**Selenium Assignment**

1. How to Type in text box using Selenium Web Driver

> sendKeys()

– driver.findElement().sendKeys();

> JavascriptExecutor

– JavascriptExecutor jse = (JavascriptExecutor)driver;

– jse.executeScript(“document.getElementById(“input-email”).setAttribute(‘value’,’QAFox’)”);

2. How to read the placeholder value of a Text Box using getAttribute() method

getAttribute(String attributeName), as discussed above, is used to fetch the attribute value of an HTML tag of the web element. There are multiple attributes like style, src, placeholder, href, etc. which are passed as a string parameter to the method. The value corresponding to the parameter is returned.

3.Selecting the Radio Button by Label Text / value

Selenium includes different ways using which we can select the Radio elements on a Web Page.

The different methods are as below:

By ID attributes

Using Is Selected()

Using Name

Using element Value

By CSS Selector

With XPath

A:Selenium includes different ways using which we can select the Radio elements on a Web Page.

**The different methods are as below:**

1. By ID attributes
2. Using Is Selected()
3. Using Name
4. Using element Value
5. By CSS Selector
6. With XPath

4.Find out number of elements in a radio group

A: checkbox, we can select multiple options. Using **Click() method** in Selenium we can perform the action on the Radio button and on Checkbox. WebElement maleRadioBtn = driver. findElement (By.id("gender-male"))

1. Selecting the Radio Button by Label Text / value

We can do this by providing the radio's ID attribute value in the for attribute of the **<label> element**. Now, you can click the associated label to select the corresponding radio button. If your radio button doesn't have an id attribute, you can wrap the radio button directly inside the <label> element.

[2.Find out number of elements in a radio group?](https://www.techiedelight.com/select-radio-button-by-clicking-text/" \l ":~:text=We%20can%20do%20this%20by,select%20the%20corresponding%20radio%20button.&text=If%20your%20radio%20button%20doesn,inside%20the%20element.)

We can count the total number of radio buttons in a page in Selenium with the help of find\_elements method. While working on any radio buttons, we will always find an attribute type in the html code and its value should be radio.

3. Find out all radio button values

Ans:

1. isSelected(): Checks whether a radio button is selected or not.
2. isDisplayed(): Checks whether a radion button is displayed on the web page or not.
3. isEnabled(): Checks whether a radion button is enabled or not.
4. How to get the selected radio button label text?

We can do this by providing the radio's ID attribute value in the for attribute of the **<label> element**. Now, you can click the associated label to select the corresponding radio button. If your radio button doesn't have an id attribute, you can wrap the radio button directly inside the <label> element.

5. Check if Radio Button is selected?

Using Input Radio checked property: The Input Radio checked property is used to return the checked status of an Input Radio Button. Use **document.** **getElementById('id')**. checked method to check whether the element with selected id is check or not.

.6. Check if Radio Button is enabled or disabled?

1. isSelected(): Checks whether a radio button is selected or not.
2. isDisplayed(): Checks whether a radion button is displayed on the web page or not.
3. isEnabled(): Checks whether a radion button is enabled or not.

**Dropdown/List Box/Combo Box**

1. Print all the options available in the dropdown

We can extract all the options in a dropdown in Selenium with the help of Select class which has the **getOptions() method**. This retrieves all the options on a Select tag and returns a list of web elements. This method does not accept any arguments.11-Jun-2020

1. Print first selected option from a dropdown

*Ans:selectByIndex*

*selectByValue*

*selectByVisibleText*

3. Select an option by value from a dropdown

*Ans:*Import the package **org.openqa.selenium.support.ui.Select**

Instantiate the drop-down box as an object, Select in Selenium WebDriver

4. Select an option by visible text from a dropdown

We can get a selected option in a dropdown in Selenium webdriver. The **method getFirstSelectedOption() returns** the selected option in the dropdown. Once the option is fetched we can apply getText() method to fetch the text.

5. Select an option by index from a dropdown

selectByIndex - It is used to select an option based on its index, beginning with 0. dropdown.selectByIndex(5); ...

1. selectByValue - It is used to select an option based on its 'value' attribute. ...
2. selectByVisibleText - It is used to select an option based on the text over the option.
3. Clicking a link using partialLinkText

In order to access link using link text in Selenium, the below-referenced code is used:  
driver.findElement(By.linkText("this is a link text"));

**Note:** In a scenario where, multiple links having similar text exists, it will automatically select the first one.

2. clicking a link using link text in selenium

In order to access link using link text in Selenium, the below-referenced code is used: driver. findElement(By. linkText("this is **a** link text"))

4. clicking on an image link in selenium

We can get the source of an image in Selenium. An image in an html document has <img> tagname. Each image also has an attribute src which contains the source of image in the page. To fetch any attribute in Selenium, we have to use the **getAttribute() method**.

