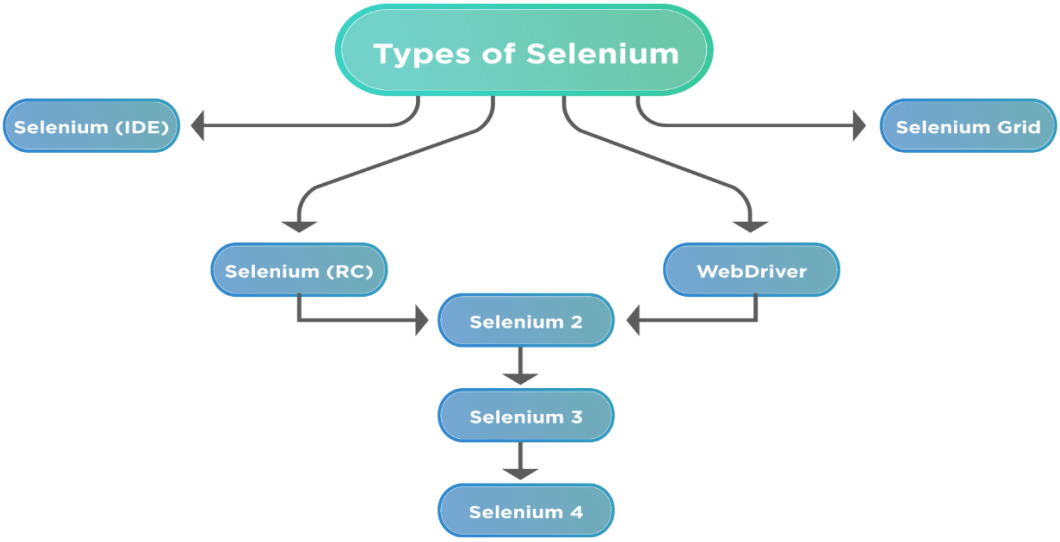
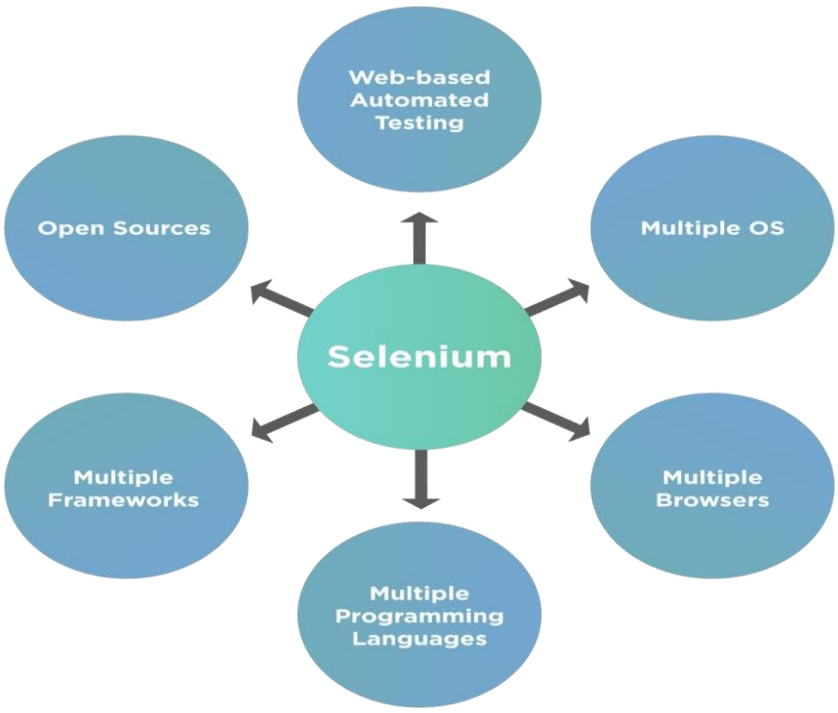
SELENIUM TESTING

Selenium

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms.



Features



Site to learn: https://toolsqa.com/selenium-webdriver/selenium-tutorial/

Installation of Selenium:

Install Selenium WebDriver in Eclipse for a Java project

1. Install Java:
   Ensure that you have Java installed on your system. You can download and install the latest Java Development Kit (JDK) from the official Oracle website.
   https://www.oracle.com/in/java/technologies/downloads/

2. Install Eclipse:
   Download and install the latest version of Eclipse for Java development from the official Eclipse website. https://www.eclipse.org/downloads/

3. Open Eclipse:
   Launch Eclipse and create a new Java project or open an existing one.

4. Create a New Java Project:
   Go to File > New > Java Project. Enter a project name and click Finish.

5. Configure Java Build Path:
   Right-click on the newly created project and select Build Path > Configure Build Path.

6. Download Selenium WebDriver:

   Download the Selenium WebDriver for Java from the Selenium official website: https://www.selenium.dev/downloads/

   Click Java-> jar files download -> UnZip it.

7. Add Selenium WebDriver Libraries:

   Navigate to the Libraries tab.

   Click on Classpath and then Add External JARs.

   Locate and select the Selenium WebDriver JAR files you've downloaded. These files typically have names like selenium-java-x.y.z.jar (where x, y, and z represent version numbers).

   Click Apply and Close.

