SELENIUM CONCEPTS

**1. What are limitations in selenium webdriver**

* It does not support and non web-based applications, it only supports web based applications.
* Its an open source tool so in case of any technical issues you need to rely on the selenium community forums to get your issue resolved.
* You need to know at least one of the supported language very well in order to automate your application successfully.
* No inbuilt reporting capability so you need plugins like JUnit and TestNG for generating test reports.
* Captcha and Bar code cannot be tested using selenium.

**2**. **Installing/configure selenium**

* At first, go to <http://www.seleniumhq.org> and then click on downloads -> download java version (3.3.0 latest ) under  **Selenium Client & WebDriver Language Bindings** and extract the files.
* Adding jar files to Selenium:

Go to eclipse and select project then right click on it and click on Properties and click on Java Build Path - > In that click on Libraries - > click on Add External JARs -> Then go to the specific location where the selenium jars have downloaded - > select “**client-combined-3.3.0-nodeps”** and open it -> Now, click on Apply and then ok -> In the same way import the remaining jar files which are in lib folder. Finally, It got installed into eclipse.

**3. What are different ways of locating elements in selenium**

Attribute based locators - They rely on attributes of the page

1. ID
2. Name
3. ClassName
4. Tag Name
5. Link Text and Partial LinkText

Structure based locators - they rely on structure of the page for finding the elements

1. CSS Selector  
    - Tag and ID  
    - Tag and class  
    - Tag and attribute  
    - Tag, class, and attribute  
    - Inner text
2. DOM (Document Object Model)  
    - getElementById  
    - getElementsByName  
    - dom:name - dom: index
3. XPath

**Code:**

**package** PracticalExamples;

**import** java.util.concurrent.TimeUnit;

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

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

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

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

**public** **class** ElementLocators {

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

// System.setProperty("webdriver.chrome.driver",

// "C:\\Users\\sowmy\\chromedriver.exe");

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

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

// WebDriver driver = new ChromeDriver();

driver.get("https://www.idealist.org/en/");

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

driver.findElement(By.*linkText*("Sign Up")).click();

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

driver.findElement(By.*id*("form-input-first\_name")).sendKeys("Sowmya");

driver.findElement(By.*name*("last\_name")).sendKeys("Reddy");

driver.findElement(By.*xpath*(".//input[@type='email']")).sendKeys("sowmya1408@gmail.com");

driver.findElement(By.*id*("form-input-password")).sendKeys("Saibersys1408");

driver.findElement(By.*className*("btn btn-block btn-primary qa-signup-button")).click();

}

}

**4. Which is fastest way to identify elements in web page?**

The safest and fastest way to identify elements in web page is through **“id”** because it is unique and after that select **css** and then **xpath**.

**5. What is absolute path and relative path in xpath?**

**Xpath:**

* Xpath is a language for locating nodes In XML documents.
* One of the main reason for using this is when we don’t have proper id or name attribute for the element you want to locate.

**Absolute path:**

* Absolute XPath starts with the root node or a forward slash (/).
* The advantage of using absolute is, it identifies the element very fast.
* Disadvantage here is, if anything goes wrong or some other tag added in between, then this path will no longer works.

**Relative path:**

* A relative xpath is one where the path starts from the node of your choice - it doesn't need to start from the root node. It starts with Double forward slash(//).
* Advantage of using relative xpath is, you don't need to mention the long xpath, you can start from the middle or in between.
* Disadvantage here is, it will take more time in identifying the element as we specify the partial path not (exact path). If there are multiple elements for the same path, it will select the first element that is identified.
* It is mainly good for table cells and we can use **“.”** before the code to make it relative xpath.