**Popups/Alerts and Windows**

1. Capturing the alert message using getText()

We can use ***getText()*** method to capture the text from Alert Message. It will return a String variable and is used commonly to fetch data. You can use following command to get data from alert message:

1. void dismiss() // To click on the 'Cancel' button of the alert. ...
2. void accept() // To click on the 'OK' button of the alert. ...
3. String getText() // To capture the alert message.
4. Confirmation Alert with Ok and Cancel buttons

Selenium has multiple APIs to handle alerts with an Alert interface. To click on the Ok button on alert, first of all we have to **switch to alert with switchTo()**. alert() method. Next, to click on the Ok button, we have to use accept() method.

1. Clicking OK button of the alert using accept()

Selenium has multiple APIs to handle alerts with an Alert interface. To click on the Ok button on alert, first of all we have to **switch to alert with switchTo()**. alert() method. Next, to click on the Ok button, we have to use accept() method.

1. Clicking Cancel button of the alert using dismiss()

***Void dismiss()***: We use this method when the '*Cancel*' button clicks in the alert box.

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

1. Handle single window using driver.getWindowHandle() getWindowHandle() returns **the window handle of currently focused window/tab**. getWindowHandles() returns all windows/tabs handles launched/opened by same driver instance including all parent and child window.
2. Switch to window usingdriver.switchTo().window() Get Ans:the handles of all the windows that are currently open using the command: **Set<String> allWindowHandles = driver**. getWindowHandles(); which returns the set of handles. Use the SwitchTo command to switch to the desired window and also pass the URL of the web page
3. Switch to frame using driver.switchTo().frame()\

switchTo()defaultContent() This method is for switching to and fro in between frames and parent frames. The focus is shifted to the main page.

switchTo().parentFrame() This method is used to switch the control to the parent frame of the current frame.

1. Switch to popup using driver.switchTo.alert()

* ***Void accept()***: This method clicks on the '*OK*' button of the alert box.

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

* ***Void dismiss()***: We use this method when the '*Cancel*' button clicks in the alert box.

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

* ***String getText()***: This method captures the message from the alert box.

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

* ***Void sendKeys****(String stringToSend)*: This method sends data to the alert box.

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

**Selenium Miscellaneous Scenarios**

* 1. Write a test case to capture the screenshots with WebDriver
  2. package captureScreenshot;
  3. // Import all classes and interface
  4. import java.io.File;
  5. import library.Utility;
  6. import org.openqa.selenium.io.FileHandler;
  7. import org.openqa.selenium.By;
  8. import org.openqa.selenium.OutputType;
  9. import org.openqa.selenium.TakesScreenshot;
  10. import org.openqa.selenium.WebDriver;
  11. import org.openqa.selenium.firefox.FirefoxDriver;
  12. import org.testng.ITestResult;
  13. import org.testng.annotations.AfterMethod;
  14. import org.testng.annotations.Test;
  15. public class FacebookScreenshot {
  16. // Create Webdriver reference
  17. WebDriver driver;
  18. @Test
  19. public void captureScreenshot() throws Exception
  20. {
  21. // Initiate Firefox browser
  22. driver=new FirefoxDriver();
  23. // Maximize the browser
  24. driver.manage().window().maximize();
  25. // Pass application url
  26. driver.get("http://www.facebook.com");
  27. // Here we are forcefully passing wrong id so that it will fail our testcase
  28. driver.findElement(By.xpath(".//\*[@id='emailasdasdas']")).sendKeys("Learn Automation");
  29. }
  30. // It will execute after every test execution
  31. @AfterMethod
  32. public void tearDown(ITestResult result)
  33. {
  34. // Here will compare if test is failing then only it will enter into if condition
  35. if(ITestResult.FAILURE==result.getStatus())
  36. {
  37. try
  38. {
  39. // Create refernce of TakesScreenshot
  40. TakesScreenshot ts=(TakesScreenshot)driver;
  41. // Call method to capture screenshot
  42. File source=ts.getScreenshotAs(OutputType.FILE);
  43. // Copy method specific location here it will save all screenshot in our project home directory and
  44. // result.getName() will return name of test case so that screenshot name will be same
  45. try{
  46. FileHandler.copy(source, new File("./Screenshots/"+result.getName()+".png"));
  47. System.out.println("Screenshot taken");
  48. }
  49. }
  50. catch (Exception e)
  51. {
  52. System.out.println("Exception while taking screenshot "+e.getMessage());
  53. }
  54. }
  55. // close application
  56. driver.quit();
  57. }
  58. }

