SELENIUM WEBDRIVER

BASIC SETUP

```
System.SetProperty(K,V);
WebDriver driver = new ChromeDriver();
Driver.get();
```

KEY

```
Webdriver.chrome.driver
Webdriver.gecko.driver
Webdriver.io.driver
```

VALUE

Set the path of executable file

Illegal state exception throws when not set key value properly
Webdriver manager takes care by bonicarcia.
Selenium 4 → Selenium Manager takes care no need to set key value.

WEBDRIVER → I

METHODS

```
get("URL");
getCurrentUrl();
getTitle();
getPagesource()
close();
quit();
findElement();
findElements();
getWindowHandle();
getWindowHandles();
switchTo();
```

manage().
window().maximize()/minimize();
cookies();
deleteAllCookies();
navigate().
to(“URL”);
forward();
back();
refresh();

SET SIZE OF WINDOW

```
Dimension d = new Dimension(100,100);
driver.manage().window().setSize(d);
```

SET POSITION OF WINDOW

```
Point p = new Point ();
driver.manage().window().setPosition(p);

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

WEBELEMENT → I

METHODS

```
Sendkeys(“”);
Click();
Clear();
getText();
getAttribute(“attributename”);
getValue(“value”);
getTagName();
getCssValue();
```

getLocation();
getSize();
findElement();
findElemnents();

isSelected();
isEnabled();
isDisplayed();
submit();

GET LOCATION AND SIZE

Webelemnt wref.

Dimension d = Wref.getSize();
d.getHeight():
d.width();

Point p = wref.getSize();
p.getX():
p.getY();

GET CSS VALUE

String css = wref.getCssValue(“ ”);
font-size
colour
font-family
background
font-weight

LOCATORS

By is the abstract class , where all methods are static
Is used to identify the webelements in Dom Html elements.

id();
name();
className();
tagName()
linkText();
partialLinkText();
cssSelector();
xpath();

XPATH

ABSOLUTE IS FINDING FROM ROOT TO CONTEXT NODE
RELEATIVE IS FINDING LOCATING DIRECT CONTEXT NODE

XPATH AXES

CSS SELECTOR

:: POINTING THE RIGHT HTML ELEMNTS ONLY CAN GIVE PROPER WEBELEMENT.

READ ABOUT
HTML ELEMENTS LIKE DIV,SPAN,LABEL,ETC…

CODE NUMBER OF LINKS IN A WEBPAGE

```
LAUNCH BROWSER
GET URL
List<WebElements> lw  = driver.findElemnts(By.tagName("a"));
Int count = lw.size();
```

```
        For (webelemt w : lw){
        String s = w.getAttribute("href");
        System.out.println(s);
}
```

DROPDOWN

```
Select s = new Select(wref);
```

```
SelectByIndex(int)
SelectByValue(str);
selectByVisibleText(str);
deSelectByIndex(int)
deSelectByValue(str);
deselectByVisibleText(str);
deselectall();
```

getoptions(); →List<web>
getFirstSelectedValue();→webelement } used to
getallselectedoption();→List<web> getText(),getAttribute();

isMultiple();→boolean …means drop down option can select multiple.

JAVA SCRIPT EXECUTOR→ I

