Selenium WebDriver Basics

# Installing Selenium WebDriver

Go to http://www.selenium.dev

2. Click On Downloads

3. Go to "Selenium Client & WebDriver Language Bindings" section.

4. In this section, click on Java Download.

5. Extract the file using WinZip/Winrar

6. Go to your java Project in eclipse.

7. Right click --> properties --> Java Build Path --> Libraries --> Add External Jars

8. Add all the jars in the selenium Folder.

Chrome driver path

[https://googlechromelabs.github.io/chrome-for-testing/#stable](https://googlechromelabs.github.io/chrome-for-testing/%23stable)

Selenium Workflow

1. Open the Browser
2. Navigate To Application
3. Take actions on the application

Open and Verification Commands

Manual TestCase:

1. Open application
2. Verify Page title

Expected Result: Title of the page must be ‘Create New Customer Account’

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

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

driver.get("https://magento.softwaretestingboard.com/customer/account/create/");

String actualTitle = driver.getTitle(); // getting from Facebook

String expectedTitle = "Create New Customer Account"; // Manual Testcase or Re doc

**if**(actualTitle.equals(expectedTitle)) { //Comparing actualTitle with expectedTitle

System.***out***.println("Test Case Passed");

}

**else** {

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

}

driver.close();

}

# Running the test on multiple browsers

WebDriver driver;

@BeforeTest

@Parameters("browser")

public void openBrowser(String browser) {

if(browser.equals("Chrome")) {

driver = new ChromeDriver();

}

else if(browser.equals("IE")) {

driver = new InternetExplorerDriver();

}

else if(browser.equals("Edge")) {

driver = new EdgeDriver();

}

else if(browser.equals("Firefox")) {

driver = new FirefoxDriver();

}

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

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

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

}

Copy the below code for all the browser in testing.xml file

<parameter name=*"browser"* value=*"Chrome"*></parameter>

Parallel Execution

Parallel execution is where the tests are run parallelly on all the configured browser.

Write the below line in testing.xml file at Suite level

<suite name=*"Suite"* parallel=*"tests"*>

# Webdriver Commands

|  |  |
| --- | --- |
| Command | Usage |
| get() | Navigate to URL |
| getTitle() | Get the page title |
| close() | Close browser |
| findElement(s)() | Find element(s) on the web page |
| getCurrentURL() | Get the URL |
| switchTo() | Switch to different window |

# Locators

Different locators in Selenium

Use of different types of locators

Creating a script using locators

Note:

Tag: is the name you see after “<” in chrome developer tool(usually in purple colour)

Attribute: are the ones having values(in brown color)

Attribute Values: blue color values for attributes.

To maximize the window 🡪 driver.manage().window().maximize();

Navigation Commands in Selenium

1. driver.navigate().to(“<https://facebook.com>”); 🡪 Similar to driver.get(“”);
2. driver.navigate().back();
3. driver.navigate().forward();
4. driver.navigate().refresh();

DropDown Box

To select value from the dropdown box, use the below code

Select dropdown = new Select(x);

Dropdown.selectbyVisibleText(“UK”);

1. SelectByVisibleText() 🡪 Provide the exact drop down value that you want to select
2. SelectByIndex() 🡪 Use this method when the values are countable in the dropdown box
3. SelectByValue() 🡪 Use this method when the text you need to select is lengthy. You can get the value from Chrome Developer tool.

WebElement option = dropdown.getFirstSelectedOption();

System.***out***.println(option.getText());

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

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.***SECONDS***);

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

driver.findElement(By.*linkText*("Create New Account")).click();

WebElement dropdownBox = driver.findElement(By.*id*("month"));

Select dd = **new** Select(dropdownBox);

//dd.selectByVisibleText("Mar");

//dd.selectByIndex(2);

dd.selectByValue("10");

}

Multi Select Box and Radio buttons

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

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

**driver.get("https://the-internet.herokuapp.com/");**

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

**driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);**

**driver.findElement(By.linkText("Checkboxes")).click();**

**boolean checkboxSelected = driver.findElement(By.cssSelector("#checkboxes > input[type=checkbox]:nth-child(1)")).isSelected();**

**if(checkboxSelected == false) {**

**driver.findElement(By.cssSelector("#checkboxes > input[type=checkbox]:nth-child(1)")).click();**

**}**

**}**

# Wait Statements in Selenium

1. Implicit Wait - It is the wait which is written just once in the entire script and it is applicable to all the ‘driver.findElement’ statements.