1. Capturing the alert message using getText()
2. Alert a = driver.switchTo().alert();
3. String s= driver.switchTo().alert().getText();
4. a.accept();
5. 2. Prompt Alert with Text Box to enter the text

let person = prompt("Please enter your name", "Harry Potter");  
  
if (person != null) {  
  document.getElementById("demo").innerHTML =  
  "Hello " + person + "! How are you today?";

4. Clicking OK button of the alert using accept()

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.Alert;

public class AlertAccept{

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "C:\\Users\\ghs6kor\\Desktop\\Java\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

String url ="https://www.tutorialspoint.com/selenium/selenium\_automation\_practice.htm";

driver.get(url);

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

// identify element

driver.findElement(By.xpath("//button[@name='submit']")).click();

// Alert interface and switchTo().alert() method

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

// click on OK to accept with accept()

al.accept();

driver.quit();

}

}

5. Clicking Cancel button of the alert using dismiss()

**Void dismiss():** This method is used when the 'Cancel' button is clicked in the alert box.

6.Handle multiple windows using driver.getWindowHandles()

getWindowHandle( ): When a website opens, we need to handle the main window i.e the parent window using driver.getWindowHandle( ); method. With this method, we get a unique ID of the current window which will identify it within this driver instance. This method will return the value of the String type.

getWindowHandles( ): To handle all opened windows which are the child windows by web driver, we use driver.getWindowHandles( ); method. The windows store in a Set of String type and here we can see the transition from one window to another window in a web application. Its return type is Set <String>.

10. Switch to popup using driver.switchTo.alert()

switchto(): Using this method we perform switch operation within windows.

package Guru99TakeScreenshot;

import java.io.File;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.Test;

public class Guru99TakeScreenshot {

@Test

public void testGuru99TakeScreenShot() throws Exception{

WebDriver driver ;

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

driver = new FirefoxDriver();

//goto url

driver.get("http://demo.guru99.com/V4/");

//Call take screenshot function

this.takeSnapShot(driver, "c://test.png") ;

}

/\*\*

\* This function will take screenshot

\* @param webdriver

\* @param fileWithPath

\* @throws Exception

\*/

public static void takeSnapShot(WebDriver webdriver,String fileWithPath) throws Exception{

//Convert web driver object to TakeScreenshot

TakesScreenshot scrShot =((TakesScreenshot)webdriver);

//Call getScreenshotAs method to create image file

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

//Move image file to new destination

File DestFile=new File(fileWithPath);

//Copy file at destination

FileUtils.copyFile(SrcFile, DestFile);

}

}

1. Click on an element which is in iFrame

You need to use **switchTo().** **frame()** to access content within a frame or iframe

3. Find out the broken links on a web page

Collect all the links present on a web page based on the <a> tag.

Send HTTP request for each link.

Verify the HTTP response code.

Determine if the link is valid or broken based on the HTTP response code.

Repeat the process for all links captured with the first step.

5. Action class with the following operations

Keyboard key press event

Pressing enter button on the key board

ClickAndHold event, Drag and Drop

MoveToElement, Mouse Hover Event

Double Click event

|  |  |
| --- | --- |
| **Method** | **Description** |
| **clickAndHold()** | Clicks (without releasing) at the current mouse location. |
| **contextClick()** | Performs a context-click at the current mouse location. (Right Click Mouse Action) |
| **doubleClick()** | Performs a double-click at the current mouse location. |
| **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.  **Parameters:**  source- element to emulate button down at.  target- element to move to and release the mouse at. |
| **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.  **Parameters**:  source- element to emulate button down at.  xOffset- horizontal move offset.  yOffset- vertical move offset. |
| **keyDown(modifier\_key)** | Performs a modifier key press. Does not release the modifier key – subsequent interactions may assume it’s kept pressed.  **Parameters**:  modifier\_key – any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **keyUp(modifier \_key)** | Performs a key release.  **Parameters**:  modifier\_key – any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **moveByOffset(x-offset, y-offset)** | Moves the mouse from its current position (or 0,0) by the given offset.  **Parameters**:  x-offset- horizontal offset. A negative value means moving the mouse left.  y-offset- vertical offset. A negative value means moving the mouse down. |
| **moveToElement(toElement)** | Moves the mouse to the middle of the element.  **Parameters**:  toElement- element to move to. |
| **release()** | Releases the depressed left mouse button at the current mouse location |
| **sendKeys(onElement, charsequence)** | Sends a series of keystrokes onto the element.  **Parameters**:  onElement – element that will receive the keystrokes, usually a text field  charsequence – any string value representing the sequence of keystrokes to be sent |