**Code:**

package practicalExamples;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class AbsoluteRelativeXpath {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

//Absolute Xpath - starting from the root node

driver.findElement(By.xpath("html/body/div[1]/div[1]/table/tbody/tr/td[2]/div/div/span[2]")).click();

//Relative Xpath- starting from the node of the element

driver.findElement(By.xpath("//input[@id='signin\_email']")).sendKeys("sowmya123");

driver.findElement(By.xpath(".//input[@type='password']")).sendKeys("s12345");

}

}  
**6. Write code on how to use xpath functions**

**Code:**

**package** PracticalExamples;

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

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

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

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

**public** **class** XpathFunctions {

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

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

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

driver.get("https://www.facebook.com/");// pass the url to the driver

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

System.***out***.println(driver.getTitle()); // to know the title of the page

System.***out***.println(driver.getPageSource()); // to know the page source

System.***out***.println(driver.getCurrentUrl()); // to verify with the url

driver.findElement(By.*id*("u\_0\_1")).sendKeys("Sruthi");

driver.findElement(By.*id*("u\_0\_3")).sendKeys("Reddy");

driver.findElement(By.*id*("u\_0\_6")).sendKeys("sowmya2160@gmail.com");

driver.findElement(By.*id*("u\_0\_9")).sendKeys("sowmya2160@gmail.com");

driver.findElement(By.*id*("u\_0\_d")).sendKeys("Wrongpassword@14");

Select sel1 = **new** Select(driver.findElement(By.*xpath*(".//\*[@id='month']")));

sel1.selectByVisibleText("Feb");

Select sel2 = **new** Select(driver.findElement(By.*xpath*(".//\*[@id='day']")));

sel2.selectByValue("28");

Select sel3 = **new** Select(driver.findElement(By.*xpath*(".//\*[@id='year']")));

sel3.selectByIndex(24);

driver.findElement(By.*id*("u\_0\_g")).click();

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

Select drp = **new** Select(driver.findElement(By.*xpath*(".//\*[@id='userNavigationLabel']]")));

drp.selectByVisibleText("Log Out");

driver.close();

}

}

**7. Different types of waits or synchronization in selenium webdriver**  
 **Implicit wait:**

Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.

* It is effectively used when you know how much time it will take to identify the element for example: 10seconds or 20 seconds

**Code:**

**package** practicalExamples;

**import** java.util.concurrent.TimeUnit;

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

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

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

**public** **class** Waits {

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

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

System.*setProperty*("webdriver.chrome.driver", "Drivers/chromedriver.exe");

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

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

driver.manage().timeouts().implicitlyWait(5000, TimeUnit.***SECONDS***);

driver.findElement(By.*id*("nav-link-accountList")).click();

}

}

**Explicit wait:**

Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, explicit waits are applied for a particular instance only.

* When we don’t know the exact time it will take to identify the element. At that time we can give a condition like until the element is visible then u click on that.
* Here 2 things are important

1. time elapsed – WebDriverWait wait = **new** WebDriverWait(driver,5);
2. expected condition –

WebElement element = wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*name*("email")));

* It is best suited for **AJAX** based application because sometimes all the **DOM** elements will be present but they are not visible/clickable/enable.
* Waits in Selenium = synchronization in QTP/UFT

**Code:**

**public** **class** Waits {

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

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

System.*setProperty*("webdriver.chrome.driver", "Drivers/chromedriver.exe");

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

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

driver.manage().timeouts().implicitlyWait(5000, TimeUnit.***SECONDS***);

driver.findElement(By.*id*("nav-link-accountList")).click();

WebDriverWait wait = **new** WebDriverWait(driver,5);

WebElement element = wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*name*("email")));

element.sendKeys("sowmyareddy");

}

}

**Fluent wait:**

Each FluentWait instance defines the

a) maximum amount of time to wait for a condition

b) Frequency with which to check the condition (it will act like a loop)

c) Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as NoSuchElementExceptions when searching for an element on the page.

**Important:** In below code first it will wait for 20 seconds, then after 5 seconds the loop will run again and also it will eliminate certain conditions also.

