Manual Testing:

==========================

Account  Creation form → 100 fields, drop downs, input text box, checkboxes, radio button, button, submit button

→ scenario:

> creating a user account in various cities of various states

> Create user account creation in various regions

> create user accounts with a specific package

> Create user account with specific city

> creating user account with different phone number

> Create account and activate account various cities of various states

> Create account and activate account  in various regions

> Create account and activate account with a specific package

Same scenarios were distributed across different testers

Execute and repeat scenarios on different browsers

1 tester → 10 days

Repeat → 40 days → to complete the testing → time taken was more

There are chances that we find out that some requirements have been missed or not developed towards the end of testing phase. SO time taken to develop and deliver the application may be more or applications delivery may be delayed

Just because we are doing manual testing → Cost will be more for repetitive task

Because the task are repetitive, some bugs are overlooked by testers

We may make mistakes while doing repetitive tasks.

Regression testing was also done manually :

==========================

Executing a set of test case to check that your existing code is working as it is after the new build has been deployed in the test environment.

Sprint1 → login → tested -> webpage working fine

Sprint 2 → logout →deployed in test env → login & logout → Smoke/Sanity testing → Regression testing → execute set of test case to check if existing feature is working fine or not, the new code has not impacted the exciting code → functional testing

Automation testing:

=====================================

> Using a tool to perform your manual tasks and time take to complete the tasks will be less.

> It has become popular as we can use a tool to perform repetitive tasks for us.

> we have 2 types of automation tools:

       > Functional automation testing tools

> Selenium , QTP

                       > selenium + testNG + cucumber

                       > Junit - selenium

                       >  cypress

> appium

> SonarQube -> Static analysis testing tool

       > Non -Functional automation testing tools

> Jmeter , Load runner

> postman, Rest Assured

> Test scenarios with different test data can easily be tested with automation testing tools

> Time taken to complete testing is less

> errors will be less→ as it is the tool that is doing the testing

> Reliability- the results will be consistent

> helps to do testing faster on various browsers or OS

> less expensive.

> As a tester you must know thoroughly how to use the automation tool

Automation Tool  -> Selenium

==========================================

> Used for Functional Testing

> Used to test web application on various browsers (Chrome, edge, IE, Safari, Firefox, Opera)

> You can set up selenium on Linux OS, Mac OS or windows OS

> Selenium is written in JAVA

> Every automation tool comes with 2 parts -> GUI and scripting

> In selenium we can write scripts using JAVA, python, ruby, C#, JSP

> 70 % of selenium testers will be using java with selenium

> Open source tool → free of cost

> Introduced in the year of 2004 ->Jason huggins -> Selenium version1

> Selenium version 2 -> 2008

> selenium version 3 -> 2016

> Selenium Version 4 -> 2021

Selenium is a set of tools → It has 3 main components

1. **Selenium IDE**

Download Selenium IDE on chrome browser using the below link

<https://chromewebstore.google.com/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd?pli=1>

Features:

* It is not a complete tool, it is a small extension/plugin which can be downloaded for chrome and firefox browsers
* It is a record and playbook
* It records user actions of manual execution of test and we save that test
* Using selenium IDe, we can play the test case again and again.
* This is a very simple and easy tool
* Suitable for testers who are new to testing
* gives you an idea about what is automation testing
* We may never use this tool for automation testing on major webpages
* You don't write any code or scripts in IDE

Disadvantage:

* only tests on 2 browsers, we cannot do testing on other browsers
* Test data is fixed, if new data has to be passed, you have to enter manually
* IDE does not support reading of data from excel sheets-> Data driven testing no possible
* No wait time in IDE
* Parametrization cannot be handled in IDE
* No reports are generated
* Does not support much with dynamic page testing
* No way to prioritize test cases
* No test scheduling is possible
* You cannot set up any pre-condition or post condition.

1. Selenium WebDriver → Write selenium scripts using Java

=======================================================

Features:

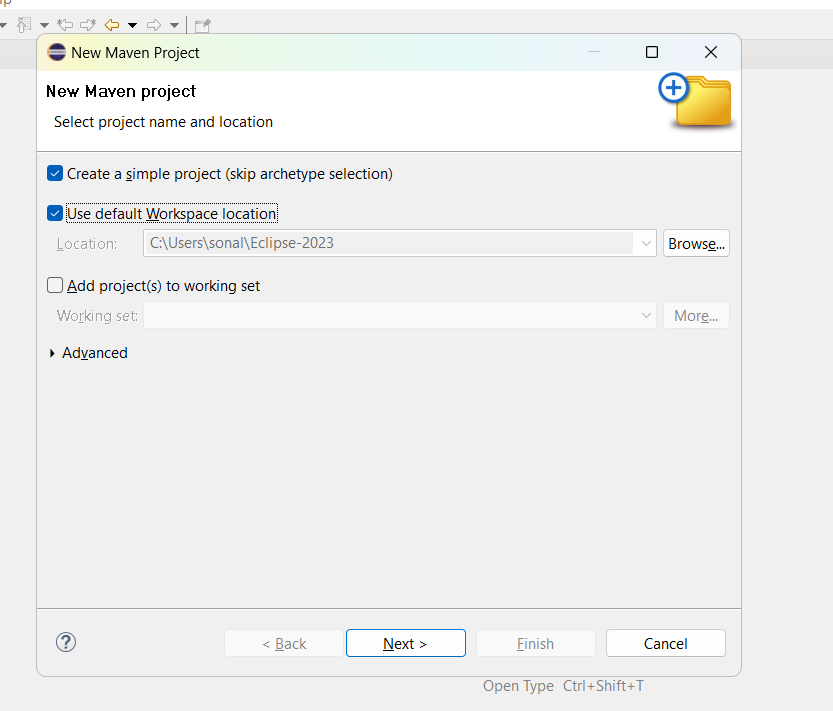
* Selenium webdriver is the main component of Selenium tools
* This tool has no front end, we will install it in our Eclipse
* It is a java library that has to be added in the eclipse java project
  + Add it manually
  + Add it automatically by using Maven project
* Using selenium webdriver, we will write selenium automation scripts
* You write the script once and same works on all browsers
* All the complex automation testing will be done using webdriver component only
* Support testing on all browsers and all OS
* Webdriver can be integrated with other frameworks like testNG, cucumber, apache poi
* Data driven testing, parametrization of scripts
* Prioritization, pre condition & post condition
* Generate test reports

Setting up of Selenium WebDriver:

==============================

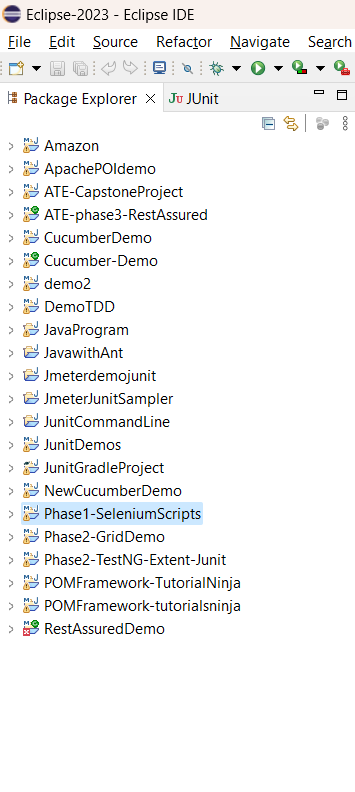
> Go to Eclipse → Go to file→ New project → Maven project → Select checkbox-  Create a simple project -> pls make sure your workspace location is your default workspace location

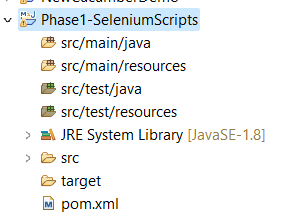
If it is not default → uncheck the second checkbox→ from the drop down select the workspace



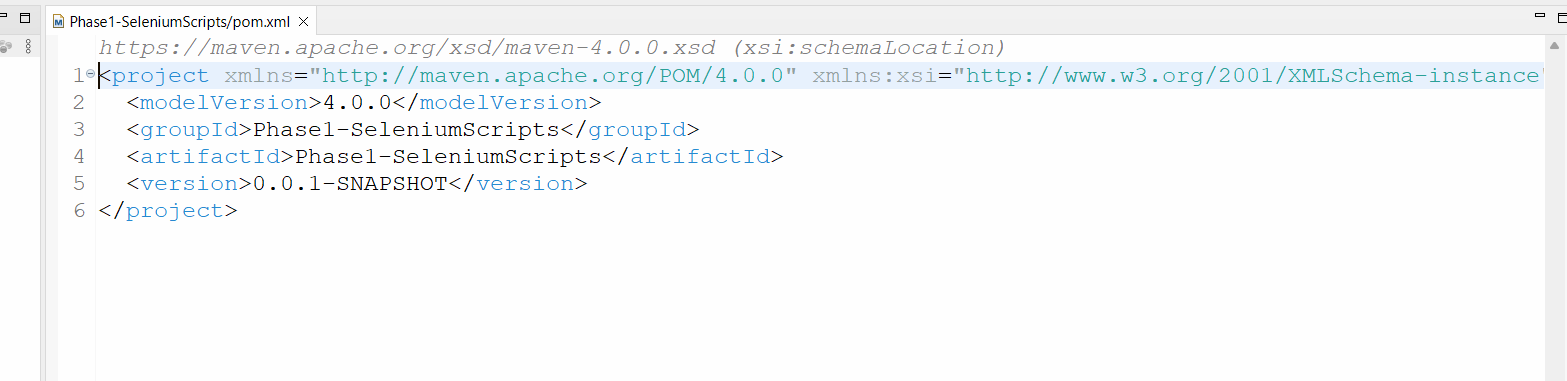
Press Next → enter group id : Phase1-SeleniumScripts → enter artifact id = Phase1-SeleniumScripts -> click on Finish button

You will see your project on the left side:





Double click on POM.xml file and open it



In the Pom file add below dependencies after line number 5 or before the tag </project>

Save the file after adding the dependencies.