8. Add WebDriver Executable:

   If you're using WebDriver for a specific browser (e.g., Chrome, Firefox), you'll also need to download the corresponding WebDriver executable and add it to your project. You can download WebDriver executables from the WebDriver GitHub repository: https://github.com/SeleniumHQ/selenium

   Chrome WebDrivers : Chrome -> settings -> about chrome -> see your chrome Version and select appropriate web driver https://chromedriver.chromium.org/downloads

9. Write Selenium Code:

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

public class MySeleniumTest {
   public static void main(String[] args) {
      // Set the path to the WebDriver executable (e.g., chromedriver)
```

```
        System.setProperty("webdriver.chrome.driver",
"path/to/chromedriver"); // use forward slash

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

        // Navigate to a website
        driver.get("https://amazon.com");

        // Perform actions, assertions, etc.

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

How to locate Web Elements?

1. Open the Web Page:
   Navigate to the web page where you want to locate the element.

2. Open Developer Tools:
   Right-click on the element you want to inspect and select "Inspect"

3. Access Elements Tab:
   In the developer tools, you'll see various tabs. Click on the "Elements" tab (or similar) to view the HTML structure of the web page.

4. Locate the Element:
   Use the mouse to hover over the HTML elements displayed in the developer tools. As you hover over the elements in the HTML view, the corresponding parts of the web page will be highlighted. Locate and click on the HTML element that represents the element you want to interact with.

5. Inspect Element Details:
   Once you've located the element in the HTML structure, you can view its attributes (e.g., ID, class, name) and other details. Take note of the attributes that can be used to uniquely identifythe element.

<u>Locators</u>

By ID:

```
WebElement elementById = driver.findElement(By.id("elementId"));
```

By Name:

```
WebElement elementByName = driver.findElement(By.name("elementName"));
```

By XPath:

```
WebElement elementByXPath = driver.findElement(By.xpath("//input[@id='elementId']"));
```

By CSS Selector:

```
WebElement elementByCssSelector = driver.findElement(By.cssSelector("#elementId"));
```

By Class Name:

```
WebElement elementByClassName = driver.findElement(By.className("className"));
```

By Tag Name:

```
WebElement elementByTagName = driver.findElement(By.tagName("input"));
```

By Link Text:

```
WebElement elementByLinkText = driver.findElement(By.linkText("Link Text"));
```

By Partial Link Text:

```
WebElement elementByPartialLinkText = driver.findElement(By.partialLinkText("Partial Link Text"));
```

Types of Xpath In Selenium

Absolute Xpath
/html/body/div[2]/div[1]/div/h4[1]/b/html[1]/body[1]/div[2]/div[1]/div[1]/h4[1]/b[1]

Relative Xpath:

Relative XPath: //div[@class='featured-box cloumnsize1']//h4[1]//b[1]

Dynamic XPath In Selenium
1. Basic Xpath
Xpath=//input[@name='uid']

2. Contains()
Xpath=//*[contains(@name,'btn')]

3. Using OR & AND
Xpath=//*[@type='submit' or @name='btnReset']
Xpath=//input[@type='submit' and @name='btnLogin']

4. Xpath Starts-with
Xpath=//label[starts-with(@id,'message')]

5. XPath Text() Function
Xpath=//td[text()='UserID']

XPath axes methods

1. Following
a. Xpath=//*[@type='text']//following::input
b. Xpath=//*[@type='text']//following::input[1]

2. Ancestor
a. Xpath=//*[text()='Enterprise Testing']//ancestor::div
b. Xpath=//*[text()='Enterprise Testing']//ancestor::div[1]

3. Child
a. Xpath=//*[@id='java_technologies']//child::li

b. Xpath=//*[@id='java_technologies']//child::li[1]

4. Preceding
a. Xpath=//*[@type='submit']//preceding::input
b. Xpath=//*[@type='submit']//preceding::input[1]
5. Following-sibling
Xpath=//*[@type='submit']//following-sibling::input

6. Parent
a. Xpath=//*[@id='rt-feature']//parent::div
b. Xpath=//*[@id='rt-feature']//parent::div[1]

7. Self
Xpath =//*[@type='password']//self::input

8. Descendant
a. Xpath=//*[@id='rt-feature']//descendant::a
b. Xpath=//*[@id='rt-feature']//descendant::a[1]

isDisplayed, isEnabled, isSelected Methods in Selenium

isSelected:

- isSelected is a method applicable to checkboxes, radio buttons, and options within a dropdown.
- It checks if an element is selected or not, primarily used with input elements like checkboxes and radio buttons.
- Returns a boolean value - true if the element is selected, false if not.

isEnabled:

- isEnabled is a method applicable to input elements like text fields, buttons, etc.
- It checks if an element is enabled for interaction or not, considering factors like being visible, not disabled, etc.
- Returns a boolean value - true if the element is enabled, false if not.

isDisplayed:

- isDisplayed is a method applicable to all web elements (e.g., buttons, links, text fields, etc.).
- It checks if an element is currently visible on the page or not.
- Returns a boolean value - true if the element is displayed, false if not.