**8. How to handle multiple windows in selenium webdriver**

**Code:**

package practicalExamples;

import java.util.Iterator;

import java.util.Set;

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 MultipleWindowsAmazon {

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

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

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

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

Thread.sleep(5000);

driver.findElement(By.id("nav-link-accountList")).click();

driver.findElement(By.id("ap\_email")).sendKeys("sowmya3452@gmail.com");

driver.findElement(By.name("password")).sendKeys("Sowmya@3452");

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

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

Actions action = new Actions(driver);

action.moveToElement(logout).build().perform();

driver.findElement(By.id("nav-item-signout")).click();

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

driver.findElement(By.xpath("//\*[@id='a-page']/div[1]/div[5]/div[2]/a[1]")).click();

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

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

Iterator<String> it = allWindows.iterator();

String a = it.next();

driver.switchTo().window(a);

System.out.println(a);

String child = it.next();

driver.switchTo().window(child);

System.out.println(child);

System.out.println(allWindows.size());

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

}

}

**9. How to launch web page using chrome driver**

**Code:**

**package** PracticalExamples;

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

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

**public** **class** ChromePage {

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

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

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

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

}

}

**10. Write code to verify any application login page is working or not**

**package** PracticalExamples;

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

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

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

**public** **class** LoginPage {

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

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

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

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

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

driver.navigate().back();

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

driver.findElement(By.*xpath*("//\*[@name='email']")).sendKeys("sowmya2160@gmail.com");

driver.findElement(By.*xpath*(".//input[@type='password']")).sendKeys("Wrongpassword@14");

driver.findElement(By.*xpath*(".//input[@value='Log In']")).click();

driver.findElement(By.*cssSelector*("#userNavigationLabel")).click();

driver.close();

}

}

**11. How to select items from dropdown/select box**

**Code:**

**package** PracticalExamples;

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

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

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

**public** **class** DropDownAndSelect {

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

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

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

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

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

driver.findElement(By.*xpath*(".//input[@id='trip-type-round-trip']")).click();

driver.findElement(By.*xpath*(".//input[@id='air-city-departure']")).sendKeys("MCI");

driver.findElement(By.*xpath*(".//input[@id='air-city-arrival']")).sendKeys("dallas");

driver.findElement(By.*id*("air-date-departure")).click();

driver.findElement(By.*xpath*(".//\*[@data-date='4/4/2017']")).click();

driver.findElement(By.*cssSelector*("#air-date-return")).click();

driver.findElement(By.*cssSelector*("#calendar-may-9")).click();

driver.findElement(By.*id*("price-type-points")).click();

driver.findElement(By.*name*("adultPassengerCount")).click();

**int** i=0;

**while**(i<4){

driver.findElement(By.*xpath*(".//\*[@id='jb-number-selector-more']")).click();

i++;

}

driver.findElement(By.*cssSelector*("#jb-booking-form-submit-button")).click();

}

}

**12. Difference between assert and verify?**

Assert:

When an “**assert**” command fails, the test execution will be aborted. So when the Assertion fails, all the test steps after that line of code are skipped. The solution to overcoming this issue is to use a try-catch block. We use the Assertion in the try catch block. Mostly, the assert command is used when the end result of the check value should pass to continue to the next step. Simply, if the assert condition is true then the program control will execute the next test step but if the condition is false, the execution will stop and further test step will not be executed.

Verify:

When a “**verify**” command fails, the test will continue executing and logging the failure. Mostly, the Verify command is used to check non-critical things. In such cases where we move forward even though the end result of the check value is failed. Simply, there won’t be any halt in the test execution even though the verify condition is true or false.

**Code:**

**package** practicalExamples;

**import** java.util.Iterator;

**import** java.util.Set;

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

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

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

**import** org.testng.Assert;

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

**public** **class** AssertAndVerify {

@Test

**public** **static** **void** Assert() {

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

System.*setProperty*("webdriver.chrome.driver", "Drivers/chromedriver.exe");

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

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

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

String title = driver.getTitle();

Assert.*assertEquals*("Best Buy: Expert Service. Unbeatable Price.", title); // it will compare actual with expected value

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

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

Iterator<String> it = wh.iterator();

**while**(it.hasNext()){

String noOfHandles = it.next().toString();

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

}

}

}

**13. Difference between driver.close and driver.quit methods?**

These two methods are for closing the browser session in Selenium WebDriver

**driver.close:**

This command closes the Browser window which is in focus. If there are more than one Browser window opened by the Selenium Automation, then the close( ) command will only close the browser window which is having focus at that time. It won’t close the remaining Browser windows.

**driver.quit:**  
This command is generally used to shut down the WebDrivers instance. Hence it closes all the Browser windows that are opened by the Selenium Automation.

