Date:17/08/2020

**Practical No 1**

**Aim :-** Install Selenium IDE; Write a test suite containing minimum 4 test cases for different formats.

**Installation of Selenium IDE in Chromium Browser**

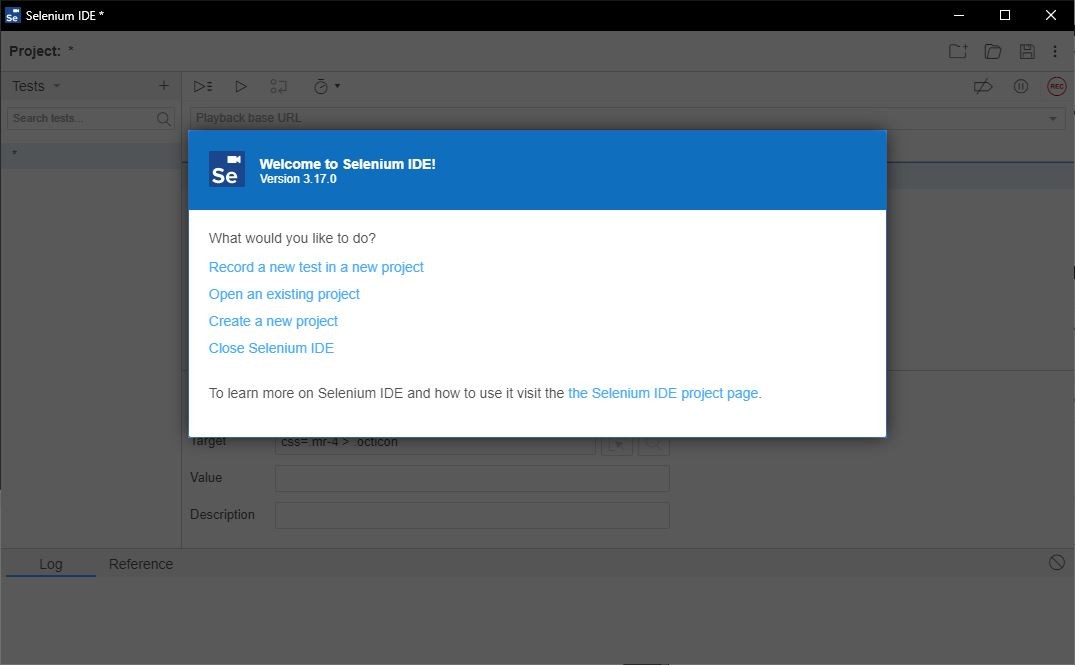
**Step-1 :-** Launch Chromium browser.

**Step-2 :-** Search for Selenium IDE in the ”chrome web store”.

**Step-3 :-** Select ”Add to Chrome” Button which will lead to chromium pop-up asking for permission to add the extension, click ”Add Extension” button.

**Step-4 :-** Chromium will automatically download and install the extension. Which can be accessed from the ”Extensions” Icon Button on the Navigation Bar.

**Step-5 :-** Clicking on the selenium IDE icon, the extension will open a new window with the selenium IDE.



# Creating/Recording and Running a Test Suite

**Step-1 :-** Launch the Chromium browser and open the selenium IDE extension.

**Step-2 :-** When the Welcome screen of selenium IDE is visible, select the option ”create a new project”.

**Step-3 :-** Give an appropriate name to the project and rename the untitled default test case to another appropriate title.

**Step-4 :-** Set the ”Playback base URL” to the URL value of the website/webpage you wish to test.

**Step-5 :-** At this point you have two choices where in you can either manually type the test steps in the IDE or let selenium record the steps for you by clicking the ”Record button ON”.

**Step-6 :-** We will first Record the steps using selenium’s Record functionality. On click- ing the ”Record button ON” or keyboard shortcut ”Ctrl-U”.

**Step-7 :-** A new window will pop up with the base URL provided by us earlier , this window is special because selenium will record our mouse clicks and key inputs. *Be careful not to reveal any sensitive information in the record mode.*

**Step-8 :-** Now perform the testing on various elements of the HTML document either by direct clicks/key inputs or by selecting various methods from the ”right-click selenium drop-down menu” which will provide various options for assert various elements and validate various values.