```
executeScript();
executeAsyncScript();
```

```
JavaScriptExecutor  js =   (JavascriptExecutor) driver ;
```

```
Js.executescript ( "js code " , webref optional);
```

JS CODES:

Send values

“arguments[0].setAttribute(‘value’, “anu”);

Get value

Object obj = “return arugements[0].getAttribute(‘value’)”
String s = obj.tostring();
Sysout(s);

Click

“arguments[0]. Click()”

Scroll up

“arguments[0]. scrollIntoView(false)”

Scroll down

“arguments[0]. scrollIntoView(true)”

Horizontal scroll

“window. scrollBy(500,0)”
“window. scrollBy(0,500)”

Eg.

Js.excuteScript (“arguments[0]. scrollIntoView(true), “arguments[0]. scrollIntoView(false), w1, w2)”
Here→w1,w2 are index line

arguments[0] arguments[1] are indicates of web element index.

JavascriptExecutor consists of two methods that handle all essential interactions using JavaScript in Selenium.

1. **executeScript method** – This method executes the test script in the context of the currently selected window or frame. The script in the method runs as an anonymous function. If the script has a return statement, the following values are returned:
    1. For an HTML element, the method returns a **WebElement.**
    2. For a decimal, the method returns **Long.**
    3. For a non-decimal number, the method returns **Long.**
    4. For a Boolean, the method returns **Boolean.**
    5. For other cases, the method returns a **String.**
2. **executeAsyncScript method** – This method executes the asynchronous piece of JavaScript on the current window or frame. An asynchronous script will be executed while the rest of the page continues parsing, which enhances responsiveness and application performance.

EX:

```
[java]

//importing the package

Import org.openqa.selenium.JavascriptExecutor;

//creating a reference

JavascriptExecutor js = (JavascriptExecutor) driver;

//calling the method

js.executeScript(script, args);

[/java]
```

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

```
[java]

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

[/java]
```

- **JavascriptExecutor in Selenium to send text**

```
[java]

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

[/java]
```

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

```
[java]

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

[/java]
```

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

```
[java]

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

[/java]
```

PRO1

```
[java]

package newpackage;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.Test;

import org.openqa.selenium.JavascriptExecutor;

public class LoginAutomation {

@Test



public void login() {

System.setProperty("webdriver.chrome.driver", "path");

WebDriver driver = new ChromeDriver();

JavascriptExecutor js = (JavascriptExecutor)driver;
```

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

driver.get("https://www.browserstack.com/users/sign_in");

js.executeScript("document.getElementById('user_email_login').value='rbc@xyz.com';");

js.executeScript("document.getElementById('user_password').value='password';");

js.executeScript("document.getElementById('user_submit').click();");

js.executeScript("alert('enter correct login credentials to continue');");

sleep(2000);

}



public static void sleep(int ms)

{

try



{

Thread.sleep(ms);

}

catch(InterruptedException e)



{

e.printStackTrace();

}



}

}

[/java]
```

PRO2

```
[java]

package newpackage;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.Test;



import org.openqa.selenium.JavascriptExecutor;

public class LoginAutomation {

@Test

public void login() {

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

WebDriver driver = new ChromeDriver();

JavascriptExecutor js = (JavascriptExecutor)driver;

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

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

js.executeAsyncScript("window.scrollBy(0,document.body.scrollHeight)");

}

}

[/java]
```

//Navigate to new Page i.e to generate access page. (launch new url)

```
js.executeScript("window.location = 'https://demo.guru99.com/'");
```

//To generate Alert window using JavascriptExecutor. Display the alert message

```
js.executeScript("alert('Welcome to Guru99');");
```

To scroll to the Bottom of the Web Page using Selenium WebDriver:

```
js.executeScript("window.scrollBy(0,document.body.scrollHeight)");
```

PROGRAM EX:

```
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;
public class geeksforgeeks {
	public static void main(String args[])
	{
		System.setProperty(
			"webdriver.edge.driver",

	"C:\\Users\\ADMIN\\Documents\\Selenium\\msedgedriver.exe");

			// Instantiate a Driver class.
			WebDriver driver = new EdgeDriver();

			// Maximize the browser
			driver.manage().window().maximize();

			// Launch Website
			driver.get("https://www.geeksforgeeks.org/");

			WebElement java = driver.findElement(
				By.xpath("//*[@id=\"hslider\"]/li[6]/a"));

			// Create a reference
```

```
            JavascriptExecutor js = (JavascriptExecutor)driver;

            // Call the JavascriptExecutor methods
            js.executeScript("arguments[0].click();", java);
        }
}
```

HIGHLIGHT THE WEBELEMENT

Js.excuteScript(“arguments[0].setAttribute(‘style’,’background:red’)”, wref1));

How to find an element in JavaScript?

1. document.getElementById("myId").
2. document.getElementsByTagName("div").
3. document.getElementsByClassName("myClass").
4. document.querySelector("div.myClass").
5. document.querySelectorAll("div.myClass").
6. document.getElementsByName("myName").