**<dependencies>**

**<dependency>**

**<groupId>org.seleniumhq.selenium</groupId>**

**<artifactId>selenium-java</artifactId>**

**<version>4.11.0</version>**

**</dependency>**

**<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->**

**<dependency>**

**<groupId>io.github.bonigarcia</groupId>**

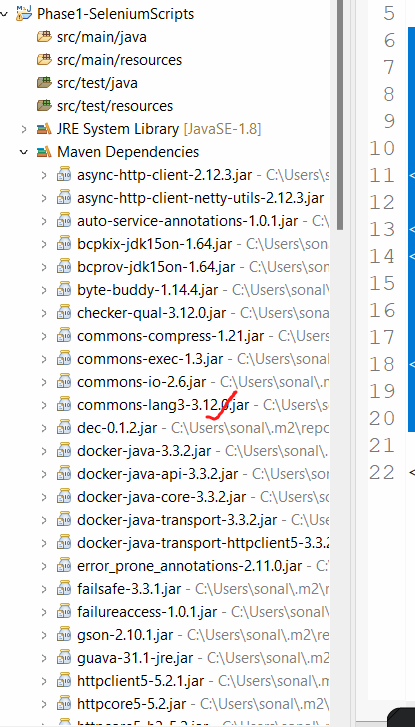
**<artifactId>webdrivermanager</artifactId>**

**<version>5.4.1</version>**

**</dependency>**

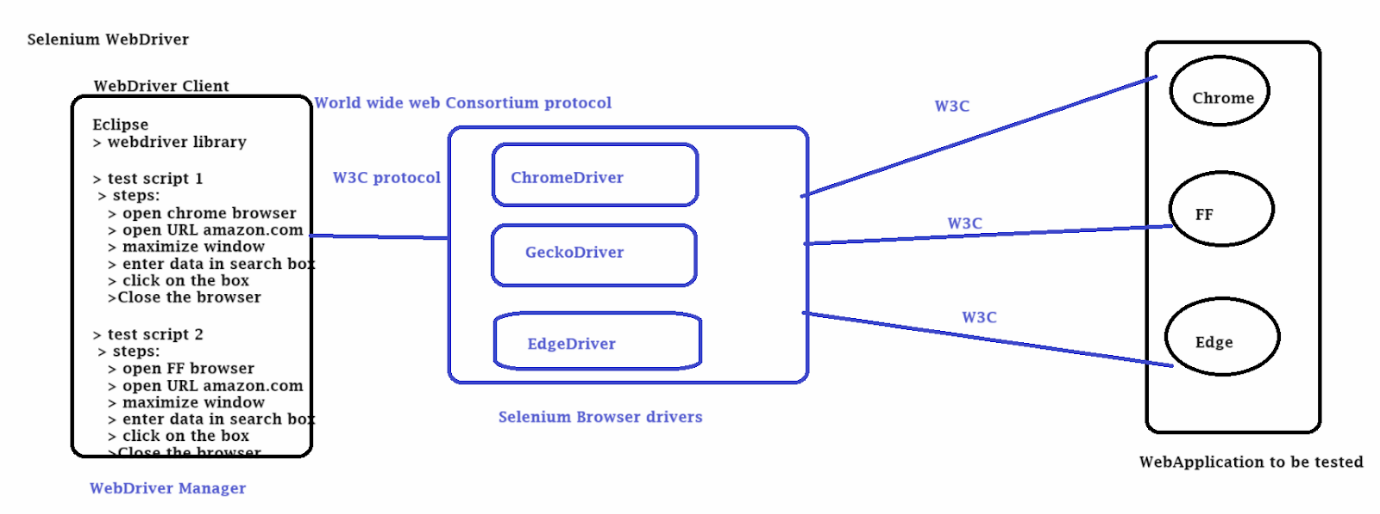
**</dependencies>**





Your setup for selenium is now complete.

Selenium Architecture:

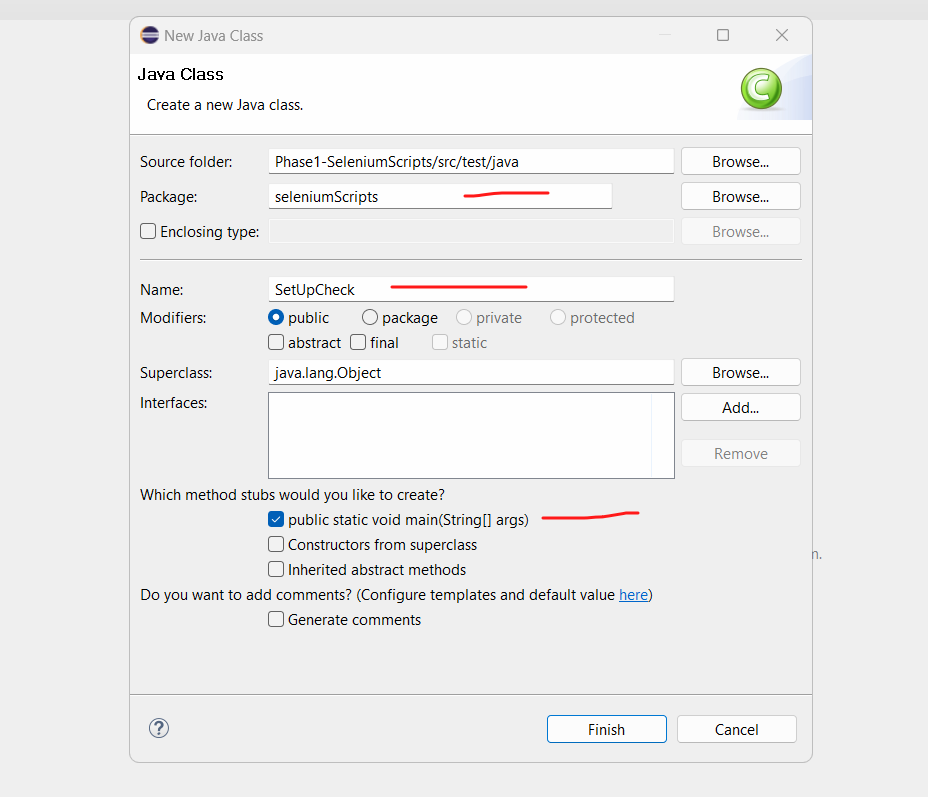


Demo 1:

======================

Let's check the setup on eclipse to check if selenium webdriver is able to create a session with your machine browser.

Create a Java class in the src/test/java folder



Demo 1:

**package** seleniumScripts;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** SetUpCheck {

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

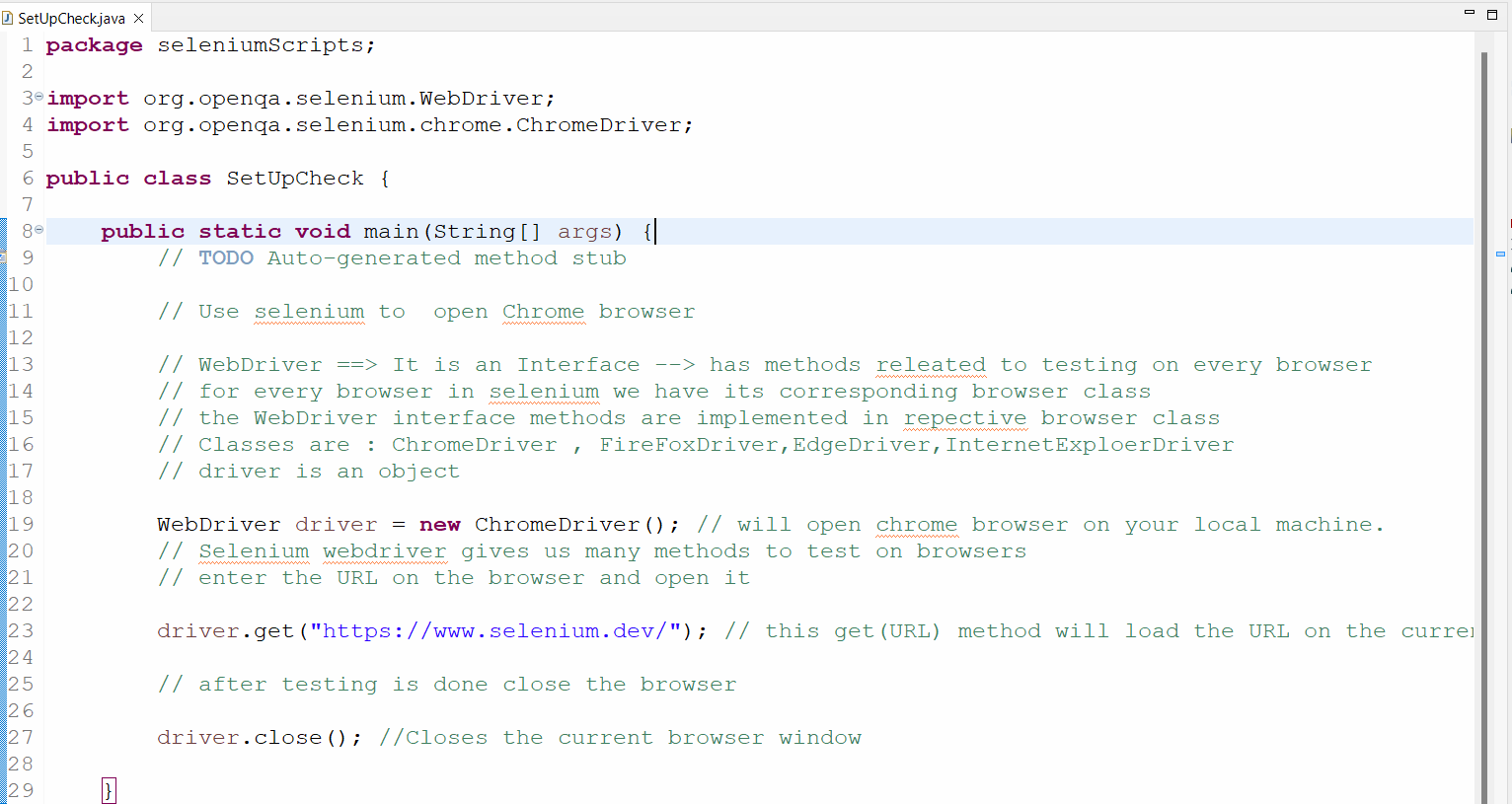
WebDriver driver = **new** ChromeDriver();