*→*

*→*

**Step-9 :-** You can try out filling forms, dummy login credential test and a lot more.

**Step-10 :-** After you are done with the test script click on ”Stop Recording” button or keyboard shortcut ”Ctrl-U”.

**Step-11 :-** Save the test script by clicking on ”Save project icon” or keyboard shortcut ”Ctrl-S”.

**Step-12 :-** Next save the script in the desired folder path and selenium will download the ”.side” file to the same .

**Step-13 :-** You can reopen the saved Project by clicking ”Open Project icon” or key- board Shortcut ”Ctrl-O” and selecting the ”.side” file from the desired location.

**Step-14 :-** To run the test click on ”Playback icon” button and selenium will run the test as per the script and log all the events in the window. You can verify which steps succeed and which ones fail with Error messages as well.

**Step-15 :-** The test script can be written manually as well and will produce the same results.

**Selenium IDE Features**

1. Menu Bar Menu bar is positioned at the top most portion of the Selenium IDE

interface. It allows the user to change name/open project/save project and more.

Figure

2: Menu Bar 2. ToolBar The toolbar offers options such as play test/record test

and play all the tests in test suite. More advanced features such as

breakpoints/debugger are also available.

3. Test case panel This panel allows the user to add/delete /modify the selenium

commands. The Command, Target, and Value entry fields display the currently

selected command along with its parameters. These are entry fields where you

can modify the currently selected command. The first parameter specified for a

command in the Reference tab of the bottom pane always goes in the Target

field. If a second parameter is specified by the Reference tab, it always goes in the

Value field. The Comment field allows you to specify in a human readable format

a description of the current command.

4. Navigation Panel Navigation between test cases and test suites is done

through the right hand side of Selenium IDE. Clicking on Tests with the small caret

will open up a menu. When saving the project will be saved as the new .side

format, which will include all test cases and suites combined. It is refered to as a

project.

5. Console Panel The bottom pane, called Console Panel for it’s similarity with

web devtools, is used for different utility functions: Log, Reference, depending on

which tab is selected.

**Test Case 1 :**

**Test URL :** https://login.mailchimp.com/

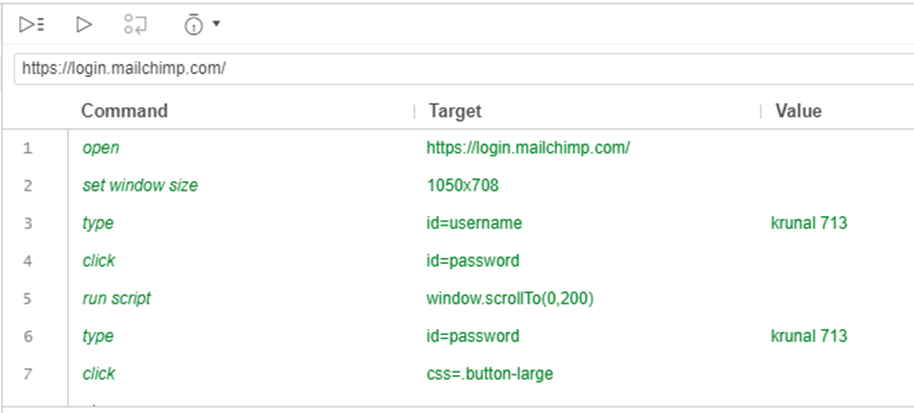
Selenium Commands used :

Command-1 :-set window size

Command-2 :-click

Command-3 :-type

Command-4 :-close



**Test Case 2 :**

**Test URL :** <https://www.shopify.in/tour/ecommerce-website>/

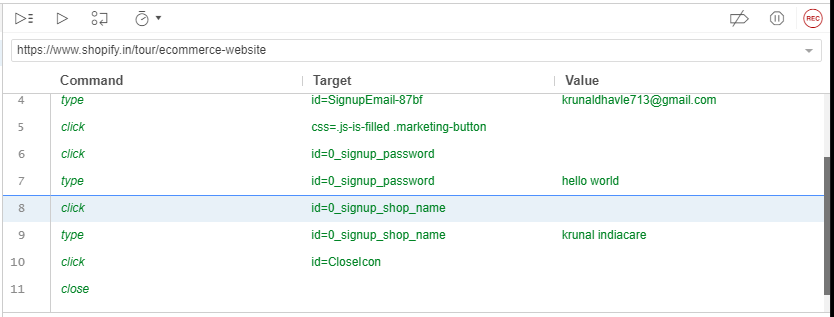
Selenium Commands used :