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(2));

1. Explicit Wait – Applicable only to one Web Element. Typically used with AJAX Requests

WebDriverWait wait = new WebDriverWait(driver,10);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#finish > h4")));

AJAX – Asynchronous JavaScript and XML

1. Fluent Wait – Explicit Wait + Control over polling period

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver).withTimeout(Duration.ofSeconds(30)).

pollingEvery(Duration.ofSeconds(2)).

ignoring(NoSuchElementException.class);

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

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

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

driver.get("https://the-internet.herokuapp.com/dynamic\_loading");

driver.manage().timeouts().implicitlyWait(30, TimeUnit.***SECONDS***);

driver.findElement(By.*partialLinkText*("that is hidden")).click();

driver.findElement(By.*cssSelector*("#start > button")).click();

String expectedResult = "Hello World!";

//WebDriverWait wait = new WebDriverWait(driver,10);

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

withTimeout(Duration.*ofSeconds*(30)).

pollingEvery(Duration.*ofSeconds*(2)).

ignoring(NoSuchElementException.**class**);

wait.until(ExpectedConditions.*visibilityOfAllElementsLocatedBy*(By.*cssSelector*(("#finish > h4"))));

String actualResult = driver.findElement(By.*cssSelector*("#finish > h4")).getText();

System.***out***.println(actualResult);

**if**(expectedResult.equals(actualResult)) {

System.***out***.println("Test Case Passed");

}

**else** {

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

}

//driver.close();

# Handling Alerts, Frames and Multiple windows

Handling Frames

1. To go inside a frame use the below line

driver.switchTo().frame("<FrameName>");

1. To come out of the frame use below line:

driver.switchTo().defaultContent();

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

driver.get("https://www.selenium.dev/selenium/docs/api/java/index.html");

driver.switchTo().frame("classFrame");

driver.findElement(By.linkText("DEPRECATED")).click();

driver.switchTo().defaultContent();

driver.switchTo().frame("packageFrame");

driver.findElement(By.linkText("GetExpression")).click();

}

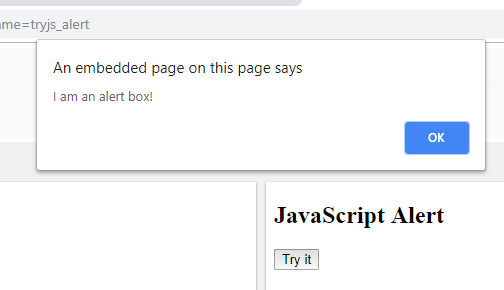
Alert Boxes

Alert boxes are created by Javascript and cannot be located using Chrome developer tool.

To click on Alert box, use the below code

Driver.switchTo().alert.accept();

Example of Alert Box:



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

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.***SECONDS***);

driver.get("https://the-internet.herokuapp.com/");

driver.findElement(By.*linkText*("JavaScript Alerts")).click();

driver.findElement(By.*cssSelector*("#content > div > ul > li:nth-child(1) > button")).click();

Thread.*sleep*(5000);

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

}

Multiple Windows

Selenium Keeps track of all the windows opened during a session. Existing windows or previous

execution windows are not counted.

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

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

driver.get(&quot;https://the-internet.herokuapp.com/windows&quot;);

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

String mainWindow = driver.getWindowHandle();

driver.findElement(By.linkText(&quot;Click Here&quot;)).click();

Set&lt;String&gt; allWindows = driver.getWindowHandles();

Object[] getAllWindows = allWindows.toArray();

String subWindowHandle = getAllWindows[1].toString();

driver.switchTo().window(subWindowHandle);

String actualResult = driver.findElement(By.cssSelector(&quot;body &gt; div &gt;

h3&quot;)).getText();

String expectedResult = &quot;New Window&quot;;

if(expectedResult.equals(actualResult)){

System.out.println(&quot;Test Case Passed&quot;);

}

else {

System.out.println(&quot;Test Case Failed&quot;);

}

driver.close();

driver.switchTo().window(mainWindow);

driver.close();

# Actions

Actions class is a collection of individual Action that you want to perform. For e.g. we may want to perform a mouse click on an element. In this case we are looking at two different Action

1. Moving the mouse pointer to the element
2. Clicking on the element

Complex interactions like Drag-n-Drop and Double-click which cannot be done by simple WebElement commands. To handle those types of advance actions we have the Actions class in Selenium.

**Different Methods for Performing Keyboard Events:**

* keyDown(modifier key): Performs a modifier key press.
* sendKeys(keys to send ): Sends keys to the active web element.
* keyUp(modifier key): Performs a modifier key release.

**Different Methods for performing Mouse Events:**

* **click():** Clicks at the current mouse location.
* **doubleClick():** Performs a double-click at the current mouse location.
* **contextClick() :** Performs a context-click at middle of the given element.
* **clickAndHold():** Clicks (without releasing) in the middle of the given element.
* **dragAndDrop(source, target):** Click-and-hold at the location of the source element, moves to the location of the target element
* **dragAndDropBy(source, xOffset, yOffset):**  Click-and-hold at the location of the source element, moves by a given offset
* **moveByOffset(x-offset, y-offset):** Moves the mouse from its current position (or 0,0) by the given offset
* **moveToElement(toElement):** Moves the mouse to the middle of the element
* **release():** Releases the depressed left mouse button at the current mouse location

Example of Key strokes

------------------perform copy paste operation------------------------

driver.get("https://magento.softwaretestingboard.com/customer/account/create/");

System.***out***.println(driver.getTitle());

driver.findElement(By.*id*("firstname")).sendKeys("John");

driver.findElement(By.*id*("password")).sendKeys("Aabcd@123");