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

driver.close();

}

}



**Browser and Navigation methods Demo:**

**===============================================**

All methods information can be read at:

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/chrome/ChromeDriver.html#method-summary>

Navigation methods:

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebDriver.Navigation.html>

=====================================

**package** seleniumScripts;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** BrowserMethods {

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

// **TODO** Auto-generated method stub

// press CTL+Shift+O => to import the packages automatically

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

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

// maxamize the browser window

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

// delete browser cookies

driver.manage().deleteAllCookies();

// Fetch the title of the page

String title = driver.getTitle();

System.***out***.println("The title of page-1 is : " + title);

// Fetch the URL of the webpage

String url = driver.getCurrentUrl();

System.***out***.println("The URL of page is : " + url);

// Methods to naviagate form one webpage to another webpage

driver.navigate().to("https://www.selenium.dev/downloads/") ;

    String title1 = driver.getTitle();

System.***out***.println("The title of webpage-2 is : " + title1);

// Again naviage to  webpage 3

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

        String title2 = driver.getTitle();

System.***out***.println("The title of webpage-3 is : " + title2);

// Navigation methods : to go forward and backword on browser window

driver.navigate().back();

  String title3 = driver.getTitle();

System.***out***.println("The title of webpage is : " + title3);

driver.navigate().forward();

  String title4 = driver.getTitle();

System.***out***.println("The title of webpage is : " + title4);

driver.navigate().refresh();

driver.close();

}

}

=========================================================

**WebElements:**

====================================================

A webPage includes HTML objects which we can call as WebElements.

A web page is constructed using HTML

These HTML objects are:

==========================

> Text boxes

> Button/Submit button

> CheckBox

> Radio button

> Drop Down

> Date Picker

> Tables

> Links

> Images

As an automation tester you will have to use selenium to interact(action) with these elements

Selenium client library gives us  simple method to locate the element on the webpage and perform an action on it

**User → selenium → findElement(which Element) on webpage → perform an Action(what action)**

Now selenium gives us methods →

**findElement(address of the element) or findElements(address of the element)**

Selenium gives us some action method -**>**

**isDisplayed() -> return true if the element to be tested is visible on the webpage**

**isEnabled() -> return true if the element to be tested is enabled on the webpage**

**isSelected() -> return true if the element to be tested is selected on the webpage**

**sendKeys(“input”)  -> helps us to enter input in a text box**

**click() → helps us to click on button or link**

**getText() → helps us to get the text from webpage**

**getattribute() → helps us to get the attribute value from the webpage.**

**==================**

**Locators:**

======================

Locators will help selenium to locate an element on the web page on which we want to do testing or perform an action.

There are 8 different ways that one can use to find an element on the webpage but in your selenium script you will use any one of the ways.

There are 8 different Locators in selenium. While writing your selenium script you will use any one of the locator to find the element on the webpage.

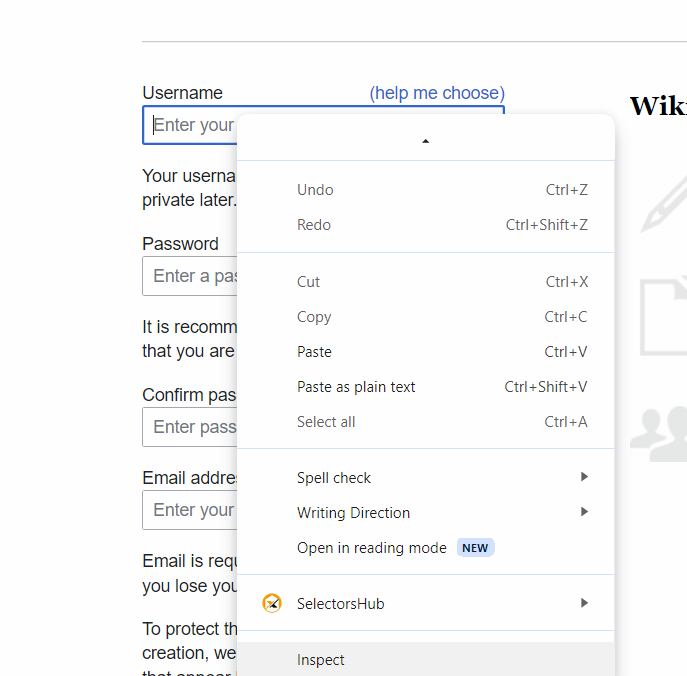
Locator = > **gives us the address of the webelement on the webPage**

Selenium Syntax:

**driver.findElement(By.locator).actionMethod();**

**How can we get the attributes of the Element on the webpage?**

**>** Bring your cursor on the object and right click → click on inspect.. Developer tool page will open at the bottom→ it will highlight in bluecolor the attributes of the element



**If we want to choose another element?**

> Click on inspector/selecto on the extreme left side and bring cursor to the new element, automatically it will highlight the attributes of new element now



Here are the attributes of an element:

In HTML everything is a tag. A tag has a name.

Every opening tag has a closing tag

Tag is represented as

<>

Tag has a name

<input    element attributes>

Closing tag

</input>

**<input id="wpName2" name="wpName" size="20" class="loginText mw-userlogin-username cdx-text-input\_\_input" placeholder="Enter your username" tabindex="1" required="" autocomplete="username">**

**Input → tag name**

**Attributes of the element:**

**id="wpName2"**

**name="wpName"**

**class="loginText mw-userlogin-username cdx-text-input\_\_input"**

**Type = “submit”**

Now we need to look at various locators:

1. ID Locator :

This locator will help us to find the element based on the id attribute value of the element.

Id locator value = id attribute value of the element

1. Name Locator:

This locator will help us to find the element based on the name attribute value of the element.

Name locator value = Name attribute value of the element

For example:

<input id="wpRetype" name="retype" size="20" class="loginPassword cdx-text-input\_\_input" placeholder="Enter password again" tabindex="3" required="" autocomplete="new-password" type="password">

**Id locator value =** wpRetype

        Name localtor value = retype

Note: The attribute value has to be unique on the webpage

==================================================

Script for ID and Name locator:

==================================================

**package** seleniumScripts;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** LocatorsIDName {

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

// **TODO** Auto-generated method stub

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

driver.get("https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Wikipedia%3ASign+up");

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

// find the element username and perform an action on it using id locator

// check if the element username is visible on the webpage

**boolean** display = driver.findElement(By.*id*("wpName2")).isDisplayed(); // returns true or false

System.***out***.println("the element is visible : " + display);

// check if the element username is enabled on the webpage

**boolean** enable = driver.findElement(By.*id*("wpName2")).isEnabled();

System.***out***.println("the element is enabled : " + enable);

// now enter the data in the text box

// sendKeys("inputdata") - method used to enter data in text box on webpage.

driver.findElement(By.*id*("wpName2")).sendKeys("admin");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Find the id locator for password field on the webpage

       display = driver.findElement(By.*id*("wpPassword2")).isDisplayed(); // returns true or false

System.***out***.println("the element is visible : " + display);

enable = driver.findElement(By.*id*("wpPassword2")).isEnabled();

System.***out***.println("the element is enabled : " + enable);

driver.findElement(By.*id*("wpPassword2")).sendKeys("admin@123");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Find the name locator for retype password field on the webpage

display = driver.findElement(By.*name*("retype")).isDisplayed();

System.***out***.println("the element is visible : " + display);

enable = driver.findElement(By.*name*("retype")).isEnabled();

System.***out***.println("the element is enabled : " + enable);

driver.findElement(By.*name*("retype")).sendKeys("admin@123");

//Find the name locator for email  field on the webpage

display = driver.findElement(By.*name*("email")).isDisplayed();

System.***out***.println("the element is visible : " + display);

enable = driver.findElement(By.*name*("email")).isEnabled();

System.***out***.println("the element is enabled : " + enable);

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

=================================================

1. ClassName Locator:

This locator will help us to find the element based on the class attribute value of the element.

1. TagName Locator :

This locator will help us to find the element based on the tag of the element.