Command-1 :-open

Command-2 :-type

Command-3 :-click

Command-4 :-close



**Test Case 3 :**

**Test URL :** http://www.google.com/

Selenium Commands used :

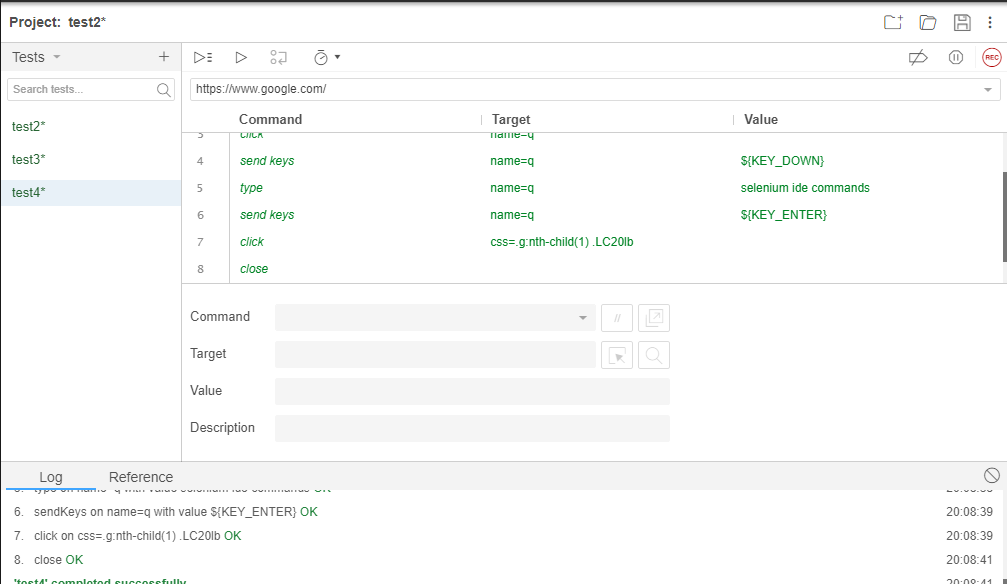
Command-1 :-open

Command-2 :-mouse

Command-3 :-send

Command-4 :-click

Command-5 :-close



**Test Case 4 :**

**Test URL :** https://www.netflix.com/in/Login

Selenium Commands used :