7. Handling Ajax Auto suggestion

package Testng\_Pack;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class Ajax\_Handle {

WebDriver driver = null;

@BeforeTest

public void setup() throws Exception {

System.setProperty("webdriver.gecko.driver", "D:\\Selenium Files\\geckodriver.exe");

driver = new FirefoxDriver();

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

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

driver.get("http://www.google.com");

}

@AfterTest

public void tearDown() throws Exception {

driver.quit();

}

**//Data provider Is used for supplying 2 different values to Search\_Test method.**

@DataProvider(name="search-data")

public Object[][] dataProviderTest(){

return new Object[][]{{"selenium webdriver tutorial"},{"auto s"}};

}

@Test(dataProvider="search-data")

public void Search\_Test(String Search){

driver.findElement(By.xpath("//input[@id='gbqfq']")).clear();

driver.findElement(By.xpath("//input[@id='gbqfq']")).sendKeys(Search);

int i=1;

int j=i+1;

try{

**//for loop will run till the NoSuchElementException exception.**

for(i=1; i<j;i++)

{

**//Value of variable i Is used for creating xpath of drop list's different elements.**

String suggestion = driver.findElement(By.xpath("//\*[@id='gsr']/table/tbody/tr[1]/td[2]/table/tbody/tr["+i+"]/td/div/table/tbody/tr/td[1]/span")).getText();

System.out.println(suggestion);

j++;

}

}catch(Exception e){**//Catch block will catch and handle the exception.**

System.out.println("\*\*\*Please search for another word\*\*\*");

System.out.println();

}

}

8. Select a specific date from a calendar

import java.util.List;

import lombok.SneakyThrows;

import org.junit.jupiter.api.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.ui.ExpectedConditions;

public class DatePickerTestsFlyDubai extends BaseTest {

@Test

@SneakyThrows

public void flyDubaiDatePicker() {

//Go to website

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

//Click Departure Place

wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".airportPickerTo.makeBookingTo.mat-form-group"))).click();

//Wait Dropdown and click the first city

wait.until(ExpectedConditions.visibilityOfElementLocated(

By.cssSelector(" .makeBookingTo .search-list-dropdown [data-metro-active='false']:nth-of-type(1)"))).click();

//Wait the DatePicker Opens

wait.until(ExpectedConditions.visibilityOfElementLocated((By.cssSelector(".lightpick\_\_inner"))));

//This are the cell of the from date picker table for departure. Get all elements.

List<WebElement> cellsOfDepartureDate = wait.until(

ExpectedConditions.visibilityOfAllElementsLocatedBy(By.cssSelector("section:nth-of-type(1) > .lightpick\_\_days > div")));

//Click the today for Departure

DateUtil.clickGivenDay(cellsOfDepartureDate, DateUtil.getCurrentDay());

//This are the cell of the from date picker table for arrival. Get all elements.

List<WebElement> cellsOfArrivalDate = wait.until(

ExpectedConditions.presenceOfAllElementsLocatedBy(By.cssSelector("section:nth-of-type(2) > .lightpick\_\_days > div")));

//Click the 4th day (element).

cellsOfArrivalDate.get(4).click();

//Wait and see the selection.

Thread.sleep(5000);

}

}

**6. TestNG**

First of all, **beforeSuite() method is executed only once**. Lastly, the afterSuite() method executes only once. Even the methods beforeTest(), beforeClass(), afterClass(), and afterTest() methods are executed only once. beforeMethod() method executes for each test case but before executing the test case.

2.Create testing.xml file to run the test cases in a class file

1. **public** **class** test
2. {
3. @Test
4. **public** **void** test1()                      // First test case.
5. {
6. System.out.println("test1");
7. }
8. @Test
9. **public**  **void** test2()                     // Second test case.
10. {
11. System.out.println("test2");
12. }}
13. Create a test suite and test groups
14. Create a java test class, say, **GroupTestExample.java** in **/work/testng/src**.
15. Add test methods, testPrintMessage() and testSalutationMessage(), to your test class.
16. Group the test method in two categories −
    1. **Check-in tests (checkintest)** − These tests should be run before you submit new code. They should typically be fast and just make sure no basic functionality is broken.
    2. **Functional tests (functest)** − These tests should cover all the functionalities of your software and be run at least once a day, although ideally you would want to run them continuously.