Demo on ClassName and tagName locator:

================================

**package** seleniumScripts;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** LocatorClassTagName {

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

// **TODO** Auto-generated method stub

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

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

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

// find the element Source using tagname locator

// Note: Selenium will find the first element with the tag value as input and enter the input

driver.findElement(By.*tagName*("input")).sendKeys("Paris");

Thread.*sleep*(1500);

// find the element Destination using className locator

driver.findElement(By.*className*("sc-jlyJG kxRJeL")).sendKeys("London");

}

}

================================================

Linktext and PartialLinktext

================================================

**package** seleniumScripts;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** LocatorsLinkTextPartialLinktext {

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

// **TODO** Auto-generated method stub

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

driver.get("https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Wikipedia%3ASign+up");

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

// find the link Login  on the page and click on it

// for links, selenium has given exlusive locator called as linkText

// the value of this locator is the text on the link

driver.findElement(By.*linkText*("Log in")).click();

// wait for 1.5 seconds

Thread.*sleep*(1500);

// whenever we click on a link we will be navigated to a new page

// We want to find the title of the new page

String title = driver.getTitle();

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

driver.findElement(By.*id*("wpName1")).sendKeys("admin");

driver.findElement(By.*name*("wpPassword")).sendKeys("admin@123");

// click on the button

driver.findElement(By.*id*("wpLoginAttempt")).click();

Thread.*sleep*(1500);

// find the link and click on it

driver.findElement(By.*linkText*("Forgot your password?")).click();

title = driver.getTitle();

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

Thread.*sleep*(1500);

// this locator will search link which contains text as given in partialLinkText locator value

  driver.navigate().to("https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Wikipedia:Sign+up");

  driver.findElement(By.*partialLinkText*("logging in")).click();

  title = driver.getTitle();

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

}

}

===========================================

XPATH Locator and Axes

==============================================

XPATH => It is a locator which will find your element by traversing through the various tag in your HTML document

Whenever we dont have a unique id or name or tag or class then we use XPATH locator

OR

Always in realtime, with selenium scrips we always use xpath locator

> with locator we will construct our own locator value to find an element.

Syntax to construct our own locator value:

===================================

1. Relative Xpath syntax:

=====================================

//tagname[@attributeName=’value’]

1. Relative XPATH expression but  expression returns many values

============================================

(//tagname[@attributeName=’value’])[indexnumber]

Example:

(//input[@type='radio'])[3]

1. Relative Xpath expression using xpath methods:

* starts-with():

//tagname[starts-with[@attributeName,’startingtextvalue’]

Example:

<input type="text" onblur="fieldTrack(this);" name="namef6564a03" value="" style="width:185px;" maxlength="61">

/**/input[starts-with(@name,'name')]**

**//input[starts-with(@name-'login')]**

* **contains()**

//tagname[contains[@attributeName,’anyuniquetextvalue’]

<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="Search Amazon.in" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search Amazon.in" spellcheck="false">

//input[contains(@id,'textbox')]

//input[contains(@name,'keywords')]

//input[starts-with(@aria-label,'Search')]

* text()

//tagname[text()=’Completetextvalue’]

//option[text()='Beauty']

(//a[text()='API Docs'])[3]

DropDown demo with XPATH and Select class and its methods:

============================================

**package** seleniumScripts;

**import** java.util.List;

**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;

**public** **class** DropDownXpath {

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

// **TODO** Auto-generated method stub

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

driver.get("https://www.ironspider.ca/forms/dropdowns.htm");

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

// write xpath for the main drop down not of the dropdown option

// to handle drop down, selenium gives you a class Select class and method

// Select dd = new Select(dropdownelement)));

Select dd = **new** Select(driver.findElement(By.*xpath*("//select[@name='coffee']")));

// selectByIndex(),selectByVisibleText(),selectByValue() methods to select a dropdown option

dd.selectByIndex(1); // select with cream

Thread.*sleep*(1500);

dd.selectByValue("sugar");

Thread.*sleep*(1500);

dd.selectByVisibleText("Crisp (har har...)");

// Get all the options from the dropdown and store it in a list object. Print it on console

List<WebElement> li = dd.getOptions(); // this method returns a list of elements in the dropdown

System.***out***.println("The size of the list" + li.size());

**for**(WebElement e:li)

{

System.***out***.println(e.getText()); // get the each element in the list and print it.

}

}

}

==========================================

Xpath Axes

===========================================

Whenever the webelements tags don't have any attribute, we can still write Xpath locator for it.

You can write using xpath using xpath axes in which we will take  a tag which has attributes and then from that tag we will traverse upwards to downwards to reach our desired element Tag

<html>

<body>

   <td name=city1>

         <div>element1 </div>

          <div>element2</div>

    </td>

     <td name=country>

         <div>element3</div>

          <div>element4</div>

    </td>

</body>

<html>

Self: xpath will return the current

Syntax of Axes:

**//tagname[@attribute=’value’]/axesname::\***

**OR**

**//tagname[@attribute=’value’]/axesname::tagname[indexnumber]**

Child axes: Xpath will return all children of your current tag

You can then choose the required element tag and give the index number

//div[@class='hmmtctable']/child::ul[3]

|  |  |
| --- | --- |
| descendant | will return all the children and grandchildren of the current node |

Example:

//div[@id='div\_bseindices']/descendant::li[3]

|  |  |
| --- | --- |
| following | will return all the nodes that apppear after the current node has closed |

Example: //div[@id='div\_nseindices']/following::li[4]

==============================================

**Dynamic DropDown Demo:**

**package seleniumScripts;**

**import java.util.List;**

**import org.openqa.selenium.By;**

**import org.openqa.selenium.WebDriver;**

**import org.openqa.selenium.WebElement;**

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

**public class DynamicDropdown {**

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

**// TODO Auto-generated method stub**

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

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

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

**driver.findElement(By.*xpath*("//div[@class='DPXIy']/child::textarea[1]")).sendKeys("Simplilearn");**

**//Wait for few seconds for the dropdown to be visible**

**Thread.*sleep*(7000);**

**// Write an XPATH where selenium will fetch all the options in the drop down**

**// Create a list object that will store the fetched options from the dropdown**

**List<WebElement> li = driver.findElements(By.*xpath*("(//div[@role='presentation'])[2]/descendant::li"));**

**// print the size of the list**

**System.*out*.println(li.size());**

**// iterate over the list of webelement, print them and click on the element that your desire**

**for(WebElement text: li)**

**{**

**String elementText = text.getText();**

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

**if(elementText.contains("customer"))**

**{**

**text.click();**

**break;**

**}**

**}**

**===============================**

**Calender Demo 1:**

**package seleniumScripts;**

**import org.openqa.selenium.By;**

**import org.openqa.selenium.WebDriver;**

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

**public class CalanderDemo1 {**

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

**// TODO Auto-generated method stub**

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

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

**driver.get("https://seleniumpractise.blogspot.com/2016/08/how-to-handle-calendar-in-selenium.html");**

**// Click on the text box field , so that the calender is displayed.**

**driver.findElement(By.*xpath*("//div[@class='date-posts']/descendant::input")).click();**

**Thread.*sleep*(1500);**

**// On the calander lets inpsect the month and year tab and print the month and year text**

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

**System.*out*.println(monthyear);  // January 2024**

**// Split the String monthyear in to 2 new strings**

**// Create a new array object that will store the string monath and year**

**String my[] = monthyear.split(" "); // split based on the space**

**String month = my[0]; //January**

**String year = my[1]; //2024**

**// write a loop**

**while(!(month.equals("August") && year.equals("2025")))**

**{**

**//click on the next button in the calender**

**driver.findElement(By.*xpath*("//span[text()='Next']")).click();**

**// come out of the loop**

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

**month = monthyear.split(" ")[0];**

**year = monthyear.split(" ")[1];**

**}**

**// select the date : //a[text()='8']**

**//table[@class='ui-datepicker-calendar']/descendant::a[8]**

**driver.findElement(By.*xpath*("//table[@class='ui-datepicker-calendar']/descendant::a[8]")).click();**

**}**

**}**

**Calendar Demo with JavaScript Executor**

**package seleniumScripts;**

**import org.openqa.selenium.JavascriptExecutor;**

**import org.openqa.selenium.WebDriver;**

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

**public class CalanderDemoJavaScriptExecutor {**

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

**// TODO Auto-generated method stub**

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

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

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

**Thread.*sleep*(1000);**

**JavascriptExecutor js = ((JavascriptExecutor)driver);**

**js.executeScript("document.getElementById('src').value='Paris Beauvais Airport,Paris,France'");**

**js.executeScript("document.getElementById('onward\_cal').value='02-Feb-2024'");**

**}**

**}**

**MouseHover on an Element**

**package seleniumScripts;**

**import org.openqa.selenium.By;**

**import org.openqa.selenium.WebDriver;**

**import org.openqa.selenium.WebElement;**

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

**import org.openqa.selenium.interactions.Actions;**

**public class MouseHoveAmazon {**

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

**// TODO Auto-generated method stub**

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

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

**driver.manage().deleteAllCookies();**

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

**Thread.*sleep*(1500);**

**// store the element location in a WebElement object**

**WebElement e1= driver.findElement(By.*xpath*("//span[@class='nav-line-2 ']"));**

**// MouseHover activites are done using methods moveToElement(e1) ==> class Action**

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

**a.moveToElement(e1).perform();**

**Thread.*sleep*(1500);**

**WebElement e2 = driver.findElement(By.*xpath*("//div[@id='nav-xshop-container']/descendant::a[6]"));**

**e2.click(); // click on mobiles link**

**Thread.*sleep*(3000);**

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

**// Store the location of element mobiles and accessories.**

**WebElement e3 = driver.findElement(By.*xpath*("//div[@id='nav-progressive-subnav']/descendant::a[2]"));**

**a.moveToElement(e3).perform();**

**}**

**}**

==========================

Frames and MouseActions

**package** seleniumScripts;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.interactions.Actions;

**public** **class** MouseActionsFramesDemo {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

driver.switchTo().frame(0);

WebElement e1 = driver.findElement(By.*id*("draggable"));

WebElement e2 = driver.findElement(By.*id*("droppable"));

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

a.clickAndHold(e1).moveToElement(e2).release().perform();

}

}

==========================

Alert Box

=======================

Alert box is an action which occurs on the webpage when wrong input or no input is entered by the user.