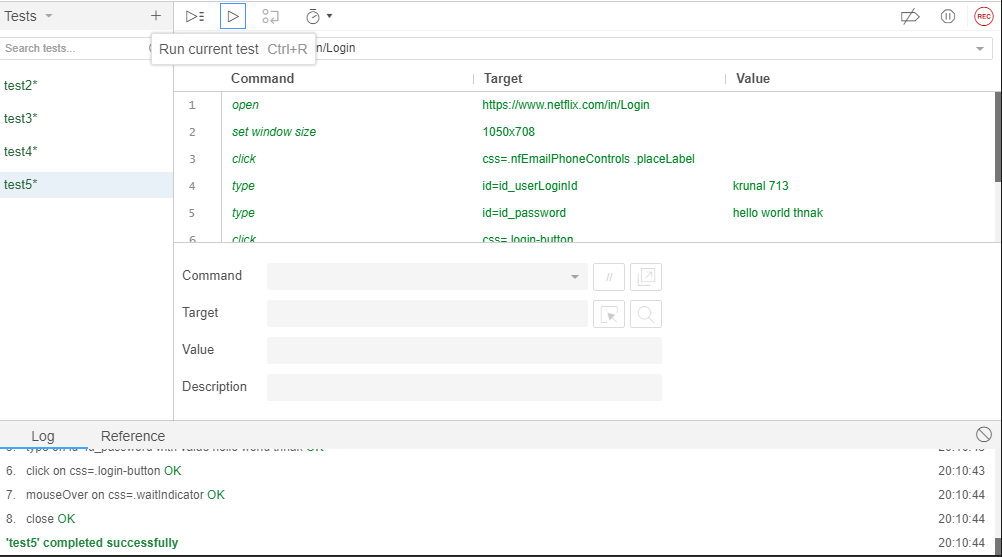
Command-1 :- wait for element

Command-2 : click

Command-3 :- assert checked

Command-4 :- type

Command-5 :- close



Date: 25/08/2020

**Practical No 2**

**AIM:** Install Selenium server and demonstrate it using a script in Java.

**Theory**

**Selenium Java**

Selenium is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers. It provides extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the W3C Web-Driver specification that lets you write interchangeable code for all major web browsers.

**ChromeWebDriver**

WebDriver is an open source tool for automated testing of webapps across many browsers.It provides capabilities for navigating to web pages, user input, JavaScript execution,and more. ChromeDriver is a standalone server that implements the W3C WebDriver standard. ChromeDriver is available for Chrome on Android and Chrome on Desktop (Mac, Linux, Windows and ChromeOS).

**Eclipse IDE**

Eclipse IDE is a famous IDE for java development , Eclipse IDE is open source and has been long in development since 2001. It is licensed under ”Eclipse Public License”

Eclipse has a lot of extensions and now has many IDEs for different Programming languages as well.

**Installation of Selenium server in Eclipse IDE**

Step-1 :- Install JDK,Eclipse IDE for ”java developers”.

Step-2 :- Download ”Chromedriver”,”selenium-java-3.13.0.zip” and ”selenium-server- standalone-3.141.59.jar”.

Step-3 :- Launch the Eclipse IDE.

Step-4 :- Create a new java project give a suitable name and save it.

Step-5 :- Eclipse IDE will open the project. Now add a new java class and give it a name.

Step-6 :- To include the libraries and ”jar” files, Unzip the zipped archives,next ”rightclick on the project icon → BuildP ath → Conf igureBuildP ath..”

Step-7 :- Here we will choose ”Add External JARs...” button and navigate to the ”.jar”

file we downloaded/extracted and select them to include in our project. Software Testing & Quality Assurance

Step-8 :- Save the configuration. You have successfully installed selenium for java. Step-9 :- For linking the chrome driver you need to include it in the code by adding a line

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

**Testing a Website using selenium(java) in eclipsed IDE**

Step-1 :- Import all the important classes we need from the jar packages we included earlier.

import org.openqa.selenium.By;

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

Step-2 :- Next we will set the path property for chromedriver.

Step-3 :- Next Create a WebDriver object for accessing web contents.

Step-4 :- Use ”WebDriver.get()” method to get the base URL .

Step-5 :- we can manipulate the browser DOM elements using ”WebDriver.manage() method”.