Example: -> Checkbox and Radio Button testing

```
package Seleniumpkg;


import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

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

import org.openqa.selenium.WebElement;


// isSelected() checks that if an element is selected on the web page or not

//isEnabled() check if the web element is enabled or disabled within the web page

// isDisplayed() check whether an element is displayed on a  web page or not


public class methoddemo {

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

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

		WebDriver driver = new ChromeDriver();

	driver.get("https://softwaretestingo.blogspot.com/2020/08/checkbox-radio-button.html");
```

```
WebElement radio1 = driver.findElement(By.id("java"));
 System.out.println("Radio Button 1:");
 if (!radio1.isSelected()) {
    radio1.click();
    System.out.println("Clicked the java radio button.");
 }
System.out.println("Selected: " + radio1.isSelected());
 System.out.println("Displayed: " + radio1.isDisplayed());
 System.out.println("Enabled: " + radio1.isEnabled());
 System.out.println();

 WebElement radio2 = driver.findElement(By.id("checkbox"));
System.out.println("Radio Button 2:");
System.out.println("Selected: " + radio2.isSelected());
System.out.println("Displayed: " + radio2.isDisplayed());
System.out.println("Enabled: " + radio2.isEnabled());
System.out.println();


 WebElement checkbox1 = driver.findElement(By.id("sing"));
 System.out.println("Checkbox 1:");
 System.out.println("Selected: " + checkbox1.isSelected());
 System.out.println("Displayed: " + checkbox1.isDisplayed());
 System.out.println("Enabled: " + checkbox1.isEnabled());
```

```
        System.out.println();

        WebElement checkbox2 = driver.findElement(By.id("code"));
        System.out.println("Checkbox 2:");
        if (!checkbox2.isSelected()) {
            checkbox2.click();
            System.out.println("Clicked the coding checkbox.");
        }
        System.out.println("Selected: " + checkbox2.isSelected());
        System.out.println("Displayed: " + checkbox2.isDisplayed());
        System.out.println("Enabled: " + checkbox2.isEnabled());
    }
}
```

Output:

```
Radio Button 1:
Clicked the java radio button.
Selected: true
Displayed: true
Enabled: true

Radio Button 2:
Selected: false
Displayed: true
Enabled: true

Checkbox 1:
Selected: false
Displayed: true
Enabled: true

Checkbox 2:
Clicked the coding checkbox.
Selected: true
Displayed: true
Enabled: true
```

KEYBOARD HANDLING EXAMPLE

```
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.interactions.Actions;

public class keyhandling {
   public static void main(String[] args) {

      System.setProperty("webdriver.chrome.driver", "path to chrome");
      WebDriver driver = new ChromeDriver();
      driver.get("https://www.amazon.in/");

      // Find the search input element by its name attribute
      WebElement searchBox = driver.findElement(By.name("field-
keywords"));

      // Create an Actions object to perform keyboard actions
      Actions actions = new Actions(driver);

      // Perform multiple key actions in sequence
      actions
         .sendKeys(searchBox, "laptops") // Type "laptops" in the search
box
         .build().perform();
      System.out.println("Typed 'laptops'");

      actions
         .keyDown(Keys.CONTROL) // Hold down the Control key
         .sendKeys("a") // Press 'A' key (select all)
         .keyUp(Keys.CONTROL) // Release the Control key
         .build().perform();
      System.out.println("Selected all");

      actions
```

```
            .sendKeys(Keys.BACK_SPACE) // Press Backspace (clear the
input)
            .build().perform();
        System.out.println("Cleared input");

        actions
            .sendKeys(Keys.RETURN) // Press Enter
            .build().perform();
        System.out.println("Pressed Enter");

        try {
            Thread.sleep(3000); // Sleep for 3 seconds to see the results
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Close the WebDriver
        driver.quit();
    }
}
```

Output:

```
Typed 'laptops'
Selected all
Cleared input
Pressed Enter
```

MOUSE HANDLING EXAMPLE

```
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 mousehandling {
```

```
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path to chrome");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.amazon.in/");

        try {
            Actions actions = new Actions(driver);

            //  search bar
            WebElement element =
driver.findElement(By.id("twotabsearchtextbox"));
                // Right-click (Context Click)
            actions.contextClick(element).build().perform();
            System.out.println("Performed Context Click");
            Thread.sleep(2000);
            actions.click(element).build().perform();
            System.out.println("Performed Left Click");
            Thread.sleep(2000);
            // Double-click
            actions.doubleClick(element).build().perform();
            System.out.println("Performed Double Click");
            Thread.sleep(2000);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            driver.quit();
        }
    }
}
```

Output

```
Performed Context Click
Performed Left Click
Performed Double Click
```

Excel Example

```
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFCell;

import java.io.*;

public class ExcelRW {
   public static void main(String[] args) {
      // Specify your Excel file paths
      String inputFilePath = "D://studentdata.xlsx"; // Input file exist
      String outputFilePath = "D://output.xlsx";     // Output file

      // Read Excel file
      try {
         FileInputStream fileInputStream = new
FileInputStream(inputFilePath);
         XSSFWorkbook workbook = new
XSSFWorkbook(fileInputStream);
         XSSFSheet sheet = workbook.getSheetAt(0);

         // Read data from cell A1
         XSSFRow row = sheet.getRow(0);
         XSSFCell cell = row.getCell(0);
         String cellData = cell.getStringCellValue();
         System.out.println("Data in cell A1: " + cellData);

         // Write Excel file
         FileOutputStream fileOutputStream = new
FileOutputStream(outputFilePath);

         // Create a new workbook and sheet
         XSSFWorkbook newWorkbook = new XSSFWorkbook();
         XSSFSheet newSheet = newWorkbook.createSheet("Sheet1");
```

```
        // Create a new row and cell
        XSSFRow newRow = newSheet.createRow(0);
        XSSFCell newCell = newRow.createCell(0);

        // Set a value in cell A1
        newCell.setCellValue("Hello, Excel!");

        // Write the new workbook to the output file
        newWorkbook.write(fileOutputStream);

        // Close input and output streams
        fileInputStream.close();
        fileOutputStream.close();

        System.out.println("Data written to " + outputFilePath);
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Output