Syntax

```
1. document.querySelector(<<cssSelector>>);
```

Usage in Selenium code

```
1. String javascript = "document.querySelector('a')";
2. JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;
3. WebElement element = (WebElement) jsExecutor.executeScript(javascript);
```

TAKESSCREENSHOT : I

TakesScreenshot – I
getScreenshot() – m

TakesScreenshot ts = (TakesScreenshot) driver;
File src = ts.getScreenshotAs(OutputType.FILE);
File dest = new
File(“`"C:/Users/Chait/Desktop/Screenshots/screenshot01.png"`**”);**

FileUtils.copyFile(src,dest);

ex;

```
File file = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

String screenshotBase64 = ((TakesScreenshot)driver).getScreenshotAs(OutputType.BASE64);

X getScreenshotAs(OutputType(X). target) throws WebDriverException
```

EX PRO:

```
package BrowserstackScreenShot;

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 BStackTakeScreenshot {

@Test

public void testBStackTakeScreenShot() throws Exception{



WebDriver driver ;

System.setProperty("webdriver.firefox.marionette","c:\\geckodriver.exe");

driver = new FirefoxDriver();

//goto url

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

//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);

}

}
```

```
public class ScreenshotDemo {
    public static void main(String[] args) {
        //set the location of chrome browser
        System.setProperty("webdriver.chrome.driver", "C:\\chromedriver.exe");

        // Initialize browser
        WebDriver driver = new ChromeDriver();

        //navigate to url
        driver.get("https://demoqa.com");

       //Take the screenshot
        File screenshot = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

        //Copy the file to a location and use try catch block to handle exception
        try {
            FileUtils.copyFile(screenshot, new
File("C:\\projectScreenshots\\homePageScreenshot.png"));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        //closing the webdriver
        driver.close();
```

```
    }
}
```

```
static final OutputType<String>
```
BASE64

Obtain the screenshot as base64 data.

```
static final OutputType<byte[]>
```
BYTES

Obtain the screenshot as raw bytes.

```
static final OutputType<File>
```
FILE

Obtain the screenshot into a temporary file that will be deleted once the JVM exits.

DEPENDENCYS::

```
<!-- https://mvnrepository.com/artifact/ru.yandex.qatools.ashot/ashot -->
<dependency>
   <groupId>ru.yandex.qatools.ashot</groupId>
   <artifactId>ashot</artifactId>
   <version>1.5.4</version>
</dependency>
```

PROGRAM EX:

```
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import ru.yandex.qatools.ashot.AShot;
import ru.yandex.qatools.ashot.Screenshot;
import ru.yandex.qatools.ashot.shooting.ShootingStrategies;
import javax.imageio.ImageIO;
import java.io.File;

public class TakeFullPageScreenShot
{
  public static void main(String args[]) throws Exception
  {
```

System.setProperty("webdriver.chrome.driver" , “C:\\drivers\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
driver.get("https://www.lambdatest.com");

Screenshot screenshot = new AShot().shootingStrategy(ShootingStrategies.viewportPasting(1000)).takeScreenshot(driver);

ImageIO.write(screenshot.getImage(),"PNG",new File(“C:\\fullpagescreenshot.png"));

}

SELENIUM 4 :

```
public class CaptureWebelementScreenshot {

public static void main(String[] args) throws Exception {
// TODO Auto-generated method stub
WebDriver driver;
System.setProperty("webdriver.chrome.driver", "E:\\Selenium-
Softwares\\chromedriver.exe");
//System.setProperty("webdriver.gecko.driver", "E:\\Selenium-
Softwares\\geckodriver.exe");
driver=new ChromeDriver();
driver.get("http://demo.automationtesting.in/Register.html");
Thread.sleep(3000);
WebElement element = driver.findElement(By.id("imagetrgt"));

File source = ((TakesScreenshot)element).getScreenshotAs(OutputType.FILE);
FileHandler.copy(source, new File("./screenshots/element.png"));
Thread.sleep(3000);
driver.quit();

}
```

Handle Windows and Web-based Alerts/Popups in Selenium WebDriver

1. Windows-based alert pop-ups
2. Web-based alert pop-ups

What is Alert box/ Pop up box/ confirmation Box/ Prompt/ Authentication Box?