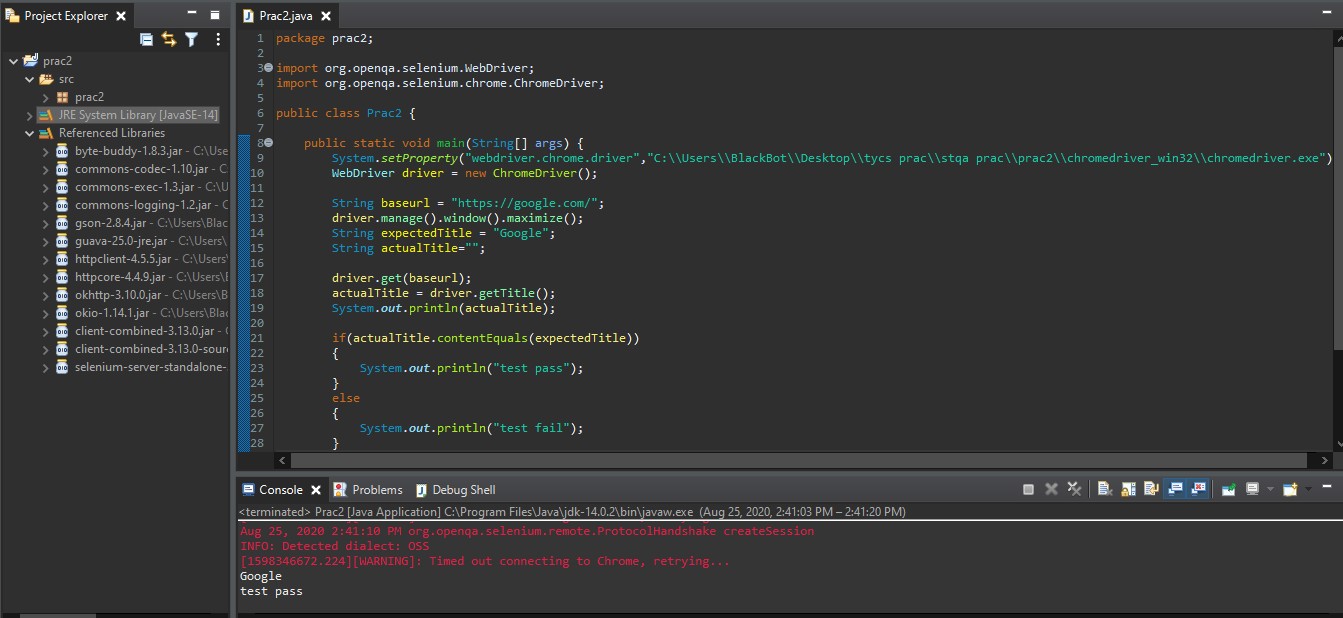
Step-6 :- Next we have a lot of methods to fetch data from the webpage and we can also manipulate them for the ”By” class.

Step-7 :- Once you are done with the testing close the window using ’WebDriver.close()” method.

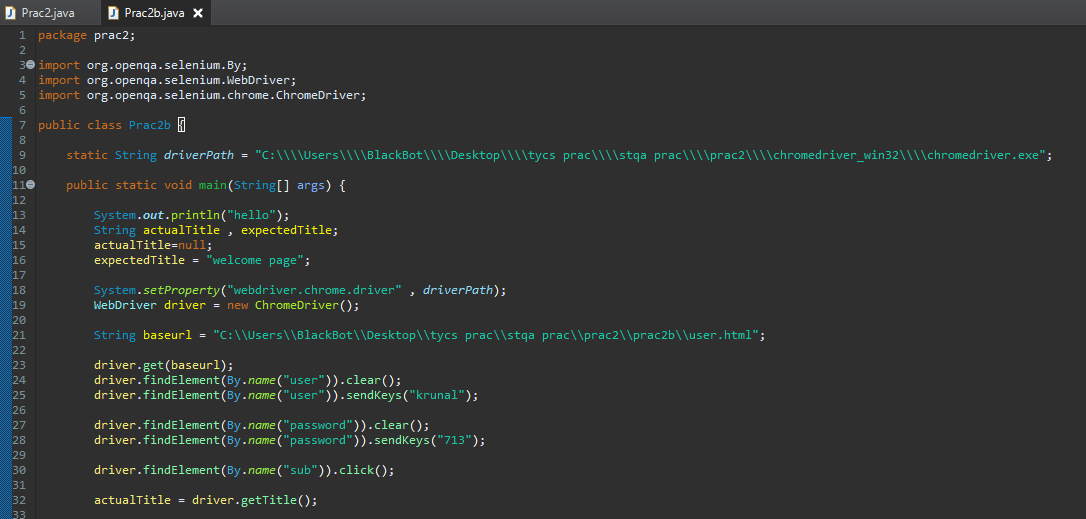
Step-8 :- The base URL can be any site hosted either on localserver or on the world wide web.

