS_025 {
WebDriver driver;

	@Test
	public void removeTextUsingJS() throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		driver = new ChromeDriver();
		driver.get("https://www.facebook.com/login/");
		driver.manage().window().maximize();

		WebElement userNameTxt = driver.findElement(By.id("email"));
		userNameTxt.sendKeys("abc@gmail.com");

		JavascriptExecutor js = (JavascriptExecutor)driver;
		js.executeScript("arguments[0].value = ';", userNameTxt);
		Thread.sleep(2000);
		driver.quit();
	}
}
```

28. How do you enter the value if textbox is disabled?
Using Java Script

```
package demo;
import java.util.concurrent.TimeUnit;
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.testng.annotations.Test;
import io.github.bonigarcia.wdm.WebDriverManager;

public class TS_026 {
```

```
WebDriver driver;

        @Test
        public void typeToDisabledField() throws InterruptedException {
                WebDriverManager.chromedriver().setup();
                driver = new ChromeDriver();

        driver.get("https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_input_disabled");
                driver.manage().window().maximize();
                driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

                WebElement disbaledField = driver.findElement(By.id("//input[@id='lname']"));
                ((JavascriptExecutor)driver).executeScript("arguments[0].value='10/10/1986'",
disbaledField);
                Thread.sleep(2000);
                driver.quit();
        }


}
```

Different methods of Web element Interface:

- ✓ clear()
- ✓ click()
- ✓ getAttribute()
- ✓ getCssValue()
- ✓ getLocation()
- ✓ getSize()
- ✓ getText()
- ✓ isDisplayed()
- ✓ isEnabled()
- ✓ isSelected()
- ✓ sendkeys()
- ✓ submit()

Methods to handle multiple elements in Selenium:

In order to handle multiple elements we use findElements() method which returns List of Web Element

[List<WebElement>]. List should be imported from java.util package.

Under the List we frequently use the following two methods.

1. list.size() - It returns element present in the List (return type is int)
2. list.get() – It returns element present in the specified index (return type WebElement)

- Find Element command returns the web element that matches the first most elements within the web page.
- Find Elements command returns a list of web elements that match the criteria.
- Find Element command throws NoSuchElementException if it does not find the element matching the criteria.
- Find Elements command returns an empty list if there are no elements matching the criteria

.Differences between findElement() and findElements() :

findElement()	findElements()
Return type is WebElement	Return Type is List<WebElement>
If the specified locator is matching with Multiple elements, it returns first matching element.	If the specified locator is matching with multiple elements it returns all the matching elements
If the specified locator is not matching with any of the element then it will throw NoSuchElementException.	If the specifies locator is not matching with any of the element it will not throw any exception instead of this it returns empty list.

Handling Listbox :

To handle the listbox, we use Select class of selenium. It should be imported from the following packages: import **org.openqa.selenium.suport.ui.Select**

Select class has parameterized constructor (single argument constructor) it takes an argument type

WebElement (address of the listbox). In order to select the required option present in the listbox we can use the following method of Select class.

1. selectByVisibleText(str) --- string argument
2. selectByIndex(int) ---integer argument
3. selectByValue(str) ---takes string argument

Note : If the specified option is duplicate it will select the first matching option in the drop down list .
If the specified option is not present then we get NosuchElementException.

Select class can also be used to handle mulitselect listbox. If the specified option is duplicate in mutliselect listbox, it selects all the matching option.
To handle multiselect Listbox we can use the following methods of the Select Class .

1. deselectByVisibleText(string)
2. deselectByIndex(Integer)
3. deselectByValue(string)

4. deselectAll()

Important Note : If the listbox developed using some other tag such as input, div, li, ul(unorderedlist),ol etc then if we try to use Select class we get UnexceptedTagNameException. To handle this situation we can use click() or sendKeys() method.

Encapsulation: **Encapsulation** in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class, therefore it is also known as data hiding.

The different steps used in Encapsulation

1. Declaration
2. Initialization
3. Utilization