```
Data in cell A1: Firstname
Data written to D://output.xlsx
```

Links:

https://www.scaler.com/topics/how-to-read-data-from-excel-in-selenium/

https://www.edureka.co/community/52118/read-test-from-excel-sheet-facebook-login-selenium-webdriver

https://learn-automation.com/read-numeric-data-excel-using-apache-poi-selenium-webdriver/

https://automationtesting.in/read-data-from-excel-using-column-number/

Screenshot Example

```
import java.io.File;
import java.io.IOException;

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.io.FileHandler;
public class Screenshots {
	public static void main (String args[]) throws
InterruptedException, IOException {
	System.setProperty("webdriver.chrome driver", "path to
chrome");
	WebDriver dri = new ChromeDriver();
	dri.get("https://demoqa.com/buttons");

//Screenshot of a page
	File PageScr = ((TakesScreenshot)
dri).getScreenshotAs(OutputType.FILE);//take screescot
	File PageDest = new File("D://Pic/img1.png");
	FileHandler.copy(PageScr, PageDest);



//Screenshot of an Element
	WebElement Ele = dri.findElement(By.id("doubleClickBtn"));
	File EleScr = ((TakesScreenshot)
Ele).getScreenshotAs(OutputType.FILE);
	File EleDest = new File("D://Pic/img2.png");
	FileHandler.copy(EleScr, EleDest);

//ScreenShot of a Section
```

```
        WebElement Eleme = dri.findElement(By.className("left-
pannel"));
        File SecScr = ((TakesScreenshot)
Eleme).getScreenshotAs(OutputType.FILE);
        File SecDest = new File("D://Pic/img3.png");
        FileHandler.copy(SecScr, SecDest);