By default driver object cannot work on the alert.

Just like frame we have to switch the driver to the alert

In the alert we cannot inspect any element

So in order to perform action on the alert box, we have to use 4 methods

 > accept() -> to click on yes or OK

> dismiss() -> to click on no or cancel

> sendKeys() -> to enter data in alert box

> getText() -> to get the message on the alert box

These methods are part of Alert class.

So we will create object of action class and then call these methods

**package** seleniumScripts;

**import** org.openqa.selenium.Alert;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** AlertBoxDemo {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

driver.get("https://mail.rediff.com/cgi-bin/login.cgi");

// just click on Sign in button without enter username and password. An alert box will be displayed

driver.findElement(By.*xpath*("//input[@type='submit']")).click();

// to perform actions on alertbox, switch to it

Thread.*sleep*(2000);

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

// get the text from alert

String text = a.getText();

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

// press ok button

a.accept();

}

}

========================================

Accept Cookies:

**package** seleniumScripts;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**public** **class** AlertCookies {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

Thread.*sleep*(1000);

WebElement e1 = driver.findElement(By.*xpath*("//div[@class='cookie-consent\_\_basic cookie-consent\_\_inner']/descendant::span[3]"));

e1.click();

}

}

=============================

Authentication pop ups

===========================

**package** seleniumScripts;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** AuthenticationPopUps {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

       // Whenever there is a URL that open with the authencation pop up then we have to pass the username

       // and password with the URL itself

       // Synatx:

       // driver.get("https://username:password@URL");

driver.get("https://admin:admin@the-internet.herokuapp.com/basic\_auth");

}

}

================================

**package** seleniumScripts;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** DialogueBox {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

Thread.*sleep*(2000);

driver.navigate().refresh();

// find the element text box to enter mobile number on the dialogue box

driver.findElement(By.*xpath*("//div[@class='\_3skCyB']/descendant::input[1]")).sendKeys("2343535345");

// find the element locator for cross symbol to lose the dialogue box

driver.findElement(By.*xpath*("//div[@class='JFPqaw']/child::span")).click();

}

}

==============================

Handling tabs and windows:

===============================

**package** seleniumScripts;

**import** java.awt.AWTException;

**import** java.awt.Robot;

**import** java.awt.event.KeyEvent;

**import** java.util.ArrayList;

**import** org.openqa.selenium.WebDriver;

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** TabsDemo {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

   /\* We will now learn a class called as ROBOT class that allows selenium to perform keyword Event

    \*/

Robot r = **new** Robot();

r.keyPress(KeyEvent.***VK\_CONTROL***); // press control key

r.keyPress(KeyEvent.***VK\_T***);

// Now relase the keys

r.keyRelease(KeyEvent.***VK\_CONTROL***);

r.keyRelease(KeyEvent.***VK\_T***);

/\*\*\*\*\*\*\*\*\*AGAIN OPEN a Tab\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

r.keyPress(KeyEvent.***VK\_CONTROL***); // press control key

r.keyPress(KeyEvent.***VK\_T***);

// Now release the keys

r.keyRelease(KeyEvent.***VK\_CONTROL***);

r.keyRelease(KeyEvent.***VK\_T***);

/\*\*\*\*\*\*\*\*\*\*\*\*\*MANAGE and SWITCH TO TABS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// first compute how many tabs are open in that browser window

ArrayList<String> tabs = **new** ArrayList<String>(driver.getWindowHandles());

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

// method to switch to a new Tab and open a new URL

driver.switchTo().window(tabs.get(1));

driver.get("https://lms.simplilearn.com/dashboard");

Thread.*sleep*(1000);

driver.switchTo().window(tabs.get(2));

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

// Use close() method to close only the current tab/window which is open

// use quit() method to close all the tabs and browser window of the session

//driver.close();

driver.quit();

}

}

=====================================

Take Screenshots Demo

=================================

In the maven project → go to POM.xml file —> add the below dependency

<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->

<dependency>

    <groupId>commons-io</groupId>

    <artifactId>commons-io</artifactId>

    <version>2.11.0</version>

</dependency>

Save the file



======================

**package** seleniumScripts;

**import** java.io.File;

**import** java.io.IOException;

**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.chrome.ChromeDriver;

**public** **class** ScreesnshotDemo1 {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

// take screeshot of current window and save it in a file

// Use class TakesScreenshot and method -> getScreenshoAs

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

// Screesnhot is saved in the object srcFile

// In the current project --> create a folder Screenshot--> create a file with name opera1.png

File destFile = **new** File("./Screenshot/opera1.png");

FileUtils.*copyFile*(srcFile, destFile);

/\*Copies a file to a new location preserving the file date.

This method copies the contents of the specified source file

to the specified destination file. The directoryholding the destination file

is created if it does not exist. If the destination file exists,

then this methodwill overwrite it \*/

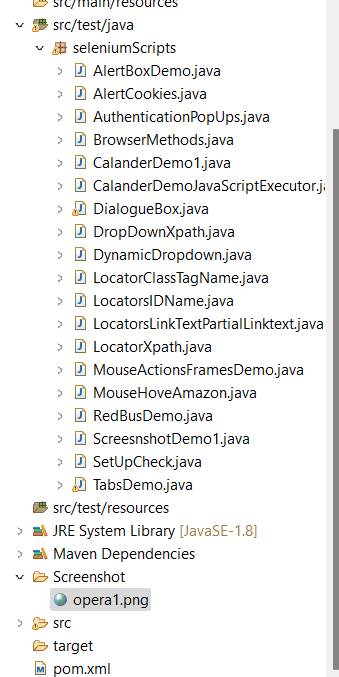
}

}

After execution of the script→ screenshot will be available in the project directory

To view that → right click on project→ click on refresh

Then in the project we will see screenshot folder.



==================================

Screenshot with border - using JavaScriptExecutor

**package** seleniumScripts;

**import** java.io.File;

**import** java.io.IOException;

**import** org.apache.commons.io.FileUtils;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**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;

**public** **class** ScreesnshotJavaScriptExecution {

**public** **static** WebDriver *driver*;

**public** **static** **void** drawborder(WebElement element,WebDriver driver)

{

JavascriptExecutor js = ((JavascriptExecutor)driver);

js.executeScript("arguments[0].style.border='3px solid red'", element);

}

**public** **static** **void** takescreenshot\_method(String filepath) **throws** IOException

{

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

File destFile = **new** File(filepath);

FileUtils.*copyFile*(srcFile, destFile);

}

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

// **TODO** Auto-generated method stub

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

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

*driver*.manage().deleteAllCookies();

*driver*.get("https://www.opera.com/download");

Thread.*sleep*(1000);

WebElement e1 = *driver*.findElement(By.*xpath*("//div[@class='cookie-consent\_\_basic cookie-consent\_\_inner']/descendant::span[3]"));

*drawborder*(e1,*driver*);

*takescreenshot\_method*("./Screenshot/opera2.png");

}

}

================

Implicit Wait

Specifies the amount of time the driver should wait when searching for an element if it is not immediately present.

When searching for a single element, the driver should poll the page until the element has been found, or this timeout expires before throwing a NoSuchElementException. When searching for multiple elements, the driver should poll the page until at least one element has been found or this timeout has expired.

Increasing the implicit wait timeout should be used judiciously as it will have an adverse effect on test run time, especially when used with slower location strategies like XPath.