```
public class A
 {
 private int i;
 public void A(int j)
 {
 i=j;
 }
 public int getValue()
 {
 return i;
 }
 }


public class B
 {
 public static void main(String[] args)
 {
 A a=new A(10);

int x=a.getValue() System.out.println(x);
 }

 }
```

Class A: Manage the variable i

Class B : Execute the code

POM Concept in selenium:

To avoid StaleElementReferenceException we use Page Object Model POM class. POM is one of the java design pattern. POM concept is used by both developers and test engineers (automation) to develop and test web pages.

In POM class we declare the element using FindBy Annotation and we write it as @FindBy. It should be imported from the following package

Import org.openqa.selenium.support.FindBy;

Syntax 1: single element

@FindBy(locator="locator value")

private WebElement elementname;

Syntax2: multiple element

@FindBy(locator="locator value")

private List<WebElement> elementname;

To initialize the element we use initElements() method of PageFactory class. It takes two arguments

- WebDriver

- Object of POM class

initElement() method will only loads the element (reference variable),but it will not initialize actually.

Element are actually initialized during runtime when we try to perform any action on the element.

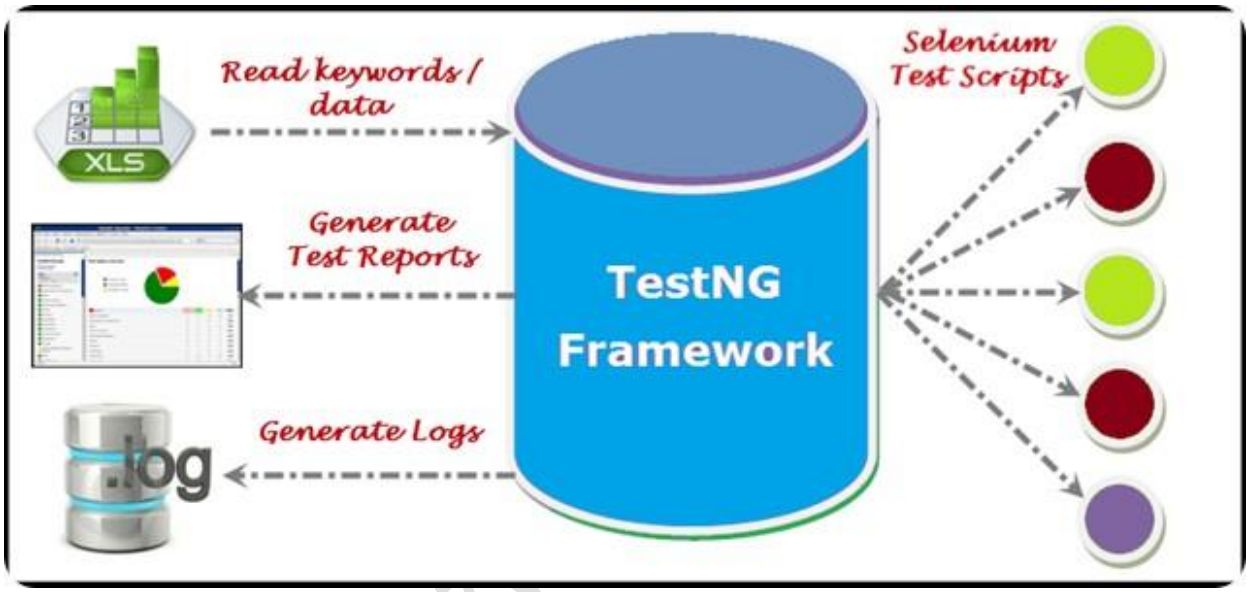
This process is called as Lazy Initialization. This will avoid StaleElementReferenceException. Since only one element is loaded at a time it will improve the performance of the script execution.

TestNG :

It is unit testing framework. Basically TestNG is used by developers to perform unit testing and it is also used in selenium to perform Black Box testing.

- TestNG provides you full control over the test cases and the execution of the test cases. Due to this reason, TestNG is also known as a testing framework.

- If you want to run a test case A before that as a pre-request you need to run multiple test cases before you begin a test case A. You can set and map with the help of TestNG so that pre-request test cases run first and then only it will trigger a test case A. In such way, you can control the test cases.
- TestNG framework eliminates the limitations of the older framework by providing more powerful and flexible test cases with help of easy annotations, grouping, sequencing and parametrizing.
- Run multiple test classes
- Generate Reports
- Rerun only failed classes etc
- TestNG supports three additional levels such as @Before/After suite, @Before/AfterTest, and Before/AfterGroup.
- TestNG does not extend any class. TestNG framework allows you to define the test cases where each test case is independent of other test cases.



TestNG is available as plug-in for Eclipse IDE

STEPS TO INSTALL TestNG

1. Open the eclipseIDE > click on Help > select Eclipse Market Place
2. Search for TestNG
3. Click Install Button of TestNG
4. Select “Accept” and click finish.
5. Click “ok” on the popup
6. Click “yes” which restarts the Eclipse

7. Right click JavaProject and go to Properties

8. Click on Java Build Path

9. Click on “Libraries” tab

10. Click on “Add Library”

11. Select TestNG > click Next > click finish and click on OK, this will associate TestNG with current Java Project.

FOLLOWING 4 ARE THE REQUIRED JAR FILES:

1. testing.jar

2. jcommander.jar

3. bash-2.ob4.jar

4. snakeyaml.jar

TestNG class - While creating TestNG class we should not use

- Default package
- No main() method