        System.out.println("See your Folder");
        dri.quit();
}
}
```

Output

```
See your Folder
```

TEST-NG

Definition

- ***TestNG*** is a testing framework inspired from ***JUnit*** and ***NUnit*** but introducing some new functionality that makes it more powerful and easier to use.

- It is an open-source automated testing framework; where ***NG*** of Test**NG** means **N**ext **G**eneration.

- TestNG is similar to JUnit but it is much more powerful than JUnit but still, it's inspired by JUnit.

- It is designed to be better than JUnit, especially when testing integrated classes. Pay special thanks to *Cedric Beust who is the creator of TestNG*.

USES

TestNG eliminates most of the limitations of the older framework and gives the developer the ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing.

What are the Benefits of TestNG:-

1. *It gives the ability to produce **HTML Reports** of execution*
2. ***Annotations** made testers life easy*
3. *Test cases can be **Grouped & Prioritized** more easily*
4. ***Parallel** testing is possible*
5. *Generates **Logs***
6. *Data **Parameterization** is possible*

Test Case Writing process in TestNG

- **Step 1 -** Write the business logic of the test
- **Step 2 -** Insert TestNG annotations in the code
- **Step 3 -** Add the information about your test (e.g. the class names, methods names, groups names, etc...) in a testng.xml file
- **Step 4 -** Run TestNG

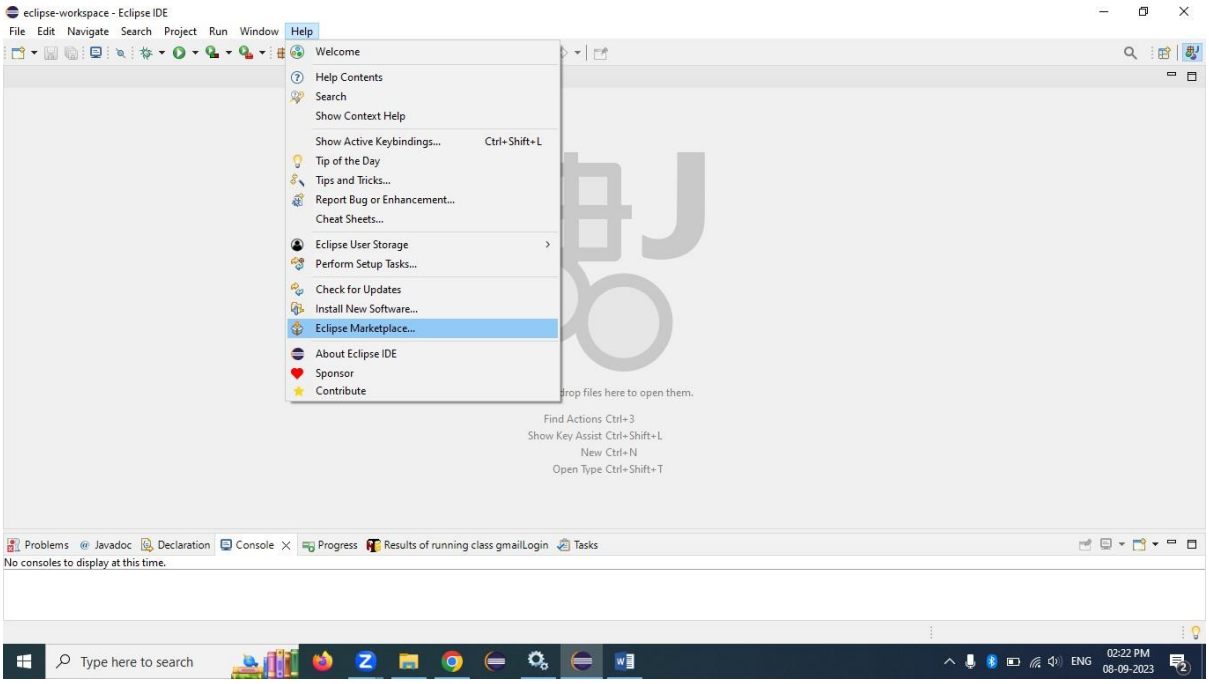
What are the different Annotations are present inTestNG?

- **@BeforeSuite**: The annotated method will be run before all tests in this suite have run.
- **@AfterSuite**: The annotated method will be run after all tests in this suite have run.
- **@BeforeTest**: The annotated method will be run before any test method belonging to the classes inside the tag is run.

- **@AfterTest**: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.
- **@BeforeGroups**: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
- **@AfterGroups**: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
- **@BeforeClass**: The annotated method will be run before the first test method in the current class is invoked.