**package** seleniumScripts;

**import** java.time.Duration;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**public** **class** ImplicitWait {

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

// **TODO** Auto-generated method stub

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

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

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

// Wait for given seconds for the webpage URL to Load,

//after the time is up, if the page is not open, then give an error

driver.manage().timeouts().pageLoadTimeout(Duration.*ofSeconds*(2));

//Implicit wait

// Selenium will poll webpage for all elements visibiltiy

// and keep wiaitng for 3 seconds

// if any one element is also visible--> it will proceed with exection

// if even after 3 seconds if no element is  visible, it will give element not found exception

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(3));

WebElement e1 = driver.findElement(By.*xpath*("//div[@class='cookie-consent\_\_basic cookie-consent\_\_inner']/descendant::span[3]"));

e1.click();

}

}

====================================

Explicit wait

=============================

**package** seleniumScripts;

**import** java.time.Duration;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

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

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** ExplicitWait {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

driver.navigate().refresh();

// we want the script to wait(10 seconds) until the mobile number text box element is visible

// In explicit wait , tester will give 2 things:

  // waititme - 10 seconds --> selenium is going to wait for 10 seconds

  // condition --> while waiting it will keep checking th econdition

// If 5 seconds have been completed and the condition is satisfied now

// selenium will immediatly comeout of the wait time and conitnue executing the script

// In this selenium doesnot have to wait for the entire time,

// as the conditon is satisfied comeout of waiting time

// If waittime is complete but condition is not satisfied --> error on console

// explicit wait is written for single element

//we want webdriver to wait for 10 seconds

WebDriverWait wait = **new** WebDriverWait(driver,Duration.*ofSeconds*(10));

// wait until the given condition is satisfied

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//div[@class='\_3skCyB']/descendant::input[1]")));

WebElement e1 = driver.findElement(By.*xpath*("//div[@class='\_3skCyB']/descendant::input[1]"));

// find the element text box to enter mobile number on the dialogue box

e1.sendKeys("2343535345");

}

}

=========================================

Fluent Wait

 FluentWait<WebDriver> wait = new FluentWait<WebDriver>(driver)

    .withTimeout(Duration.ofSeconds(10))

          .pollingEvery(Duration.ofSeconds(5)).ignoring(NoSuchElementException.class);

   WebElement element = wait.until(new Function<WebDriver, WebElement>()

     {

   public WebElement apply(WebDriver driver) {

   WebElement element = driver.findElement(By.xpath(""));

   return element;

   }

     });

=====================================

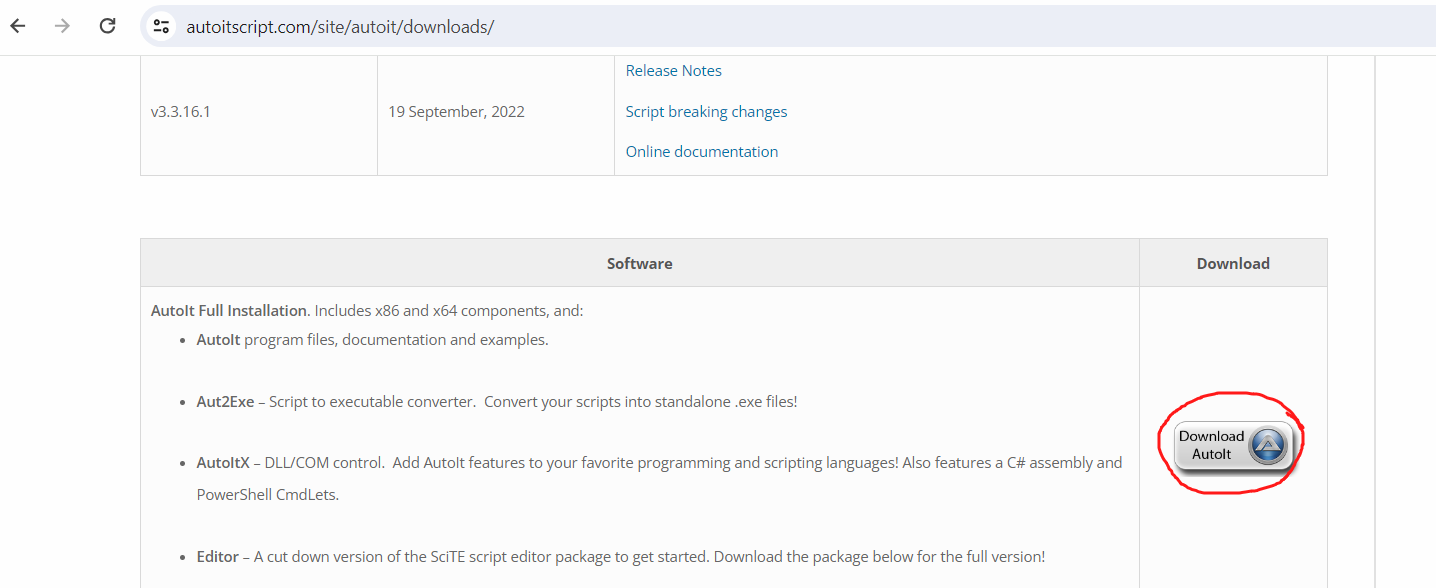
AutoIt tool

=====================================

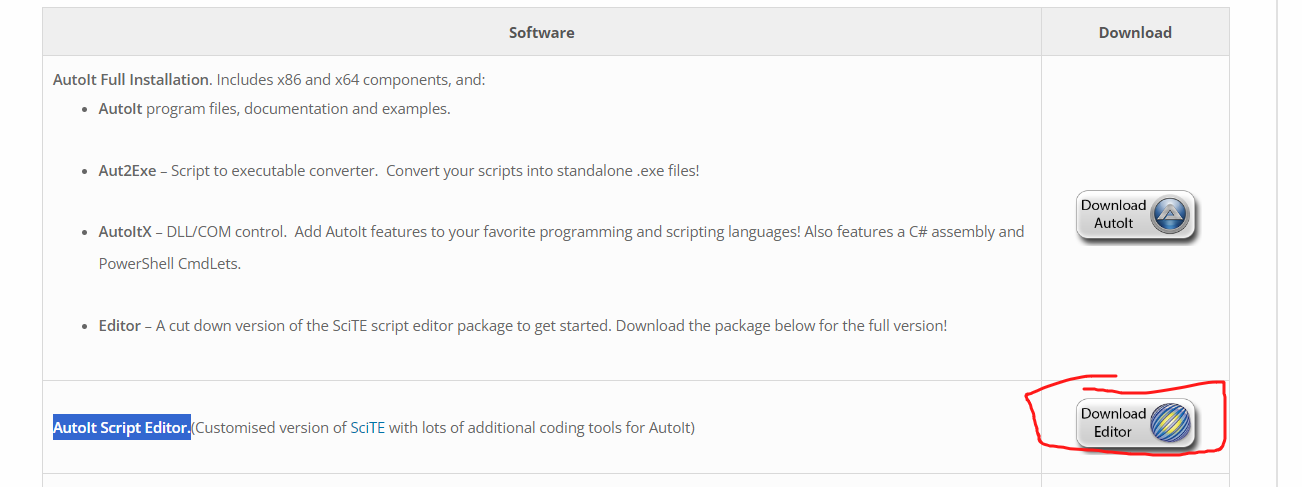
1. Download AutoIT

<https://www.autoitscript.com/site/autoit/downloads/>

Scroll down and click on button as show below



Also download AutoIt Script Editor.



Step 3: Decide the file to be uploaded

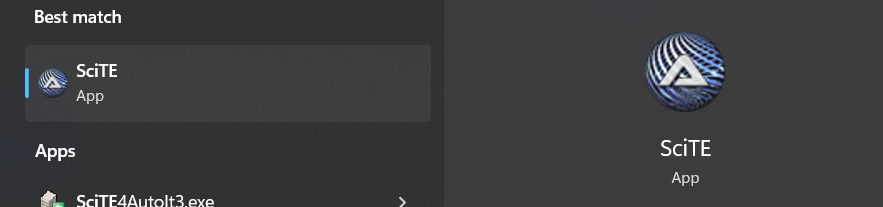
=============

For me it is :  C:\Users\sonal\Documents\ATE-Phase1-SL\JDBCconnectivity.png

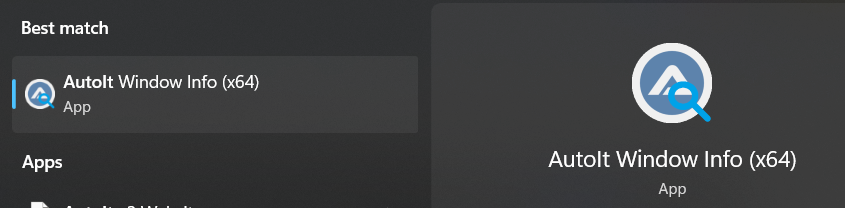
Step 4:

Now we will use AutoIT tool to write script that will first locate the file box and the open button