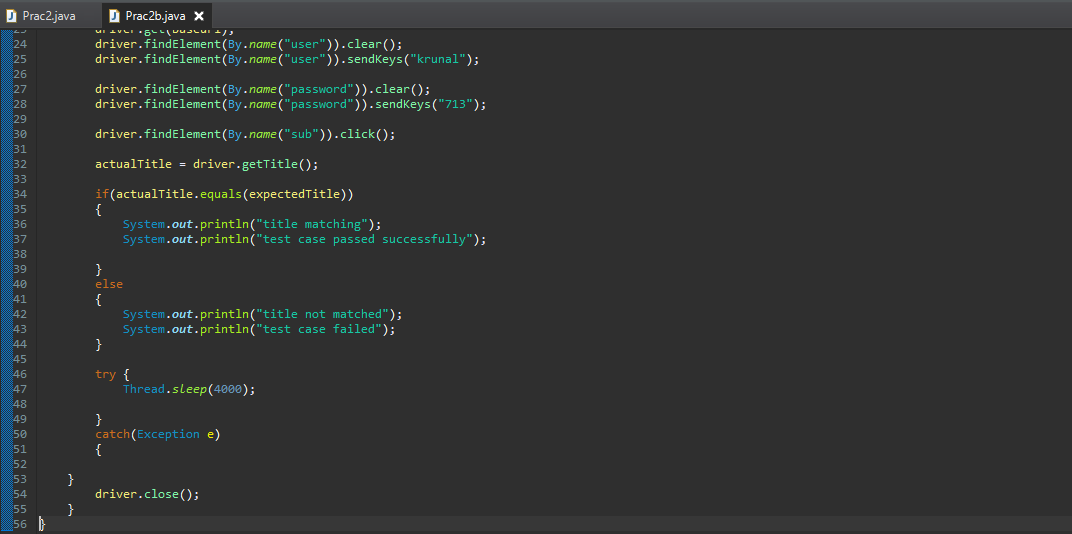
**Code**

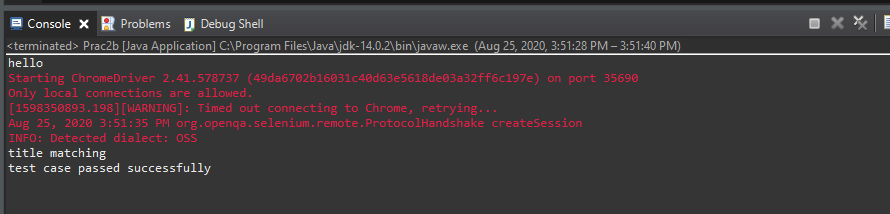
**2A. Test on an External website**



**2B. Test on an Local Webpage**





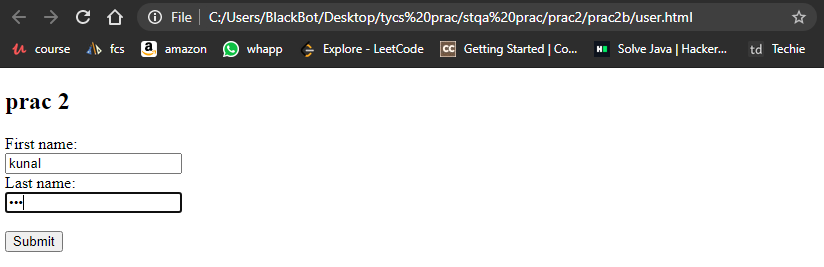


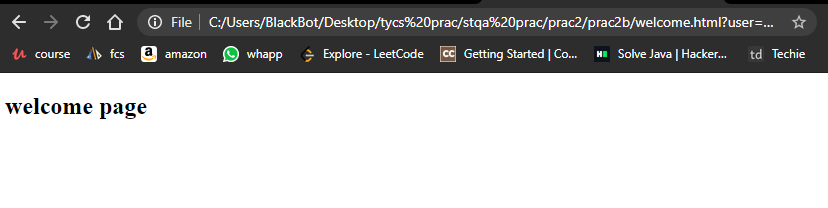
**User.html**

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h2>prac 2 </h2>  <form action="welcome.html">  <label for="fname">First name:</label><br>  <input type="text" id="fname" name="user"><br>  <label for="lname">Last name:</label><br>  <input type="password" id="lname" name="password"><br><br>  <input type="submit" value="Submit" name="sub" >  </form>  </body>  </html> |

**Welcome.html**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> welcome page </title>  </head>  <body>  <h2>welcome page</h2>  </body>  </html> |





**Practical no 3**

**AIM:** Write and test a program to login a specific web page using JUnit.

**Theory**

**What is Junit**

The JUnit Platform serves as a foundation for launching testing frameworks on the JVM.