Handling web-based pop-up box

Import org.openqa.selenium.Alert

Switch to Alert
Driver.switchTo().alert();

Object Creation for Alert class
Alert alert = driver.switchTo().alert();

Accept the Alert
alert.accept();
Reject the Alert
alert.dismiss();

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

System.out.println(driver.switchTo().alert().getText());
driver.switchTo().alert().accept();
```

Window Based Pop-Ups

let's handle a window based pop up using Robot class.

import java.awt.Robot
import java.awt.event.KeyEvent

Object Creation for Robot class
Robot rb =new Robot();

KeyPress and KeyRelease Events

rb.keyPress(KeyEvent.VK_S);
rb.keyRelease(KeyEvent.VK_S);

rb.keyPress(KeyEvent.VK_D);
rb.keyRelease(KeyEvent.VK_D);

- **Robot class** is a java based utility which emulates the keyboard and mouse actions and can be effectively used to handling window based pop up with the help of keyboard events.

- The keyPress and keyRelease methods simulate the user pressing and releasing a certain key on the keyboard respectively.

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

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

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

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

```
  // Explicitly wait for the alert to be present.
        WebDriverWait wait = new WebDriverWait(driver, 10);
        wait.until(ExpectedConditions.alertIsPresent());

        // Switch to the alert.
        org.openqa.selenium.Alert alert = driver.switchTo().alert();

        // Get the text from the alert and print it.
        System.out.println("Alert Text: " + alert.getText());

        // Accept the alert (clicking OK).
        alert.accept();

        // Close the browser.
        driver.quit();
```

USING SEND KEYS TO DIRECT ACTION IN WEBPAGE

```
from selenium.webdriver.common.keys import Keys

driver.find_element_by_name("Value").send_keys(Keys.ENTER)
```

Action class in Selenium

Actions class is an ability provided by Selenium for handling keyboard and mouse events.
In Selenium WebDriver, handling these events includes operations such as drag and drop in Selenium, clicking on multiple elements with the control key, among others.

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

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

Mouse Actions in Selenium:

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

Keyboard Actions in Selenium:

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

Methods of Actions Class

Actions class is broadly divided into two categories:

1) Mouse actions

2) Keyboard actions

Mouse Actions

Mouse actions in Selenium are the actions that can be performed using a mouse, such as clicking, double-clicking, right-clicking, dragging and dropping, etc. These actions simulate a user's interactions with a website through the mouse.

The Actions class in Selenium WebDriver provides the following mouse action:

1) click(): performs a single mouse click on the specified element.

2) clickAndHold(): holds down the left mouse button on the specified element.

3) contextClick(): performs a right-click on the specified element.

4) doubleClick(): performs a double-click on the specified element.

5) dragAndDrop(): performs a drag and drop operation between two elements.

6) release(): releases the left mouse button on the specified element.

7) moveToElement(): moves the mouse cursor to the middle of the specified element.

Keyboard Actions

Keyboard actions in Selenium are the actions that can be performed using a keyboard, such as pressing keys, holding down keys, releasing keys, etc. These actions simulate a user's interactions with a website through the keyboard.

The Actions class in Selenium provides the following keyboard actions:

1) sendKeys(CharSequence… keysToSend): sends a series of key presses to the specified element.

2) keyDown(Keys theKey): holds down the specified key.

3) keyUp(Keys theKey): releases the specified key.

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

Act.moveToElement(wref);
Act.dragAndDrop(srcwref,destwref);
Act.doubleClick(wref);
Act.senkeys(wref,"valuepass");
Act.click();
Act.contextClick();
```

DARG AND DROP

Act.clickAndHold(src).moveToElement(dest).release().perform();