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

actions.keyDown(Keys.***CONTROL***);

actions.sendKeys("a");

actions.keyUp(Keys.***CONTROL***);

actions.build().perform();

actions.keyDown(Keys.***CONTROL***);

actions.sendKeys("c");

actions.keyUp(Keys.***CONTROL***);

actions.build().perform();

actions.sendKeys(Keys.***TAB***);

actions.build().perform();

actions.keyDown(Keys.***CONTROL***);

actions.sendKeys("v");

actions.keyUp(Keys.***CONTROL***);

actions.build().perform();

Example of mouse action

To Hover on any WebElement, use the below code

Actions hover = new Actions(driver);

Hover.moveToElement(<WebElement>).perform();

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

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.***SECONDS***);

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

WebElement signIn = driver.findElement(By.*id*("nav-link-accountList"));

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

hover.moveToElement(signIn).perform();

driver.findElement(By.*linkText*("Your Lists")).click();

}

-------------Drag and Drop------------------

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

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

driver.get("https://the-internet.herokuapp.com/drag\_and\_drop");

WebElement element1 = driver.findElement(By.*id*("column-a"));

WebElement element2 = driver.findElement(By.*id*("column-b"));

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

a.dragAndDrop(element1, element2).perform();

}

# Capturing screenshots

**private** **void** takeScreenshot() **throws** IOException {

File ss = ((TakesScreenshot)driver).getScreenshotAs(OutputType.***FILE***);

FileHandler.*copy*(ss, **new** File("D:\\SelJul11\\screenshot"+(**new** Random().nextInt())+".jpg"));

}

Selenium 4 also has full page screenshot but this works only for firefox. ***getFullPageScreenshotAs()***

***File src = ((FirefoxDriver) driver).getFullPageScreenshotAs(OutputType.FILE);***

***FileHandler.copy(src, new File("FullPageScreenshot.png"));***

# Invocation count in TestNG

When you wish to run the same tests several times, you use invocation count. In TestNG, the @Test(invocationCount=x) annotation defines the number of times a test method is performed. The annotation's x value indicates how often the test method should be called.

@Test(invocationCount = 2, invocationTimeOut = 50000)

# Java script executor using selenium

Sometimes, Selenium webdriver faces problems in interacting with a few web elements, For example, the user opens a website and there is an unexpected pop-up window that will prevent the webdriver from performing operations and produce inaccurate results. This is where JavascriptExecutor comes into use.

JavaScriptExecutor is an interface that is used to execute JavaScriprt through selenium webdriver. JavaScript is a programming language that interacts with HTML in a browser.

## Steps to execute JavaScript:

* 1. import package

*Import org.openqa.selenium.JavascriptExecutor;*

* 1. Create a reference

*JavascriptExecutor js = (JavascriptExecutor) driver;*

* 1. Call the execute script method

*js.executeScript(script, args);*

Example of throwing an alert using JavaScript:

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("alert('This is from JS!');");

Performing scroll using JavaScript:

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

driver.get("https://www.selenium.dev/documentation/");

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("window.scrollBy(0,4000)");

# Page Object Modelling

It states that for a multi-page application, create a new class for each new page in the application.

* Update Account information
* Address book

WebElement username = driver.findElement(By.name(“username”)

Above line will be replaced with

@findBy(name=”username”)

Private WebElement uName;

Constructor for the pages

public Login(WebDriver driver) {

PageFactory.initElements(driver, this);

}

TestCases

# DataDriven Framework

DataDriven Framework 🡪 Separate the test data from the actual code.

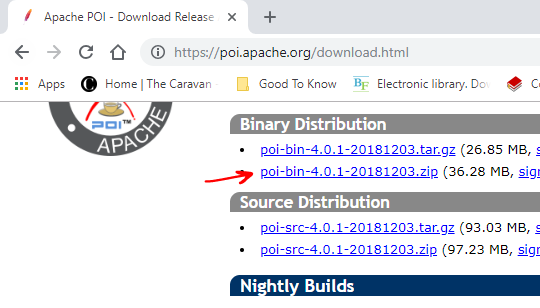
Data

**Selenium**

**Facebook**

**Response**

Apache POI jar files can be downloaded from –



S.No Username Password

1 [abc@gmail.com](mailto:abc@gmail.com) password1

2 bcd@gmail.com password2

3 cde@gmail.com password3

4 [def@gmail.com](mailto:def@gmail.com) password4

5 [edf@gmail.com](mailto:edf@gmail.com) password5

String[][] data = GenericMethods.*getData*("D:\\SelAug16\\TestData.xlsx", "Sheet1");

**for**(**int** i=1;i<data.length;i++) {

lp.applicationLogin(data[i][0], data[i][1]);

acc.clickTransferFunds();

tf.fundTransfer(data[i][2], data[i][3]);

tfv.clickSubmit();

String actualMsg = tfc.getMsg();

String expectedMsg = data[i][4];

Assert.*assertEquals*(actualMsg, expectedMsg);

tfc.logoutfromApplication();

driver.get("http://zero.webappsecurity.com/login.html");

}