It also defines the TestEngine API for developing a testing framework that runs on the

platform. Furthermore, the platform provides a Console Launcher to launch the platform

from the command line and a JUnit 4 based Runner for running any TestEngine on the

platform in a JUnit 4 based environment. First-class support for the JUnit Platform also

exists in popular IDEs like eclipse.

**Annotations Used in Junit**

**@Test** Denotes that a method is a test method. Unlike JUnit 4’s @Test annotation, this

annotation does not declare any attributes, since test extensions in JUnit Jupiter

operate based on their own dedicated annotations. Such methods are inherited

unless they are overridden.

**@BeforeAll** Denotes that the annotated method should be executed before all @Test,

@RepeatedTest, @ParameterizedTest, and @TestFactory methods in the current

class; analogous to JUnit 4’s @BeforeClass.

**@AfterAll** Denotes that the annotated method should be executed after all @Test,

@RepeatedTest, @ParameterizedTest, and @TestFactory methods in the current

class; analogous to JUnit 4’s @AfterClass.

**@BeforeEach** Denotes that the annotated method should be executed before each

@Test, @RepeatedTest, @ParameterizedTest, or @TestFactory method in the current class; analogous to JUnit 4’s @Before. Such methods are inherited unless they

are overridden.

**@AfterEach** Denotes that the annotated method should be executed after each @Test,

@RepeatedTest, @ParameterizedTest, or @TestFactory method in the current class;

analogous to JUnit 4’s @After. Such methods are inherited unless they are overridden.

**Steps for Testing a site using Junit**

Step-1 :- Create a new Project and give it a name.

Step-2 :- After the initialization create a new package inside the project and give it a

name.

Step-3 :- To include the libraries and ”jar” files, Unzip the zipped archives,next ”right

click on the project icon → BuildP ath → Conf igureBuildP ath..”Step-4 :- Here we will choose ”Add External JARs...” button and navigate to the ”.jar”

file we downloaded/extracted and select them to include in our project.

Step-5 :- Save the configuration. You have successfully installed selenium for java.

Step-6 :- For linking the chrome driver you need to include it in the code by adding a

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

Step-7 :- Next create a new JUnit Test Case file give it a name.

Step-8 :- Import the following libraries

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebDriver;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import junit.framework.Assert;

Step-9 :- Add the code.

**Code:-**

|  |
| --- |
| import static org.junit.Assert.\*;  import org.junit.After;  import org.junit.Assert;  import org.junit.Before;  import org.junit.Test;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  public class JunitEx {  WebDriver driver = null;  @Before  public void setup() {  System.setProperty("webdriver.chrome.driver" , "E:\\tycs\\stqa prac\\prac3\\chromedriver\_win32\\chromedriver.exe" );  driver = new ChromeDriver();  driver.manage().window().maximize();  }    @Test  public void test() throws InterruptedException {    driver.get("http://thedemosite.co.uk/savedata.php");  driver.findElement(By.xpath("//input[@name='username']")).sendKeys("krunal71");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='password']")).sendKeys("kd713");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='FormsButton2']")).click();  Thread.sleep(1000);  Thread.sleep(2000);  Assert.assertTrue("invalid credential" , driver.getTitle().contains("Add a user - FREE PHP code and SQL"));  System.out.println(" page title is verified user is able to login ");  }    @Test  public void demotest() throws InterruptedException {    driver.get("http://demo.guru99.com/test/newtours/");  driver.findElement(By.xpath("//input[@name='userName']")).sendKeys("krunal713");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='password']")).sendKeys("kd713");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='submit']")).click();  Thread.sleep(1000);  Thread.sleep(2000);  Assert.assertTrue("invalid credential" , driver.getTitle().contains("Welcome: Mercury Tours"));  System.out.println(" page title is verified user is able to login ");  }    @After  public void aftertest(){  driver.quit();  }  } |

**Output:**