- No System.out.println

TestNG Suit

It is an xml file, which contains list of all TestNG classes which are to be executed. Suite file is used for Batch execution. To create it:

1. Right click on Java Project

2. Goto TestNG

3. Select Convert to TestNG

4. Click finish

5. It creates TestNG.xml file inside the JavaProject

To execute it

1. Right click on xml file.

2. Go to Run as & select TestNG suite.

Listeners in TestNG:

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 allow 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.

In this tutorial, we will implement the ITestListener.

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.

Content of TestNG Suite

<suite name="Suite" parallel="none"> <test name="Test"> <classes> <class name="testNg.TestNGdemo1"/> </classes> </test> </suite>

Assert :

assertEquals()

- assertNotEquals()

- assertTrue()

- assertFalse()

- assertSame()

- assertNotSame()

- assertNull()

- assertNotNull()

- fail()

All the above methods are static methods of Assert Class. In order to continue the execution even after failure of the comparison, we can use SoftAssert . But all methods are non static.

Difference between Assert and Soft Assert

Assert	Soft Assert
If comparison fails remaining statement will not be executed in current class	Executes remaining statements even if comparison fails
All methods are static	All methods are non-static
We do not call assertAll() method	We should call assertAll() method

POM: Page Object Model

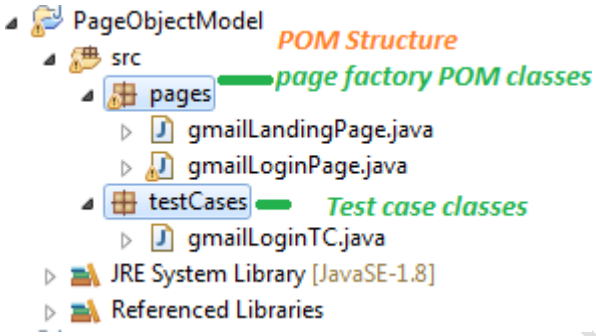
POM is a design pattern which is commonly used in Selenium for Automating the Test Cases. This design pattern can be used with any kind of framework like keyword-driven, Data-driven, hybrid framework, etc.

The Page object is an object-oriented class which acts as an interface for the page of your Application under test. Page class contains web elements and methods to interact with web elements. While automating the test cases, we create the object of these Page Classes and interact with web elements by calling the methods of these classes

Advantages of POM:

1. Page Object 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 POM 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()'.

EX:



Page Factory in Selenium:

The ***PageFactory*** Class in Selenium is an extension to the Page Object design pattern. It is used to initialize the elements of the Page Object or instantiate the Page Objects itself. Annotations for elements can also be created (and recommended) as the describing properties may not always be descriptive enough to tell one object from the other.
It is used to initialize elements of a Page class without having to use 'FindElement' or 'FindElements'. Annotations can be used to supply descriptive names of target objects to improve code readability. There is however a few differences between C# and Java implementation – Java provide greater flexibility with PageFactory.

We will follow a step by step instruction to set up Page Factory.

- Take an example of simple test case of LogIn application and understand where we can use Page factory.
- Implement Page Factory in the LogIn test case
- Use Annotations of Page Factory
- Divide the single Page Factory Objects in to different Page Classes
- Combine set of repetitive actions in to methods to implement Modularization.

Framework:

It is the standard rules, best practices and folder structure which should be followed while automating the application testing.

We should follow the Automation Framework to have consistency. In automation framework we have 3 stages.

I. Automation Framework Design

II. Automation Framework Implementation

III. Automation Framework Execution

I. AUTOMATION FRAMEWORK DESIGN

This is the initial stage where automation lead or manager will specify files and folder structures,

Naming conventions and rules which should be followed to develop and execute the automation script.

Based on the design framework is categorized into following types:

1. Method-driven Automation Framework
2. Data-driven Automation Framework
3. Module-driven Automation framework
4. Keyword-driven Automation framework
5. Hybrid Automation Framework

Note: The above type of Framework can be customized and implemented in a company with different names. Such as Cucumber, Robot, Protractor, Craft etc. Generally, folder structures are decided based on the file types

Steps to configure Automation Framework:

1. Go to required location (ex: D :) drive and create a folder (ex:ABCD6).
2. Go to File > Switch Work Space other, browse and select above folder and click OK.
3. Create a java project with the name “Automation”.
4. Create a folder with the name “exefiles”under “Automation” and copy paste

“chromedriver.exe” and “IEDriverServer.exe”.

5. Create a folder with the name “jarfiles” under Automation and copy paste “selenium server standalone” jar file.

6. Expand “jarfile” folder and right click on “selenium server standalone” jar file and select Build Path > Add to Build Path.

7. Associate TestNG to the javaproject.

8. Create a folder with the name “testdata” under “automation” which is used to store excel files.

9. Create two packages under “src” with the name “pom” and “scripts”.

Pom packages are used to store POM class. Script package is used to store TestNG class

While writing scripts we develop two types of classes:

1. POM class

2. TestNG class

First we should always develop pom class. It is used to store elements and perform the action. We use TestNg class for execution purpose.

Handling Frames:

1. Webpage present inside another webpage is called as embedded webpage. Developer creates embedded webpage using iframe html tag.
2. If right click on any element which inside the frame it will display this frame option in the context menu.
3. To transfer the control into the frame we should use following statement
   driver.switchTo().frame(arg);
4. where arg > can be index of the frame (int) or id of the frame (string) or element of the frame (WebElement).

In order to transfer the control back to the main page we can use the following statements:
driver.switchTo().defaultContent();

5. In case of nested frames, to switch from child frame to parent frame we can use following statements;
   driver.switchTo().parentFrame();

Steps to develop the POM class:

Execute the test case manual which gives more clarify on the steps which is to be automated.

2. While executing the test case note down the page, its elements and the action

3. For each page present on the application create a POM class under “pom” package.

4. The name of the class should be same as Title of the page ending with the word Page.

5. In every POM class we should declare the element using FindBy annotation @FindBy, initialize it using initElemetns() and utilize it using getters and setters methods.