On the windows machine → open the autoIT editor -> search for app ->Scite



Now open another auto It tool to inspect the file box

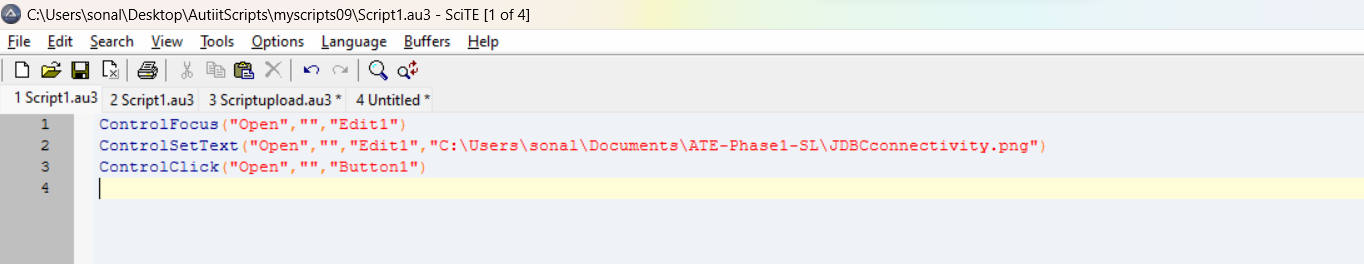


Write the script on the editor

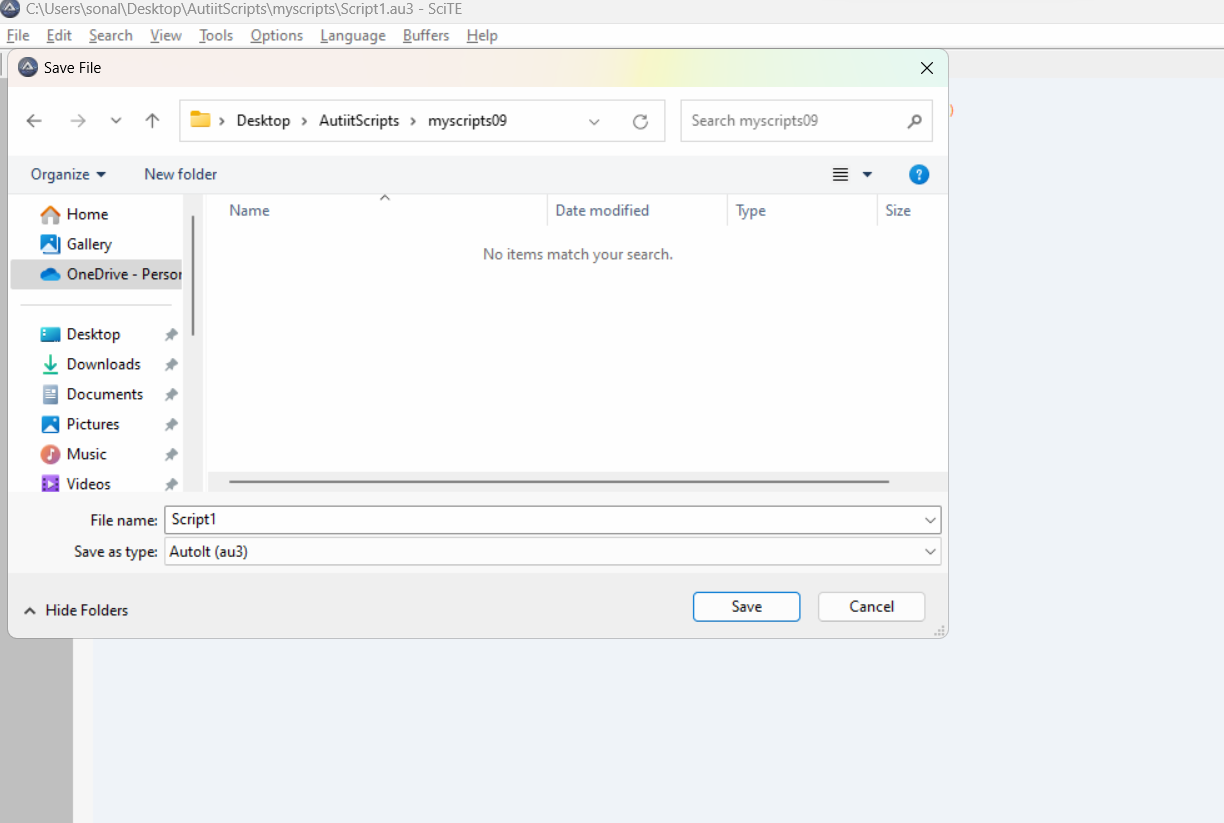
ControlFocus("Open","","Edit1")

ControlSetText("Open","","C:\Users\sonal\Documents\ATE-Phase1-SL\JDBCconnectivity.png")

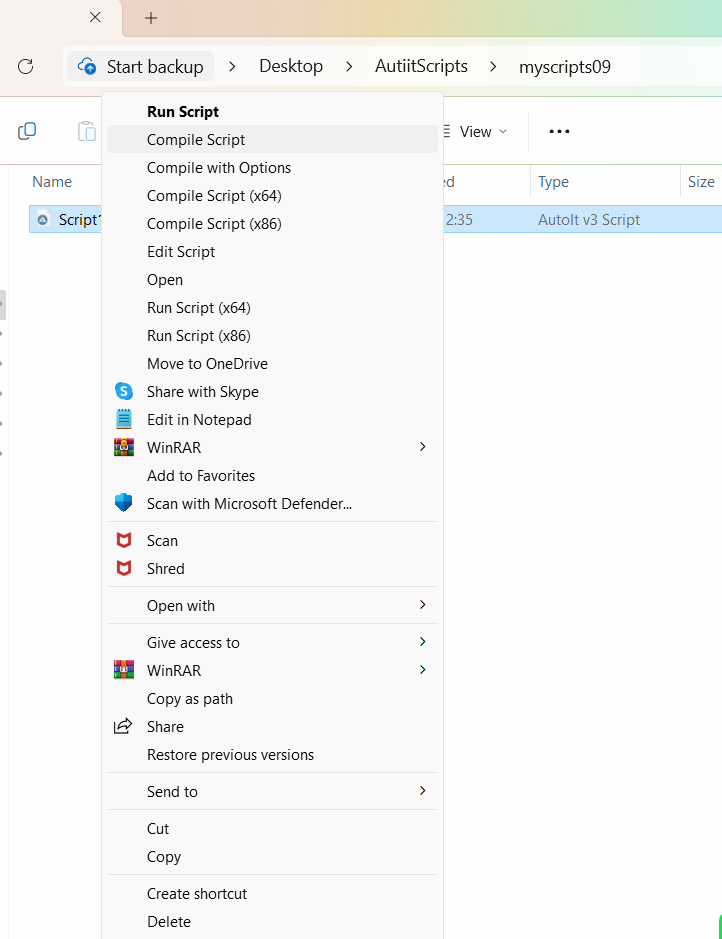
ControlClick("Open","","Button1")



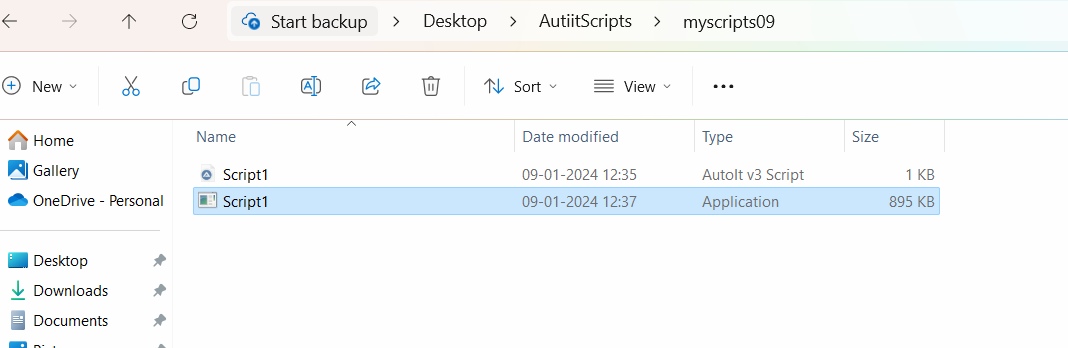
Save the script



Open the folder where the script is → right click on the script —> show more options and compile it



The compiled script will be in the folder:



**package** seleniumScripts;

**import** java.io.IOException;

**import** java.time.Duration;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

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

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** AutoItdemo {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

driver.get("https://www.remove.bg/");

WebDriverWait wait = **new** WebDriverWait(driver,Duration.*ofSeconds*(10));

// wait until the given condition is satisfied

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//div[@class='mx-auto w-full px-8 max-w-5xl relative']/descendant::button[1]")));

WebElement e1 = driver.findElement(By.*xpath*("//div[@class='mx-auto w-full px-8 max-w-5xl relative']/descendant::button[1]"));

e1.click();

// selenium to run the autoID compiled script

Runtime.*getRuntime*().exec("C:\\Users\\sonal\\Desktop\\AutiitScripts\\myscripts09\\Script1.exe");

}

}

=================================

CSS selector:

==================================

package seleniumScripts;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class CSSSeelctorDemo {

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

// TODO Auto-generated method stub

WebDriver driver = new ChromeDriver();

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

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

// CSS locator : tagname#idAttributeValue

driver.findElement(By.cssSelector("input#email")).sendKeys("admin");

// CSS locator : tagname[nameAttribute='value']

driver.findElement(By.cssSelector("input[name='pass']")).sendKeys("admin@123");

driver.navigate().to("https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Wikipedia%3ASign+up");

Thread.sleep(1500);

        driver.findElement(By.linkText("Log in")).click();

        // Fetch the inner text of the element

        // #userloginForm > form > div > div.mw-htmlform-field-HTMLTextField.loginText.mw-userlogin-username.mw-ui-vform-field > label

      String innertext =  driver.findElement(By.cssSelector("#userloginForm > form > div > div.mw-htmlform-field-HTMLTextField.loginText.mw-userlogin-username.mw-ui-vform-field > label")).getText();

        System.out.println(innertext);

}

}

Multiple Window Handles:

**package** seleniumScripts;

**import** java.util.Iterator;

**import** java.util.Set;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** MultipleWindows {

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

// **TODO** Auto-generated method stub

WebDriver driver;

driver = **new** ChromeDriver();

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

    driver.get("https://omayo.blogspot.com/2013/05/page-one.html");

/\*

\* String parent\_win = driver.getWindowHandle();

\*

\* System.out.println("Main window page id is: " + parent\_win);

\*/

    // click on the link to open a new window

    driver.findElement(By.*cssSelector*("#HTML37 > div.widget-content > p > a")).click();

    // we capture id of each window or Tab that selenium opens for this driver session.

    //  Set<String> org.openqa.selenium.WebDriver.getWindowHandles()

    Set <String> s1 = driver.getWindowHandles(); // A set of window handles which can be used to iterate over all open windows.