- **@AfterClass**: The annotated method will be run after all the test methods in the current class have been run.
- **@BeforeMethod**: The annotated method will be run before each test method.
- **@AfterMethod**: The annotated method will be run after each test method.
- **@Test**: The annotated method is a part of a test case.

How to configure TestNG in EclipseStep1: Install TestNG

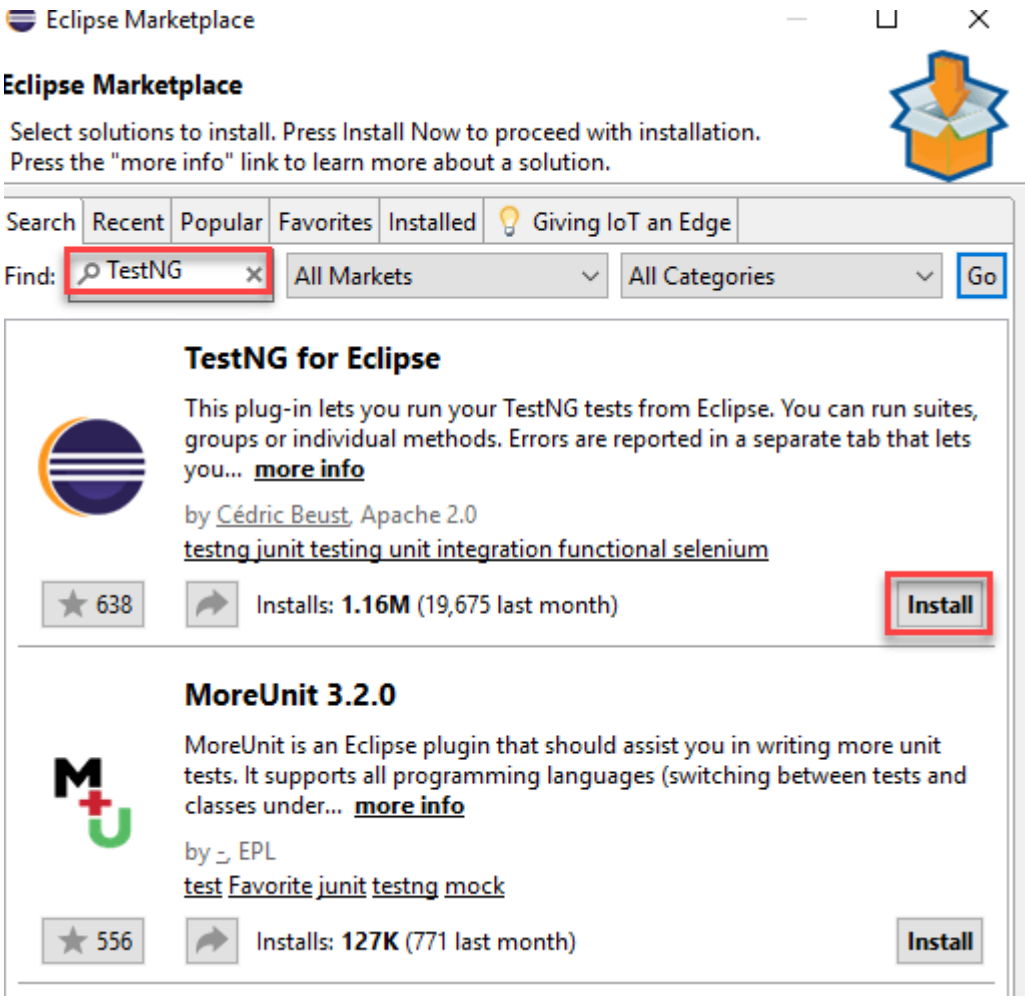
in Eclipse Help→Eclipse marketplace→Click



Step 2: Search for TestNG

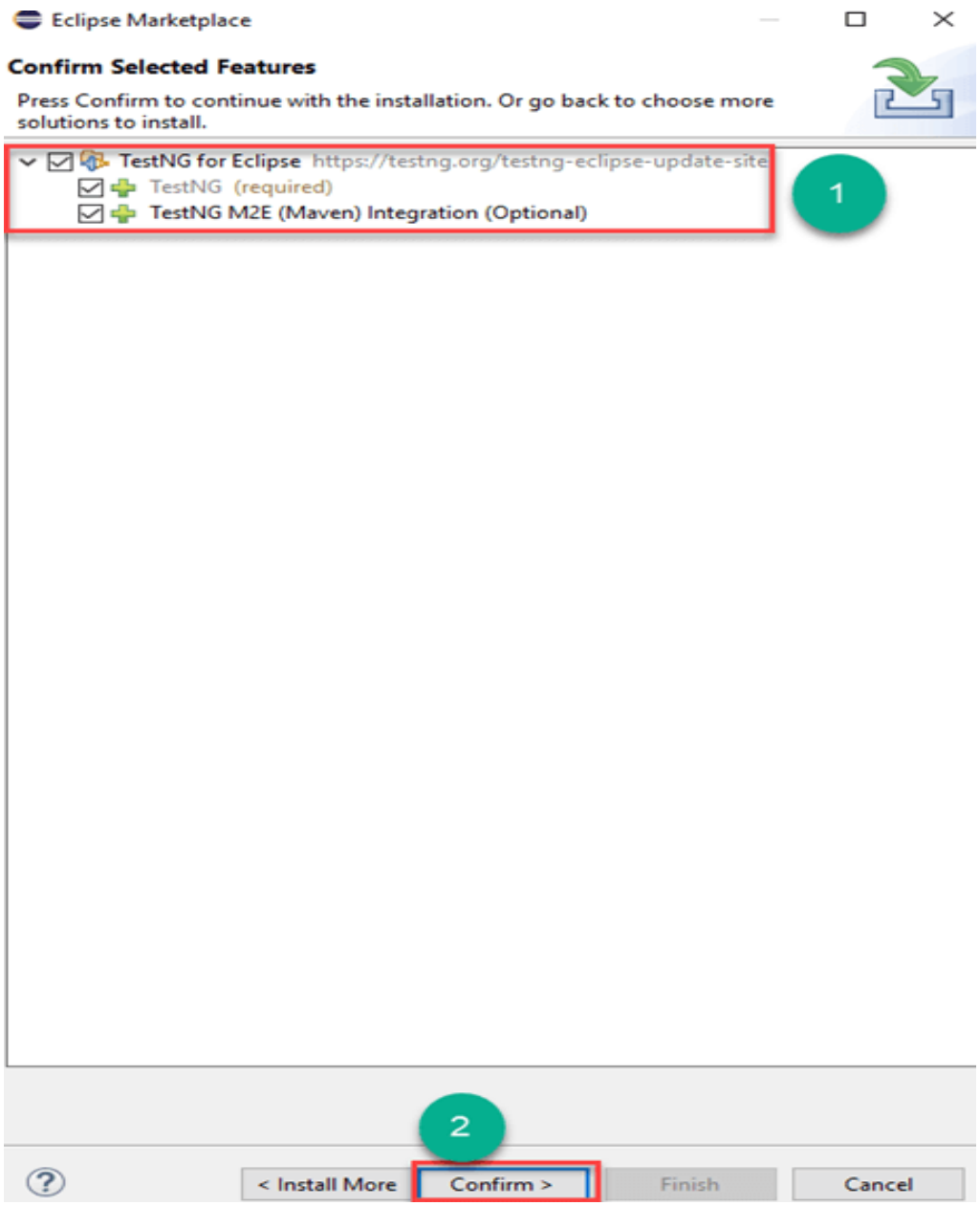
1. Searchbar →Type **TestNG** →Click Enter

2. You will see TestNG →Click **Install**


Eclipse Marketplace
Eclipse Marketplace
Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.
Search
Recent
Popular
Favorites
Installed
Giving IoT an Edge
Find:
TestNG
All Markets
All Categories
Go
TestNG for Eclipse
This plug-in lets you run your TestNG tests from Eclipse. You can run suites, groups or individual methods. Errors are reported in a separate tab that lets you... more info
by Cédric Beust, Apache 2.0
testng junit testing unit integration functional selenium
638
Installs: 1.16M (19,675 last month)