Method Driven Framework:

Executing the script by calling the methods present in the frameworks is called as 'Method Driven Framework'. Methods will avoid repetition of the steps and they will increase re-usability of the code.

In order to test the features thoroughly that is with all possible inputs only methods are not sufficient we

Should use external files such as excel file. If this feature is available in the framework then such type of framework is called 'Data-Driven Framework'.

To handle the excel file we use API provided by Apache called POI (Poor Obfuscation Implementation).

We can download it from following URL: http://mirror.fibergrid.in/apache/poi/release/bin/poi-bin- 3.13-20150929.zip. After downloading the file extract it which creates a folder with the name 'poi-3.13'

It has many jar files, only following 4 jar files are required.

1. poi-3.13-20150929

2. poi-ooxml-3.13-20150929

3. poi-ooxml-schemas-3.13-20150929

4. xmlbeans-2.6.0

Copy above jar files into 'jarfiles' folder of the Framework and then associate them to Java project.

Selenium Grid:

To run automation scripts on multiple browsers on multiple environments (remote computers) we can

Use Selenium Grid.

In Selenium Grid, we will have two types of systems:

1. Hub – it is the computer where framework is present and we start the execution from this Computer, after the execution result is also stored in this system. Selenium grid has only one Hub.

2. Node – Node refers to remote computer on which actual execution happens. In Selenium grid there could be one or more node communicating with the hub . To implement this concept, we should perform following steps

a. Start the hub

b. Start the node

c. Execute Framework using RemoteWebDriver.

a. Start the Hub

1. Go to the system where framework is present.

2. Copy-Paste selenium jar file to the required location.

(ex: D drive) rename it if it is required (s.jar).

3. Open the command prompt and execute following command java –jar d:\s.jar –role hub

4. It should display following message

"Selenium Grid hub is up and running"

b. Start the Node

1. Go to the remote system where the Brower has to be opened and script should be executed.

Node system should contain following software's:

a. jdk

b. Selenium jarfile

c. firefox browser

d. chrome browser with chromedriver exe file

2. Open the command prompt and execute following command

java –jar e:\s.jar –role node –hub http://192.168.1.23:xxxx–

Dwebdriver.chrome.driver=e:/cd.exe

(We can store it in bat file)

c. Execute Framework using remote control

1. Go to precondition method of Base Test class and update the code as shown below Old code

Driver=new ChromeDriver();

New code

driver=new RemoteWebDriver(DesiredCapabilities.chrome());

Old code

driver=new FirefoxDriver();

New code:

driver= new RemoteWebDriver(DesiredCapabilities.firefox()); Execute the framework using suitefile or .bat file

JENKINS:

It is a Continuous Integration Tool used by developers for managing build creation process, which includes following steps:

1. Download the latest source code.
2. Compile the code.
3. Compress the code.
4. Install the build.
5. Send a Email notification.

In order to execute the framework automatically after the installation of the build we should integrate framework with JenKins.

To implement the above concept we should perform following steps:

1. Install and configure JenKins.
2. Integrate Framework with JenKins.
3. Create the Build.

Install and Configure JenKins

1. Download jenkins.war file from http://jenkins-ci.org/ and copy it to required location (ex: D:)
2. Execute following command in the command prompt: java –jar d:\jenkins.war
3. It should display following messge: Jenkins is fully up and running
4. Open the browser and enter the url of the Jenkins ex: http://localhost:xxxx /
5. Click on Create New Jobs
6. Specify the Name

7. Select the first Radio button Free Style Project

8. Click OK and Click Save.

Note: all the above steps done by Developers

Integrate Framework with JenKins. (Imp: by Automation Engineers)

1. Open the homepage of Jenkins and click on the Job Name.

2. Click on Configure

3. Click on Advanced

4. Select Use Custom Workspace

5. Specify the path of Java project folder where batch file is present 123 Selenium

Directory: D:\ABCD6 \Automation

6. Click on Add Build Step and select Execute windows batch command.

7. Specify the batch file name “RunMe.bat” and click Save.

Create the Build. By dev

1. Go to homepage of Jenkins

2. Click on Job Name

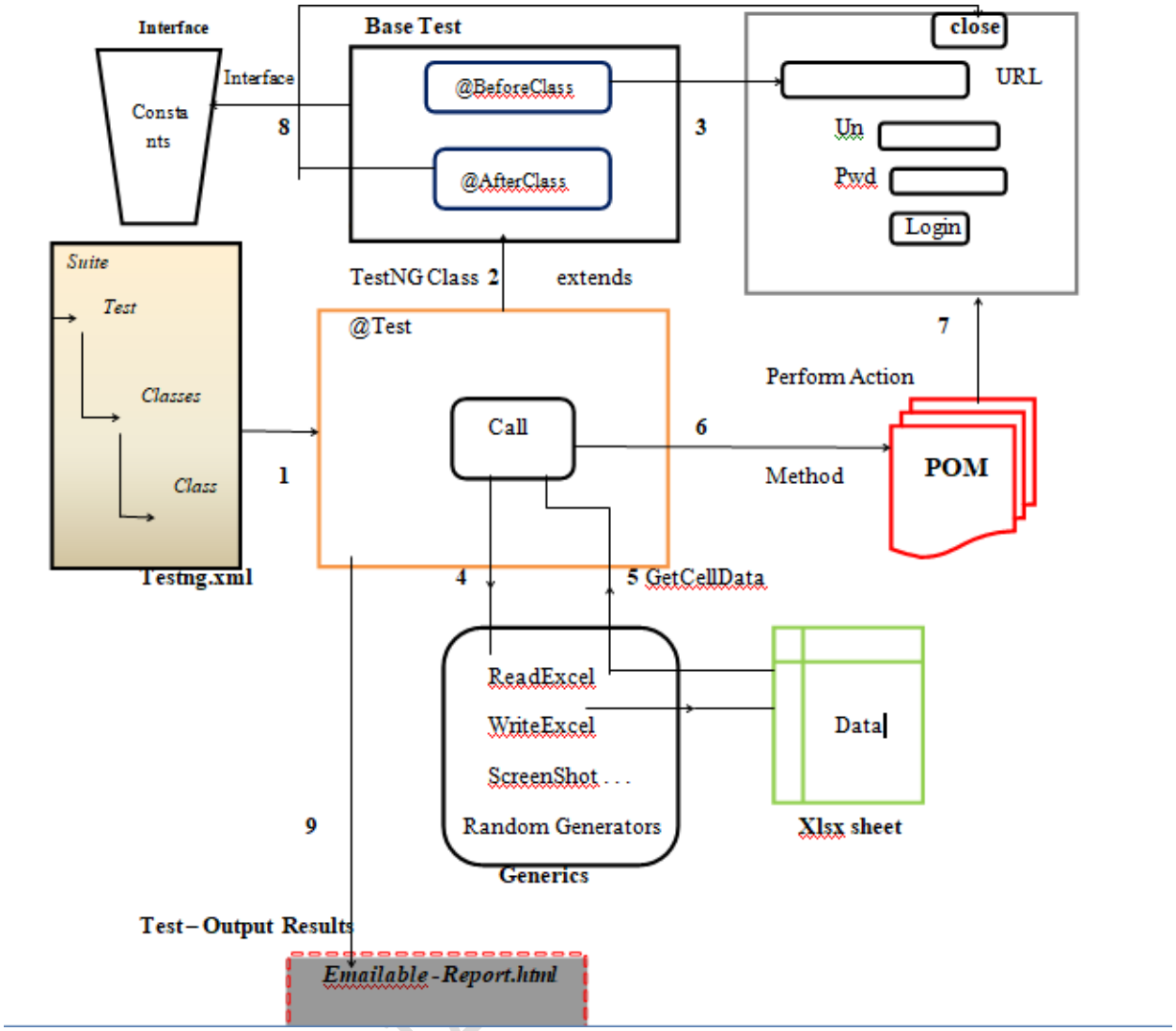
3. Click on Build Now

Note 1: When developer clicks on Build Now Jenkins will create a build, installs the build and then it will start the framework execution.

Note 2: Every time the build creation process started it will display link under Build History, which can be used to see the details.

ex: click on the link present under build history and then click on Console output. We can view execution result of the Automation in Jenkins page itself.

Architecture of the framework



Automation Framework Design Details:

1. TestNG.xml will have all Test Classes that need to be executed, so when we run TestNG.xml file as Test-NG all Test Classes get executed in specified manner.

2. Each Test Class will have each Test Case as a Method which is Preceded by **@Test** annotation with priority and they get executed by their priorities, since each Test Class extends Base Class which has **@Before Test** (which Launch Browser before Executing @Test) and **@After Test**(which Close Browser after @test get Executed)

3. **@Before Test** of Base Class will have a code to Launch Browser and Redirect to **Login Page** of the Application
4. Control comes back to Test Class which will start executing **@test** methods, required data can be accessed by calling Read and Write Data from Excel through Apache POI and those Read and Write Codes are specific throughout the project and they will be in Generics Class and even we can have a Screenshot Taker method in Generics
5. Generics Class Read and Write Methods access data from Excel and Return it Back to **@Test** Method
6. **@Test** creates an Object of required **POM Class** and The Data from Excel acts as an input to Web Elements on Webpage of Application which are identified by POM Classes and which has Methods to Act on Web Elements
7. **POM Class** based on specified x-path identifies Web Element and will Input Data on specified fields
8. once all **@Test** Methods get Executed **@After Class** Method of Base class will close the Browser
9. after **Test Classes** in **TestNG.xml** get executed completely **Test-NG** develops a Beautiful **Emailable – Report.html** which contains complete execution details and can be used to attach along with Mail's for Communication