    // We can iterate over each value in the set and print the id of the windows

    Iterator<String> itr = s1.iterator();

    // fetch parent window id

    String parent\_win = itr.next();

    System.***out***.println("Main window page id is: " + parent\_win);

    String child\_window = itr.next();

    System.***out***.println("Child window page id is: " + child\_window);

    // switch to the child window

  driver.switchTo().window(child\_window) ;

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

  String text = driver.findElement(By.*cssSelector*("body > div > h3")).getText();

    System.***out***.println(text);

    driver.quit();

}

}

==================================

LessonEnd Project

=======================

1. Class 1- Write the script for AutoIT scenario that we have above
2. Class2- Write the Selenium JDBC script

**package** seleniumScripts;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** org.openqa.selenium.WebDriver;

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

**public** **class** SeleniumHJDBC {

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException, SQLException {

// **TODO** Auto-generated method stub

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

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

// go to webpage, fetch its URL and title and store in a DB

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

String URL = driver.getCurrentUrl();

String title = driver.getTitle();

                    // send the URL and title to the DB

String dburl = "jdbc:mysql://localhost:3306/seleniumtest";

String username = "root";

String password = "root";

       Class.*forName*("com.mysql.cj.jdbc.Driver");

       Connection con = DriverManager.*getConnection*(dburl, username, password);

       PreparedStatement ps = con.prepareStatement("insert into webtest values(?,?)");

       ps.setString(1, title);

       ps.setString(2, URL);

       ps.executeUpdate();

driver.navigate().to("https://www.selenium.dev/downloads/");

URL = driver.getCurrentUrl();

title = driver.getTitle();

  ps = con.prepareStatement("insert into webtest values(?,?)");

       ps.setString(1, title);

       ps.setString(2, URL);

       ps.executeUpdate();

       con.close();

}

}

Class 3: Write demo for screenshot as show in above examples in notes.

=====================================================

Phase End Project

=====================================

**package** phaseEndProject;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**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;

**public** **class** TestSearchPage {

**public** **static** WebDriver *driver*;

**public** **static** **void** takescreenshot\_method(String filepath) **throws** IOException

{

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

File destFile = **new** File(filepath);

FileUtils.*copyFile*(srcFile, destFile);

}

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

// **TODO** Auto-generated method stub

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

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

*driver*.manage().deleteAllCookies();

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

Thread.*sleep*(1500);

// Create Page object repository--list of object that has to tested on this page

WebElement searchbox = *driver*.findElement(By.*id*("twotabsearchtextbox"));

WebElement searchbutton = *driver*.findElement(By.*id*("nav-search-submit-button"));

// Action Methods

searchbox.sendKeys("iphone 13");

searchbutton.submit();

*takescreenshot\_method*("./Screenshot/amazonsearch1.png");

Thread.*sleep*(3000);

WebElement mobilelink = *driver*.findElement(By.*xpath*("(//div[@class='a-section'])[2]/descendant::span[11]"));

mobilelink.click();

*takescreenshot\_method*("./Screenshot/amazonsearch2.png");

// new tab will open. handle the windows

ArrayList<String> tabs = **new** ArrayList<String>(*driver*.getWindowHandles());

*driver*.switchTo().window(tabs.get(1));

Thread.*sleep*(1500);

WebElement wishlist = *driver*.findElement(By.*id*("wishListMainButton"));

wishlist.click();

*driver*.quit();

}

}

================

**package** phaseEndProject;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**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.openqa.selenium.interactions.Actions;

**public** **class** TestMobilePage {

**public** **static** WebDriver *driver*;

**public** **static** **void** takescreenshot\_method(String filepath) **throws** IOException

{

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

File destFile = **new** File(filepath);

FileUtils.*copyFile*(srcFile, destFile);

}

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

// **TODO** Auto-generated method stub

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

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

*driver*.manage().deleteAllCookies();

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

Thread.*sleep*(2000);

WebElement mobilelink = *driver*.findElement(By.*linkText*("Mobiles"));

  String text = mobilelink.getText();

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

mobilelink.click();

WebElement mobile = *driver*.findElement(By.*xpath*("//div[@id='nav-progressive-subnav']/descendant::span[2]"));

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

a.moveToElement(mobile).perform();

Thread.*sleep*(2000);

*takescreenshot\_method*("./Screenshot/mobile1.png");

WebElement apple\_link = *driver*.findElement(By.*linkText*("Apple"));

        text = apple\_link.getText();

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

apple\_link.click();

Thread.*sleep*(2000);

WebElement link2 = *driver*.findElement(By.*xpath*("(//div[@class='sg-col-inner'])[6]/descendant::span[9]"));

text = link2.getText();

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

link2.click();

*takescreenshot\_method*("./Screenshot/mobile2.png");

ArrayList<String> tabs = **new** ArrayList<String>(*driver*.getWindowHandles());

*driver*.switchTo().window(tabs.get(1));

Thread.*sleep*(1500);

WebElement addtoCart = *driver*.findElement(By.*id*("add-to-cart-button"));

addtoCart.click();

*takescreenshot\_method*("./Screenshot/mobile3.png");

*driver*.quit();

}

}

================

package in.amazon.project.lesson1;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;

public class TestSignInPage {

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

// TODO Auto-generated method stub

WebDriver driver = new ChromeDriver();

driver.manage().deleteAllCookies();

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

// wait for 2 seconds the page to Load

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(5));

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

// locating the dropdown that has sign in button -> signin - accounts and lists

        WebElement e1 = driver.findElement(By.xpath("//\*[@class='nav-line-2 ']"));

Actions a = new Actions(driver);

a.moveToElement(e1).perform();

Thread.sleep(3000);

driver.findElement(By.linkText("Sign in")).click();

// Inspect the Email and continue button on sign page

Thread.sleep(3000);

driver.findElement(By.id("ap\_email")).sendKeys("mittal.sonal04@gmail.com");

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

// inspect password input box and click on sign button

driver.findElement(By.id("ap\_password")).sendKeys("password@123");

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

// fetch the error text

String text = driver.findElement(By.xpath("//div[@class='a-box-inner a-alert-container']/descendant::span")).getText();

System.out.println("Error test is >> " + text);

driver.close();

}

}

===================

**package** phaseEndProject;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.interactions.Actions;

**public** **class** SeleniumJDBCTrasaction {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

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

String title = driver.getTitle();

Thread.*sleep*(2000);

WebElement mobilelink = driver.findElement(By.*linkText*("Mobiles"));

mobilelink.click();

WebElement mobile = driver.findElement(By.*xpath*("//div[@id='nav-progressive-subnav']/descendant::span[2]"));

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

a.moveToElement(mobile).perform();

Thread.*sleep*(2000);

WebElement apple\_link = driver.findElement(By.*linkText*("Apple"));

        String text = apple\_link.getText();

apple\_link.click();

String dburl = "jdbc:mysql://localhost:3306/seleniumtest";

String username = "root";

String password = "root";

       Class.*forName*("com.mysql.cj.jdbc.Driver");

       Connection con = DriverManager.*getConnection*(dburl, username, password);

       PreparedStatement ps = con.prepareStatement("insert into endproject values(?,?,?)");

       ps.setString(1, title);

       ps.setString(2, text);

       ps.setString(3, "Yes");

       ps.executeUpdate();

}

}

==============================

Selenium Integrationw ith SikuliX

========================================

This will not work with Maven project

You have to use a Java project

In this project add the selenium jar files

you cannot execute it on you lab machine

It has to be done on windows laptop

We will go to this page https://launchpad.net/sikuli/+download

And download the jar file -->  sikulixide-2.0.5.jar (md5)

Add this jar file to Java project --> go to build path--> select class path --> add the jar --> apply and close

Sikuli interacts with the windows application via images

Which ever element we have to test on desktop application, we have to take an image of it-> using snipping tool

and feed this image to sikuli tool methods

What ever is on the image it will perform that action on webpage

So change the resultion of your desktop

go to desktop--> right click-->display settings --> resolution --> 1280 \* 720

After this .. take snapshots of the application

Save the snapshots at a folder

And then pass it to selenium script

Make sure the file to be uploaded should also have resolution same as 1280 \* 720

**package** demo;

**import** java.time.Duration;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

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

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.sikuli.script.FindFailed;

**import** org.sikuli.script.Pattern;

**import** org.sikuli.script.Screen;

**public** **class** SkuiliUploadFile {

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

// **TODO** Auto-generated method stub

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

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

       driver.manage().deleteAllCookies();

driver.get("https://www.remove.bg/");

Thread.*sleep*(4000);

WebDriverWait wait = **new** WebDriverWait(driver,Duration.*ofSeconds*(10));

// wait until the given condition is satisfied

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//div[@class='mx-auto w-full px-8 max-w-5xl relative']/descendant::button[1]")));

WebElement e1 = driver.findElement(By.*xpath*("//div[@class='mx-auto w-full px-8 max-w-5xl relative']/descendant::button[1]"));

e1.click();

Screen s = **new** Screen();

Pattern textfield = **new** Pattern("C:\\sikuli\\image10.png");

Pattern openButton = **new** Pattern("C:\\sikuli\\image101.png");

s.wait(textfield,20);

s.type(textfield,"C:\\Users\\sonal\\Documents\\ATE-Phase1-SL\\sikulidemo");

s.click(openButton);

}

}

Change back the display settings.

====================================================================

Browser Profiling

● First , close the Firefox if it is open.

● Open Run (Windows+R) and type firefox.exe -p and click OK.

● A dialogue box will open named “Firefox -choose user profile.”

● Select the option “Create Profile” from the window, and a Wizard will open. Click on Next.

● Provide your profile name which you want to create, and click on the Finish button.