Key Takeaways

- *Keyboard events are the events that any of the Keyboard keys generate.*
- *Additionally, they are a must to simulate the user behavior while automating a web application using Selenium WebDriver*
- *The Actions Class of Selenium WebDriver provides - sendKeys(),keyUp(),keyDown() methods to handle various keyboard actions*
- *The modifier key is never released implicitly after the keyDown() method - either we should call the keyUp(theKey) or sendKeys(Keys.NULL) to release the modifier.*

Key down

```
new Actions(driver)
                .keyDown(Keys.SHIFT)
                .sendKeys("a")
                .perform();
```

Key up

```

new Actions(driver)
                .keyDown(Keys.SHIFT)
                .sendKeys("a")
                .keyUp(Keys.SHIFT)
                .sendKeys("b")
                .perform();
```

Send keys

Active Element

```

new Actions(driver)
                .sendKeys("abc")
                .perform();
```

Designated Element

```

new Actions(driver)
                .sendKeys(textField, "Selenium!")
                .perform();
```

Copy and Paste

```

Keys cmdCtrl = Platform.getCurrent().is(Platform.MAC) ? Keys.COMMAND : Keys.CONTROL;

        WebElement textField = driver.findElement(By.id("textInput"));
        new Actions(driver)
                .sendKeys(textField, "Selenium!")
                .sendKeys(Keys.ARROW_LEFT)
```

```
                    .keyDown(Keys.SHIFT)
                    .sendKeys(Keys.ARROW_UP)
                    .keyUp(Keys.SHIFT)
                    .keyDown(cmdCtrl)
                    .sendKeys("xvv")
                    .keyUp(cmdCtrl)
                    .perform();

            Assertions.assertEquals("SeleniumSelenium!",
    textField.getAttribute("value"));
```

Robot Class Methods in Selenium

Some commonly and popular used methods of Robot Class during web automation:

Method 1: keyPress():

robot.keyPress(KeyEvent.VK_DOWN): This method with press down arrow key of Keyboard

Method 2: mousePress():

robot.mousePress(InputEvent.BUTTON3_DOWN_MASK): This method will press the right click of your mouse.

Method 3: mouseMove():

robot.mouseMove(point.getX(), point.getY()): This will move mouse pointer to the specified X and Y coordinates.

Method 4: keyRelease():

robot.keyRelease(KeyEvent.VK_DOWN): This method with release down arrow key of Keyboard

Method 5: mouseRelease():

robot.mouseRelease(InputEvent.BUTTON3_DOWN_MASK): This method will release the right click of your mouse

```
Robot robot = new Robot();
      robot.keyPress(KeyEvent.VK_H);
      robot.keyPress(KeyEvent.VK_E);
      robot.keyPress(KeyEvent.VK_L);
      robot.keyPress(KeyEvent.VK_L);
      robot.keyPress(KeyEvent.VK_O);
```

Robot robot = new Robot();
robot.keyPress(KeyEvent.VK_CONTROL);
robot.keyPress(KeyEvent.VK_C);
robot.keyRelease(KeyEvent.VK_C);
robot.keyRelease(KeyEvent.VK_CONTROL);

Robot robot = new Robot();
robot.mouseMove(100, 100);

Robot robot = new Robot();
robot.mouseWheel(3); //scroll up by 3 units

```
public static void  execution() throws InterruptedException, AWTException {
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("http://spreadsheetpage.com/index.php/file/C35/P10/"); // sample url
        Robot robot = new Robot();
        robot.mouseMove(630, 420); // move mouse point to specific location
        robot.delay(1500);        // delay is to make code wait for mentioned
milliseconds before executing next step
        robot.mousePress(InputEvent.BUTTON1_DOWN_MASK); // press left click
        robot.mouseRelease(InputEvent.BUTTON1_DOWN_MASK); // release left click
        robot.delay(1500);
        robot.keyPress(KeyEvent.VK_DOWN); // press keyboard arrow key to select Save
radio button
        Thread.sleep(2000);
        robot.keyPress(KeyEvent.VK_ENTER);
        // press enter key of keyboard to perform above selected action
    }
```

Handle Multiple Tabs and Windows Using Selenium

help of the Action class:

Actions actions = new Actions(driver);
actions.keyDown(Keys.CONTROL).sendKeys(“t”).keyUp(Keys.CONTROL).perform();

help of the Robot framework:

Robot robot = new Robot();
robot.keyPress(KeyEvent.VK_CONTROL);

```
robot.keyPress(KeyEvent.VK_T);
robot.keyRelease(KeyEvent.VK_CONTROL);
robot.keyRelease(KeyEvent.VK_T);
```

help of Selenium 4:

```
driver.get("https://www.google.com/");
// Opens a new Tab and switches to new Tab
driver.switchTo().newWindow(WindowType.TAB);
// Opens TestSigma homepage in the newly opened window
driver.navigate().to("https://www.testsigma.com/");
```

help of JavascriptExecutor:

```
ChromeDriver driver = new ChromeDriver (); // Chromedriver or any other
Driver JavascriptExecutor jse = (JavascriptExecutor)driver;
js.executeScript("window.open()");
```

launch a website in a new tab

```
ChromeDriver driver = new ChromeDriver (); // Chromedriver or any other
Driver JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("window.open("https://www.google.com/")");
```

```
Robot r = new Robot();
    r.keyPress(KeyEvent.VK_CONTROL);
    r.keyPress(KeyEvent.VK_T);
    r.keyRelease(KeyEvent.VK_CONTROL);
    r.keyRelease(KeyEvent.VK_T);
```

HANDLING WINDOWS

METHODS

1. **getWindowHandle()**
2. **2. getWindowHandles()**

String currentHandle= driver.getWindowHandle();

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

driver.close(); //It will close on the focused window or tab
driver.quit(); //It will quit the entire browser session.

String currentHandle= driver.getWindowHandle();
driver.switchTo().window(currentHandle);

THREE PARAMETER CAN USE:

driver.switchTo().window(STRING URL);
driver.switchTo().window(STRING TITLE);
driver.switchTo().window(STRING WINDOW ID); // BEST PRACTICES

PROGRAM

TO HANDLE TWO WINDOW

1. **HOME PAGE OF URL**

String parwinid = driver.getWindowHandle();
Set<String> allwinid = driver.getWindowHandles():

For(String x : allwinid){
If(!parwinid.equals(x))

```
        {
        Sysout("child"+ x);
        driver.switchTo().window(x);

        }
        driver.switchTo().defaultcontent(); //to move previous window

    }

    Another way
    For (String x: allwinid) {

    If ( parwinid != x)
        {
        Sysout ("child"+ x);
        driver.switchTo().window(x);

        }
    }
```

TO HANDLE MULTIPLE WINDOW

2. HOME PAGE OF URL

```
String parwinid = driver.getWindowHandle();
Set<String> allwinid = driver.getWindowHandles():
```

SET<STRING > TO LIST <STRING >

```
List<String> li = new ArrayList<String>( );

li.addAll(allwinid);
```

(OR)

```
List<String> li = new ArrayList<String>(allwinid);
Int count = li.size();
//know the count of window

For(int i =0 , i<count,i++){

String Tab = li.get(1); //after the main window first
Driver.switchTo().window(Tab);

String Tab2 = li.get(2); //after the main window second
Driver.switchTo().window(Tab2);

}
```

ANOTHER WAY

```
PARWINID
ALLWINID

Int count = 0;
For (String x: Allwinid) {
If (count==1)      {

        driver.switchTo().window(x);

}
```

iFrame in Selenium Webdriver

iFrame (inline frame) is a HTML document embedded within another HTML document.

iFrames are most commonly used for displaying advertisements within a web page. iFrames are explicitly mentioned on HTML document using the HTML tag <iframe>.

Int size = driver.findElements(By.tagName("iframe")).size();

How to Handle Frames in Selenium using WebDriver Commands

Basically, we can switch over the elements and handle frames in Selenium using 3 ways.