Architecture Details

Constants Interface: This Interface is designed to make

- Constant User Directory Path
- Constant Screenshot Folder Path

Base Class: This Class is designed to have

- Launch Browser (Code to Launch Browser Before Test Script Execution)
- Close Browser (Code to Close Browser After Test Script Execution)

Generics: This class is Designed to have Generic Codes that does not Change throughout the project and can be Used throughout Project by just Calling there Method Names like

- Screenshot Taker
- Date Picker
- Read and Write Excel . . .

Page Object Module: For Each and Every Web Page of Project a POM Class is designed which contains

- @FindBy (Path to Identify Element)
- Methods (to perform actions on the elements)

Test Scripts: Contains Code that Automate Each and Every Test Cases for Each and Every Module

Data Folder: Contains Excel Sheets Designed to have Data Required for Each and Every Module and Data is read by Using Apache POI

Screenshot Folder: Contains Screenshot of Web Page of Specific Module in a Sub-Folder named after the particular method which failed its Execution

Handling Pop-up's:

With respect to Selenium we can categorize the popup into following types:

1. Hidden Division Popup
2. Alert and Confirmation Popup
3. File Upload Popup
4. File Download Popup
5. Child Browser Popup
6. Window Popup

1. HIDDEN DIVISION POPUP

Characteristics:

- 1. We cannot move the popup
- 2. We can Inspect the popup
- 3. It will be colorful

Handling Technique: We can inspect these kinds of popup using driver.findElement (By.xpath ("xpath of the element"));

2. ALERT AND CONFIRMATION POPUP

Characteristics:

- We can move the popup
- We cannot inspect the popup
- If the popup has Alert Symbol (!) it is called as Alert Popup
- If the popup has Confirmation (?) Symbol, it is called as confirmation popup

Note: Both are called as javascript popup

3. FILE UPLOAD POPUP

Characteristics:

- Clicking on Browse button will display a popup with the title file upload.
- We can move the popup, but we cannot inspect it.
- Solution: To handle file upload popup we specify Absolute path of the file as an argument for sendKeys()

Handling Technique:
In this technique, we do not click on upload button,
To handle this Popup below are the steps followed:
1- Inspect and find Upload button
2- Use sendKeys() method and give absolute path of the file stored in local PC as and argument to sendKeys() method.
driver.findElement(By.xpath("//input[@value='Upload CV']")).sendKeys("D://sample resume.doc");

4. FILE DOWNLOAD POPUP

Characteristics:

- We can move the popup
- We cannot inspect the popup
- popup will have two radio button: Open with and Save file Solution
- To handle File Download Popup, we use setPreference() of FirefoxProfile class. Which will programmatically change the settings of the browser so that it will download the file automatically without displaying the popup?

5. CHILD BROWSER POPUP

Characteristics

- We can move the popup
- We can inspect the popup
- It will have minimize and maximize button with address bar.
  Handling Technique:
  Whenever we open any browser, one window handle id is allocated to that browser.
  So All opened browser (parent and child) are having their respective window handle ids
  To get the window handle id of current browser we use method
  driver.switchTo().window(window handle);
  Example- Switching to each child browsers