Note- They work similar way when Selenium Automation opens only single Browser window but they differ in their functionality when there are more than one Browser windows opened by the Selenium Automation.

**14. Write code for right click in selenium**

Code:

**package** PracticalExamples;

**import** java.util.concurrent.TimeUnit;

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

**import** org.openqa.selenium.Keys;

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

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

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

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

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

**public** **class** RightClick {

**public** **static** **void** main(String[] args) { System.*setProperty*("webdriver.chrome.driver","C:\\sowmya\\chromedriver.exe");

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

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

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

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

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

WebElement element = driver.findElement(By.*xpath*("//ul[@class='Pos(r) Miw(1000px) Pstart(9px) Lh(1.7) Reader-open\_Op(0) mini-header\_Op(0)']/li[2]"));

action.contextClick(element).sendKeys(Keys.***ARROW\_DOWN***).sendKeys(Keys.***ENTER***);

action.perform();

}

}

**how to use autoit to implement downlad/uplaod files**

* While automating web-application many times you will get window based activity like- file upload, file download pop-up, window authentication for secure sites etc.
* In this case, Selenium fails and will not be able to handle desktop elements to avoid this we will use **AutoIT** script that will handle desktop or windows elements and will combine AutoIT script with our Selenium code.

**AutoIT:**

* AutoIt v3 is a freeware BASIC-like scripting language designed for automating the ***Windows GUI*** and general scripting. It uses a combination of keystrokes, mouse movement and window/control manipulation in order to automate tasks in a way not possible or reliable with other languages (e.g. VBScript and SendKeys).
* AutoIt is also very small, self-contained and will run on all versions of Windows out-of-the-box with no annoying “runtimes” required!

**How to write a script in AutoIT?**

For AutoIt scripting, you should have three things ready.

1-AutoIt Editor- Editor helps us to write AutoIt scripts.

2-Tool Finder (Same as firebug on Firefox) – It will help us to identify the element and check their Attributes.

3- AutoIt Help section- This help you to understand about AutoIt functions and what are the parameter it accepts.

* Now, we have to use some methods in order to do upload the file. It requires the use of 3 methods like.

**ControlFocus- This will give focus on the window by giving control**

**Syntax: ControlFocus(“title”, “text” ,”controlID”) controlID = class + instance**

**ControlSetText- This will set the file path – by giving the path which you want to upload**

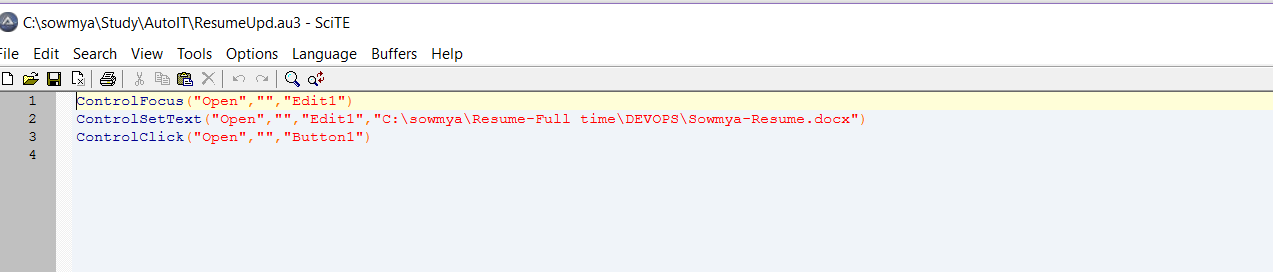
**Syntax: ControlSetText( “title”,”text”,”controlID”,”path”)**