- **By Index**
- **By Name or Id**
- **By Web Element**

How to switch back to the Main Frame

driver.switchTo().parentFrame();
driver.switchTo().defaultContent();

1. **switchTo.frame(int *frame number*)**: Defining the frame index number, the Driver will switch to that specific frame
2. **switchTo.frame(string *frameNameOrId*):** Defining the frame element or Id, the Driver will switch to that specific frame
3. **switchTo.frame(WebElement *frameElement*):** Defining the frame web element, the Driver will switch to that specific frame

EX:

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

System.setProperty("webdriver.gecko.driver","D:\\Data_Personal\\Demo \\geckodriver-
v0.23.0-win64\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.softwaretestinghelp.com/");

//Finding all iframe tags on a web page
List<WebElement> elements = driver.findElements(By.tagName("iframe"));
int numberOfTags = elements.size();
System.out.println("No. of Iframes on this Web Page are: " +numberOfTags);

// Switch to the frame using the frame id
System.out.println("Switching to the frame");
driver.switchTo().frame("aswift_0");

// Print the frame source code
System.out.println("Frame Source" +driver.getPageSource());

// Switch back to main web page
driver.switchTo().defaultContent();

driver.quit();
```

```
}
}
```

Handling Dynamic Frames in Selenium

- In some web pages, frame properties such as frame id and frame name can change dynamically on a web page, however, the frame position will remain the same. In such a case, we can't rely on the frame id or frame name to uniquely identify a frame.
- We can use the **frame index** in such a case to uniquely identify the frame based on the frame position.
- In some cases, frame id value changes every time when the page is loaded, but with a static text that does not change.

Conclusion

- iFrame is a HTML document embedded within another HTML document. iFrames are explicitly mentioned on the HTML document using the HTML tag <iframe>.
- switchTo.frame(int frameNumber) method allows the users to switch to a particular frame using the frame id.
- switchTo.frame(string frameName) method allows the users to switch to a particular frame using the developer-defined name of the frame.
- switchTo.frame(WebElement frameElement) method allows the users to switch to a frame based on the location of the Web Element

Web Tables

Web Tables are like normal tables where the data is presented in a structured form using rows and columns. The only difference is that they are displayed on the web with the help of HTML code.

<table> is the HTML tag that is used to define a web table. While **<th>** is used for defining the header of the table, **<tr>** and **<td>** tags are used for defining rows and columns respectively for the web table.

```
<table>
 <tr>
  <th>First Name</th>
  <th>Last Name</th>
  <th>Age</th>
 </tr>
 <tr>
  <td>Jill</td>
  <td>Ann</td>
  <td>24</td>
 </tr>
 <tr>
  <td>Eve</td>
  <td>Anderson</td>
  <td>34</td>
 </tr>
</table>
```

Static Web Tables

Dynamic Web Tables

```
public class WebTab {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();


// find web element on particular data in web table

WebElement w1 = driver.findElement(By.id(""));

String text = w1.getText();

System.out.println(text);



// for get all data in webTable using for each loop

List<WebElement> trows = driver.findElements(By.tagName(""));

for (WebElement rows : trows) {

System.out.println(rows.getText());

List<WebElement> tdata = driver.findElements(By.tagName(""));

for (WebElement celldata : tdata) {

System.out.println(celldata.getText());



// above same can be iterate over the for loop



// TO PRINT ONLY PARTICULAR DATA ONLY in dynamic table

List<WebElement> trows1 = driver.findElements(By.tagName(""));

for (int i = 0; i < trows1.size(); i++) {
```

```
List<WebElement> tdata1 = driver.findElements(By.tagName(""));

for (int j = 0; j < tdata1.size(); j++) {

if (tdata1.get(j).getText().equals("")) {

System.out.println(tdata1.get(j).getText());

}

}

}

}

}

}

}
```

WAITS

UNCONDITIONAL WAIT

CONDITIONAL WAIT

1. UNCONDITIONAL WAIT

Thread. Sleep (3000);

3000ms → millisecond

2. CONDITIONAL WAIT

Implicit wait

Explicit wait

IMPLICIT WAIT

```
driver.manage().timeouts().implicitlyWait(Duration.ofMinutes(2));
```

its common for entire class /session/locators

EXPLICIT WAIT

WEBDRIVER WAIT

FLUENT WAIT