```
driver.get("https://www.naukri.com/");//Enter URL in browser

String parentwindow = driver.getWindowHandle(); //get parent window handle

Set<String> allWindows = driver.getWindowHandles(); //get all opened browsers window handle ids and store in a list

System.out.println(allWindows.size());//Print the count of all opened windows

allWindows.remove(parentwindow); //remove parent window id from All windows handle ids

for (String s : allWindows) //using for each loop, iterating to each window
```

```
{
System.out.println(s);
        driver.switchTo().window(s); //switching to child windows and perform action
       System.out.println(driver.getTitle());
        }
                //switch back to parent window
        driver.switchTo().window(parentwindow);
```

6. WINDOW POPUP

If the popup displayed in application is not Hidden Division, Alert and confirmation, File upload, File Download and child Browser popup. Then it is called as Window Popup. In Selenium we cannot handle window popup. And to handle the popup we should use autoIt.

autoIt is a free window based application automation tool, which can be downloaded from following Website:

url: https://www.autoitscript.com/site/autoit/downloads/

File: autoit-v3-setup.exe

To install it: Double click on the downloaded setup file, follow the default instructions given in the setup wizard (next.. next.. till finish).

Steps to inspect element in Autoit:

1. Go to Start > All Programs > Autoit > select Window Info
2. Drag and drop “finder tool” on required element or popup.

Steps to write code in AutoIt and Executing it:

1. Go to Start > All Programs > AutoIt > select Script Editor
2. Write the code as below

WinWaitActive(“Calculator”)

WinClose(“Calculalor”)

3. Go to file and select Save
4. Navigate to required location ex: D specify the name ex: script1 and click save extension will be .au3

5. Go to tools and select compile and it creates new .exe file (same location)

6. Double click to execute it

List of Exceptions**:**

1. InterruptedException (Java checked)

2. IllegalStateException (Java Unchecked)

3. NoSuchElementException (Selenium Unchecked)

4. IOException (Java checked)

5. TimeoutException (Selenium Unchecked)

6. UnreachableBrowserException (Selenium Unchecked)

7. InvalidElementStateException (Selenium Unchecked) – when we try to enter data in a disabled textbox.

8. IndexOutOfBoundsException (Java Unchecked)

9. NumberFormatException (java Unchecked)

10. NoSuchFrameException (selenium unchecked)

11. UnexpectedTagNameException (selenium unchecked)

12. StaleElementReference Exception (selenium unchecked)

13. TestNGException (TestNG unchecked

14. EncryptedDocumentException (Java unchecked)

15. Invalid FormatException (Java unchecked)

16. NoAlertPresentException (selenium unchecked)

17. SessionNotFoundException (Selenium unchked)

18. NoWindowException (Selenium Unchecked)

Limitations of Selenium Web driver:

1. We can automate only web applications

2. We can't minimize the browser

3. Using selenium we can't perform any action on the browser which is already opened. Every time it will open the new browser.

4. We can take screenshot only in png format. Other format is not supported.

5. We can't take screenshot of popup, multiple browsers and specific area.

6. We can't specify the password in encrypted format.

7. We cannot handle file upload popup if it has attachment icon instead of Browse button.

8. We cannot handle file download popup in the browsers other than Firefox

9. We cannot handle new tab in selenium.

10. We cannot handle window popup

Important selenium interview questions.

Core java interview questions.

1. What is an Object in Java?
2. What is a Class in Java?
3. What is Constructor in Java?
4. What are the OOPs concepts?
5. What is Inheritance in Java?
6. What is Polymorphism?
7. What are the types of Polymorphism?
8. What is Method Overloading?
9. What is Method Overriding?
10. What is Abstraction in Java?
11. What is Encapsulation in Java?

Automation testing interview questions

1. What is Automation Testing?
2. What is the latest version of selenium is Selenium
3. What are the languages supported by Selenium
4. What are the OS supported by selenium
5. What type of test cases we automate
6. Which test cases are automated first
7. How do you close the browser without using close() method
8. How do you open the page without using get() method
9. How do you click on back button
10. How do you refresh the page
11. What is the difference between get() and navigate() method
12. Is it possible to achieve 100% automation?
13. List out different types of locators

14. Explain Xpath with Types
15. What is the different between single forward slash and double forward slash
16. Derive an 'xpath' which matches with all the images present on the web page
17. Write an 'xpath' which matches with all the links and all the images present on the web page
18. When do we use contains function
19. What are the differences between 'Implicit' and 'Explicit' wait
20. How do you execute an exe file in Selenium
21. What are the different ways of clicking on a button
22. What are the difference between findElement() and findElements()
23. When do you prefer Automation Testing over Manual Testing
24. Difference between Manual Testing & Automation Testing
25. What type of tests have you automated?
26. What is Framework?
27. What are the advantages of the Automation framework?
28. What are the different types of frameworks?
29. Can you explain Automation framework?
30. Explain TestNG
31. Explain POM
32. Explain popup handling methods with examples

TESTNG Interview Questions

1. What is TestNG?
2. What are the advantages of TestNG?
3. What are the annotations available in TestNG?
4. What is TestNG Assert and list out common TestNG Assertions?
5. What is Soft Assert in TestNG?
6. What is Hard Assert in TestNG?
7. What is exception test in TestNG?
8. How to set test case priority in TestNG?
9. What is parameterized testing in TestNG?
10. How can we create data driven framework using TestNG?
11. How to run a group of test cases using TestNG?
12. How to disable a test case in TestNG ?
13. What are the different ways to produce reports for TestNG results?
14. What is the use of @Listener annotation in TestNG?
15. List out various ways in which TestNG can be invoked?
16. What are @Factory and @DataProvider annotation?