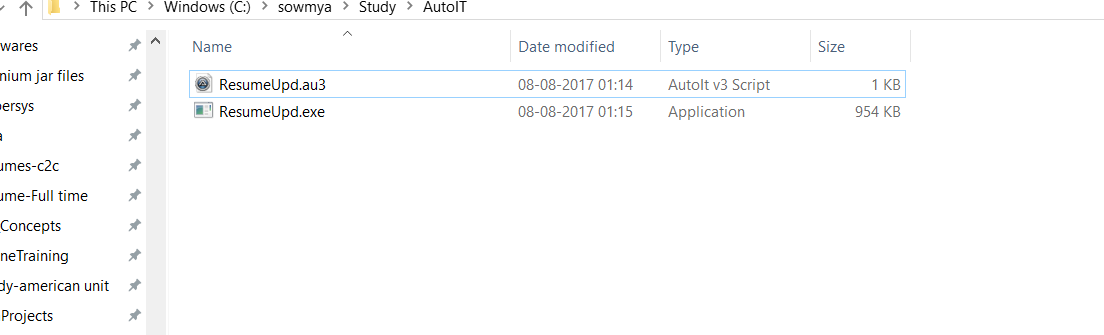
**ControlClick-This will click on button – helps in uploading your file**

**Syntax: ControlClick(“title”,”text”,”buttonId”)**

**Implementation:**

* Here we have to download AutoIt editor and AutoIt
* **AutoIt editor** – write the script
* After installing both of them we have to write the script and for that goto C:\Program Files (x86)\AutoIt3\SciTE and double click SciTE.exe
* Just like firebug and firepath, here we have a finder to locate the elements which is a finder, for that goto ->C:\Program Files (x86)\AutoIt3 and double click Au3Info\_x64.exe
* Now open script editor and write 3 methods and save it in a location -> right click on the au3 file and click on compile script(x64) -> an exe file will be generated
* Give the path of exe file in the method getRunTime.exe()





**Code:**

**package** practicalExamples;

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

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

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

**public** **class** AutoIT {

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

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

System.*setProperty*("webdriver.chrome.driver", "Drivers/chromedriver.exe");

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

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

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

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

driver.findElement(By.*id*("signin\_email")).sendKeys("sowmya3452@gmail.com");

driver.findElement(By.*id*("signin\_password")).sendKeys("Sowmya@3452");

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

driver.findElement(By.*xpath*(".//\*[@id='resPromoDisplay']/a[2]/b")).click();

driver.findElement(By.*xpath*(".//\*[@id='container']/div/div[2]/div[2]/div/div[2]/form[1]/label/span")).click();

Thread.*sleep*(5000);

Runtime.*getRuntime*().exec("C:\\sowmya\\Study\\AutoIT\\ResumeUpd.exe");

}

}

**15. How to launch different browsers (IE, safari, firefox)**

**Internet Explorer:**

At first, we need to set the system property by giving key and value.

key is **“webdriver.ie.driver”** and value is an executable file that we have to download and paste the path of it. Below is the code

**System.setProperty(“webdriver.ie.driver”,”exe path”);**

Then, for invoking the browser we have to create an object for the driver class and make class object reference to WebDriver interface (which has methods but doesn’t implement itso it make use of this driver class). So, it is given below

WebDriver d = new InternetExplorerDriver();

**Google Chrome:**

At first, we need to set the system property by giving key and value.

key is **“webdriver.chrome.driver”** and value is an executable file that we have to download and paste the path of it. Below is the code

**System.setProperty(“webdriver.chrome.driver”,”exe path”);**

Then, for invoking the browser we have to create an object for the driver class and make class object reference to WebDriver interface (which has methods but doesn’t implement itso it make use of this driver class). So, it is given below

WebDriver d = new ChromeDriver();

**Firefox:**

At first, we need to set the system property by giving key and value.

key is **“webdriver.gecko.driver”** and value is an executable file that we have to download and paste the path of it. Below is the code

**System.setProperty(“webdriver.gecko.driver”,”exe path”);**

Then, for invoking the browser we have to create an object for the driver class and make class object reference to WebDriver interface (which has methods but doesn’t implement itso it make use of this driver class). So, it is given below

WebDriver d = new FirefoxDriver();

**Safari:**

First, we need to download an extension file - “**safaridriver.safariextz”** and install it which acts like a proxy between safari browser and test cases. Then for invoking the browser we have to create an object for the driver class and make class object reference to WebDriver interface (which has methods but doesn’t implement itso it make use of this driver class). So, it like

WebDriver d = new SafariDriver();

**16) MouseHover:**

package ActionsClass;

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 MouseHover {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