There are two types of assertions in Selenium and the categorization depends on how the assertion behaves after a condition is pass or fail.

**Here, we would discuss two types of  assertions in**[**Selenium**](https://www.softwaretestinghelp.com/selenium-tutorial-1/)**:**

1. Hard Assertions
2. Soft Assertions
3. import org.openqa.selenium.WebDriver;
4. import org.testng.Reporter;
5. import org.testng.annotations.Test;
6. import org.openqa.selenium.chrome.ChromeDriver;
7. import org.testng.Assert;
8. public class TestNG {
9. WebDriver driver = new ChromeDriver();
11. @Test (priority = 0)
12. public void CloseBrowser() {
13. driver.close();
14. Reporter.log("Driver Closed After Testing");
15. }
17. @Test (priority = -1)
18. public void OpenBrowser() {
19. Reporter.log("This test verifies the current selenium compatibility with TestNG by launching the chrome driver");
20. Reporter.log("Launching Google Chrome Driver version 81 for this test");
22. driver.get("https://www.demoqa.com");
24. Reporter.log("The website used was DemoQA for this test", true);
25. String expectedTitle = "Free QA Automation Tools For Everyone";
26. String originalTitle = driver.getTitle();
27. Assert.assertEquals(originalTitle, expectedTitle);
28. }
29. }

The following code will demonstrate the use of soft asserts in TestNG. In this code, we are validating the title of the web page, bypassing two different expected titles.

import org.openqa.selenium.WebDriver;

import org.testng.asserts.SoftAssert;

import org.testng.Reporter;

import org.testng.annotations.Test;

import org.openqa.selenium.chrome.ChromeDriver;

public class TestNG {

WebDriver driver = new ChromeDriver();

@Test (priority = 0)

public void CloseBrowser() {

driver.close();

Reporter.log("Driver Closed After Testing");

}

@Test (priority = -1)

public void OpenBrowser() {

Reporter.log("This test verifies the current selenium compatibility with TestNG by launching the chrome driver");

Reporter.log("Launching Google Chrome Driver version 81 for this test");

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

SoftAssert softassert = new SoftAssert();

Reporter.log("The website used was DemoQA for this test", true);

String expectedTitle = "Free QA Automation Tools For Everyone";

String originalTitle = driver.getTitle();

softassert.assertEquals(originalTitle, expectedTitle);

System.out.println("\*\*\* Checking For The Second Title \*\*\*");

// Checking title for ToolsQA – Demo Website to Practice Automation – Demo Website to Practice Automation

softassert.assertEquals(originalTitle, "ToolsQA – Demo Website to Practice Automation – Demo Website to Practice Automation" );

softassert.assertAll();

}

}

1. Running TestNG scripts from command prompt
2. Open Eclipse and create a Java class
3. Write a Java program
4. Convert the Java Program into TestNG
5. Open command prompt
6. Run the TestNG using command prompt

7. Running test cases in parallel

**package** com.howtodoinjava.parallelism;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

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

**public** **class** ParallelMethodTest

{

@BeforeMethod

**public** **void** beforeMethod() {

**long** id = Thread.currentThread().getId();

System.out.println("Before test-method. Thread id is: " + id);

}

@Test

**public** **void** testMethodsOne() {

**long** id = Thread.currentThread().getId();

System.out.println("Simple test-method One. Thread id is: " + id);

}

@Test

**public** **void** testMethodsTwo() {

**long** id = Thread.currentThread().getId();

System.out.println("Simple test-method Two. Thread id is: " + id);

}

@AfterMethod

**public** **void** afterMethod() {

**long** id = Thread.currentThread().getId();

System.out.println("After test-method. Thread id is: " + id);

8.Create data driven framework using DataProvider

Selenium automates browsers. It’s a popular tool to automate web-based applications. To handle excel sheets to read and write data using Selenium we do use Apache POI.

Assume, you need to test login form with 50 different sets of test data

1. As a manual tester, you do log in with all the 50 different sets of test data for 50 times
2. As an automation tester, you create a test script and run 50 times by changing the test data on each run or you create 50 test scripts to execute the scenario

One of the reasons we take automation is to overcome the time consumption issue. By following the above two ways we don’t achieve anything with automation.