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

driver.findElement(By.id("nav-link-accountList")).click();

driver.findElement(By.id("ap\_email")).sendKeys("sowmya3452@gmail.com");

driver.findElement(By.name("password")).sendKeys("Sowmya@3452");

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

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

Actions action = new Actions(driver);

action.moveToElement(logout).build().perform();

driver.findElement(By.id("nav-item-signout")).click();

}

}

**17) Facebook Account Creation:**

package practicalExamples;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

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

public class AccountCreation {

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

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/");// pass the url to the driver

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

System.out.println(driver.getTitle()); // to know the title of the page

System.out.println(driver.getPageSource()); // to know the page source

System.out.println(driver.getCurrentUrl()); // to verify with the url

// which we have mentioned

// in line 15

// time between one step to another

// driver.findElement(By.linkText("Forgot account?")).click();

Thread.sleep(5000);

driver.findElement(By.id("u\_0\_2")).sendKeys("Sruthi");

driver.findElement(By.id("u\_0\_7")).sendKeys("sowmya2160@gmail.com");

driver.findElement(By.id("u\_0\_a")).sendKeys("sowmya2160@gmail.com");

driver.findElement(By.id("u\_0\_e")).sendKeys("Wrongpassword@14");

driver.findElement(By.xpath(".//\*[@id='u\_0\_4']")).sendKeys("Reddy");

Select sel1 = new Select(driver.findElement(By.xpath(".//\*[@id='month']")));

sel1.selectByVisibleText("Feb");

Select sel2 = new Select(driver.findElement(By.xpath(".//\*[@id='day']")));

sel2.selectByValue("28");

Select sel3 = new Select(driver.findElement(By.xpath(".//\*[@id='year']")));

sel3.selectByIndex(24);

driver.findElement(By.id("u\_0\_h")).click();

driver.findElement(By.name("websubmit")).click();

Thread.sleep(5000);

Select drp = new Select(driver.findElement(By.xpath(".//\*[@id='userNavigationLabel']]")));

drp.selectByVisibleText("Log Out");

driver.close();

}

}

**18) JavascriptExecutor:**

package practicalExamples;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class HandlingTextBoxes {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

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

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("document.getElementById('email').value='sowmya7817@gmail.com'");

}

}

**18) WebTables:**

package practicalExamples;

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 WebTables {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://webdatacommons.org/webtables/");

WebElement element = driver.findElement(By.xpath(".//\*[@id='toccontent']/table[1]/tbody/tr[3]/td[2]"));

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

List<WebElement> element2 = driver.findElements(By.xpath(".//\*[@id='toccontent']/table[1]/tbody/tr"));

System.out.println(element2.size());

}

}

**19) Screenshots**

1. **UIElements**

package Screenshots;

import java.io.IOException;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class UIElements {

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

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "Drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

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

driver.findElement(By.id("trip-type-round-trip")).click();

UtilityLibrary.CaptureScreenshot(driver, "first"); // it will take

// screenshot and

// give the name as

// first

System.out.println("first");

driver.findElement(By.name("originAirport")).sendKeys("MCO");

UtilityLibrary.CaptureScreenshot(driver, "Second"); // it will take

// screenshot and

// give the name as

// second

System.out.println("second");

driver.findElement(By.xpath(".//input[@name='destinationAirport']")).sendKeys("LGA");

UtilityLibrary.CaptureScreenshot(driver, "Third");// it will take

// screenshot and

// give the name as

// third

System.out.println("third");

driver.quit();

}

}

**b) UtilityLibrary:**

**package** Screenshots;

**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;

**public** **class** UtilityLibrary {

**public** **static** **void** CaptureScreenshot(WebDriver driver,String ScreenshotName) { //here we are parameterizing

**try**{

TakesScreenshot ts = (TakesScreenshot)driver; //here TakesScreenshot is an interface so we have to typecast to driver. It will take screenshot

File src = ts.getScreenshotAs(OutputType.***FILE***); //here it will store a memory in the form of File

FileUtils.copyFile(src, **new** File("Screenshots/"+ScreenshotName+".png")); // copy files from source to destination

}

**catch**(Exception mg){

System.***out***.println("Problem in taking "+ mg );

}

}

}