Install
MoreUnit 3.2.0
MoreUnit is an Eclipse plugin that should assist you in writing more unit tests. It supports all programming languages (switching between tests and classes under... more info
by -, EPL
test Favorite junit testng mock
556
Installs: 127K (771 last month)
Install

Step 3: After clicking **Install,** it shows like belowpage. **It takes some time to install**…

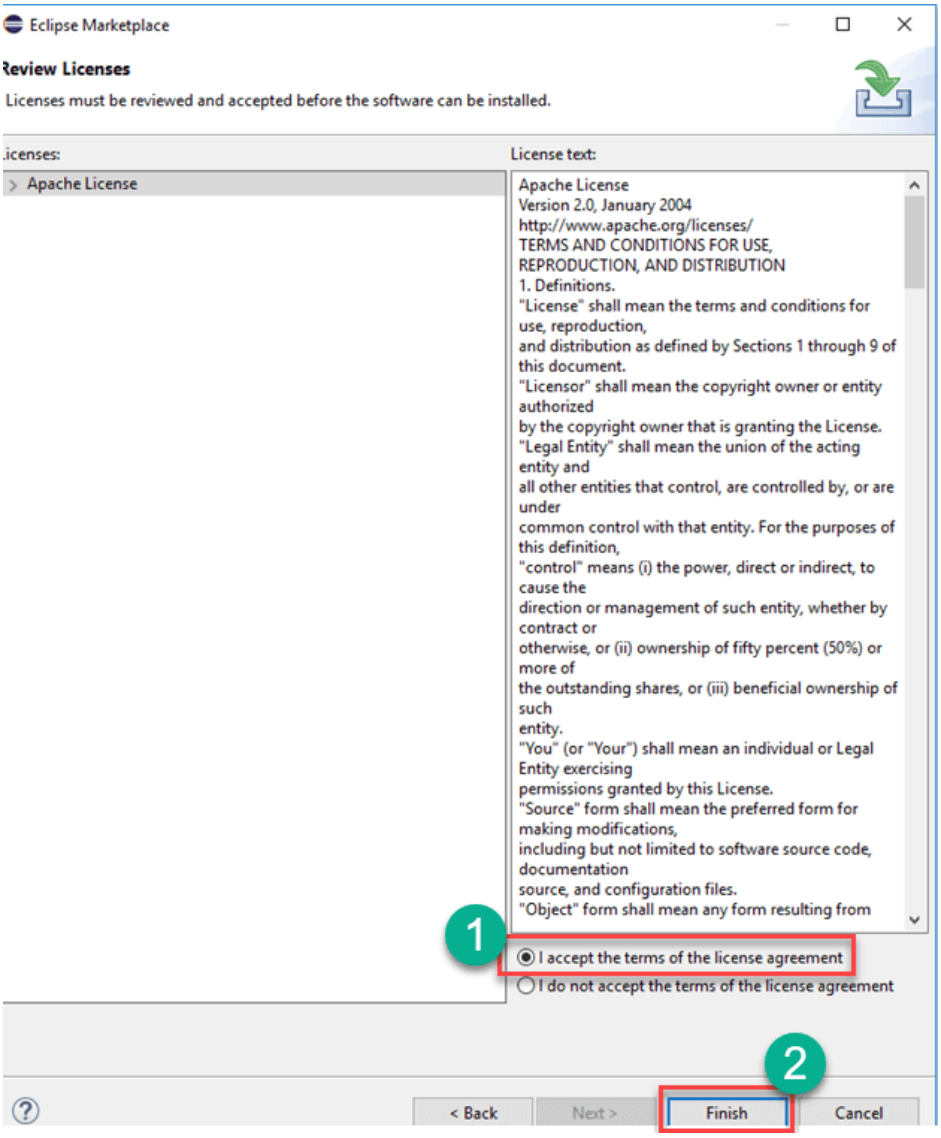
1. Click **confirm**



Step 4: It ask for the license to accept

1. Click **I accept the terms for the licenseagreement.**

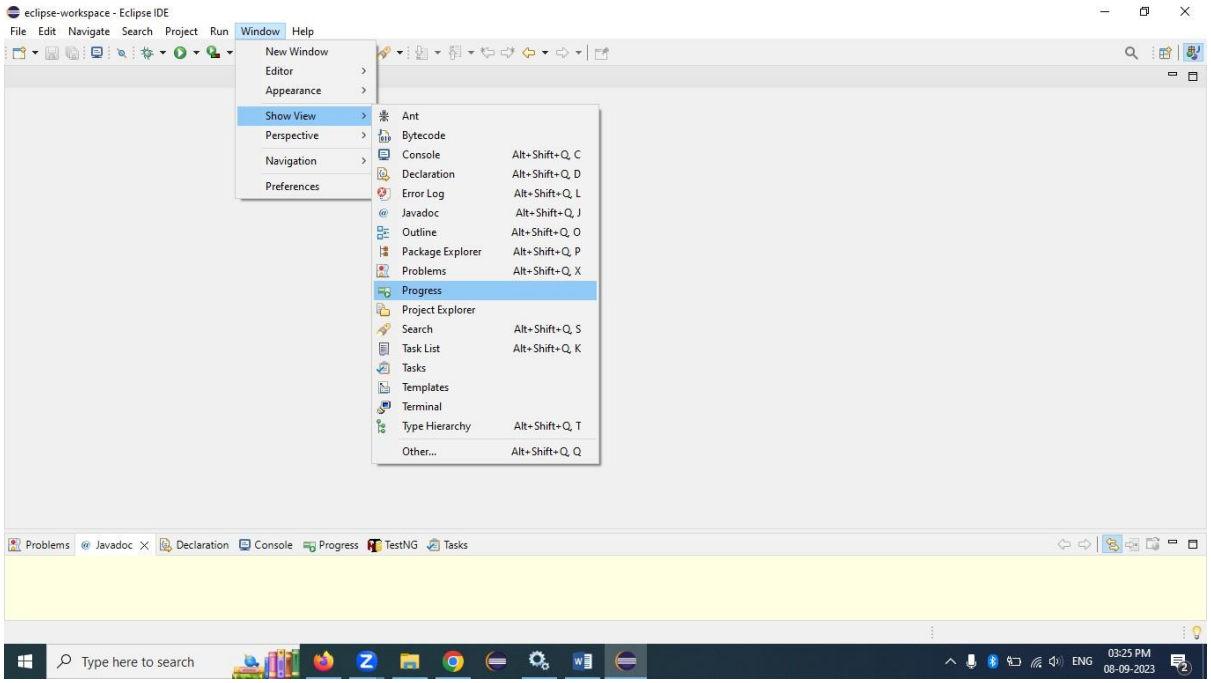
2. Click **Finish.**

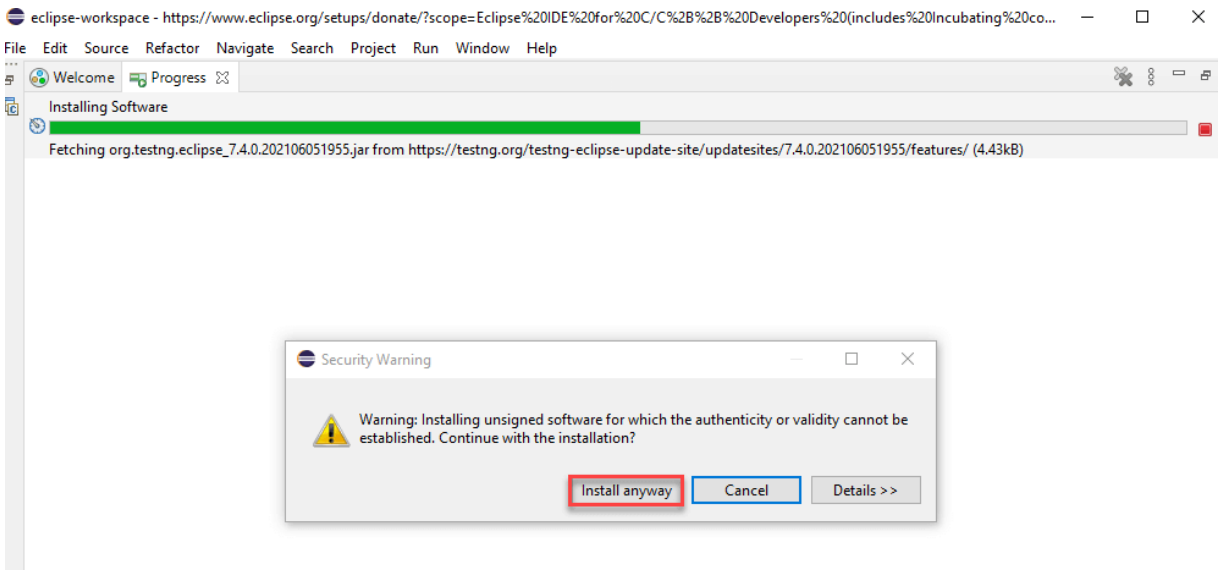


Step 5: Viewing **installation progress**

1. **window→Show view→Progress**
2. You will see the **progress** at below
3. I f it asks for any security warning→Click

   Install anyway.

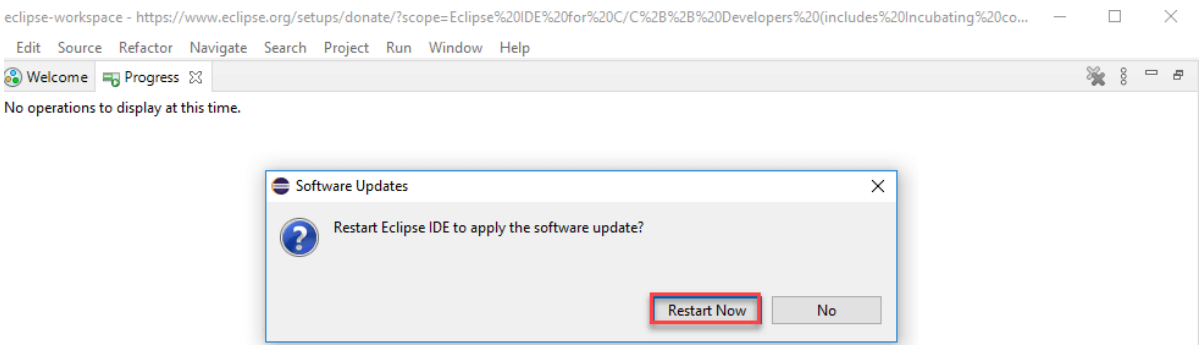




Step 6: After Installation

1. It asks for **Restart now.**

2. Click **Restart Now**

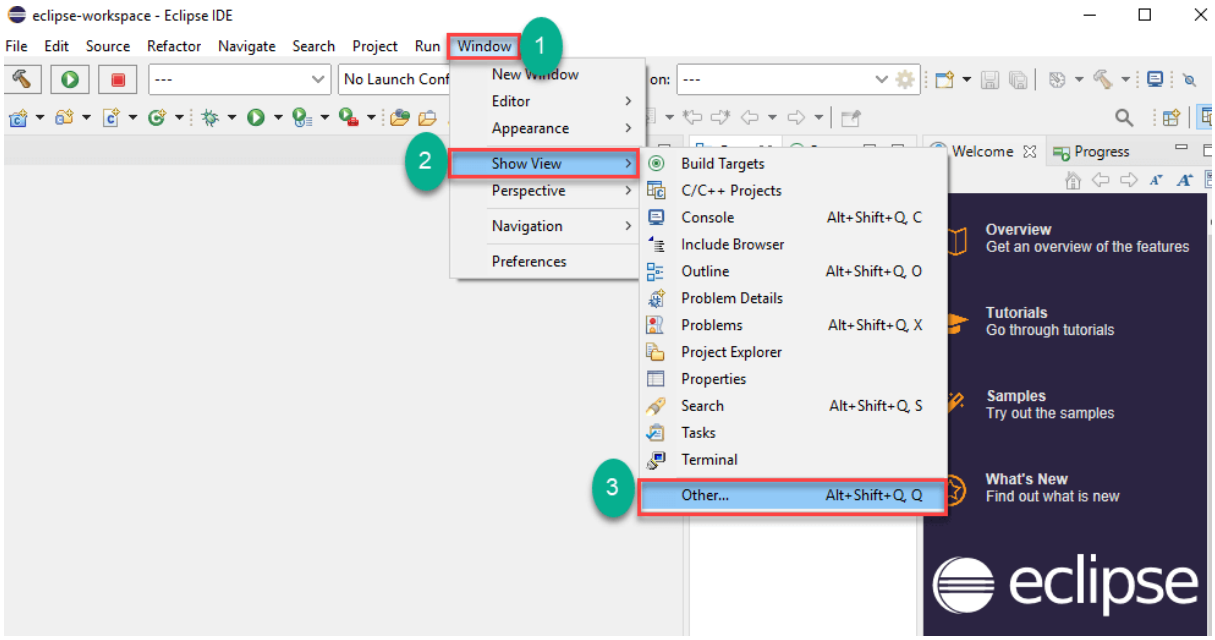


Step 7: Checking whether **TestNG is installed ornot?**

1. Click **window**

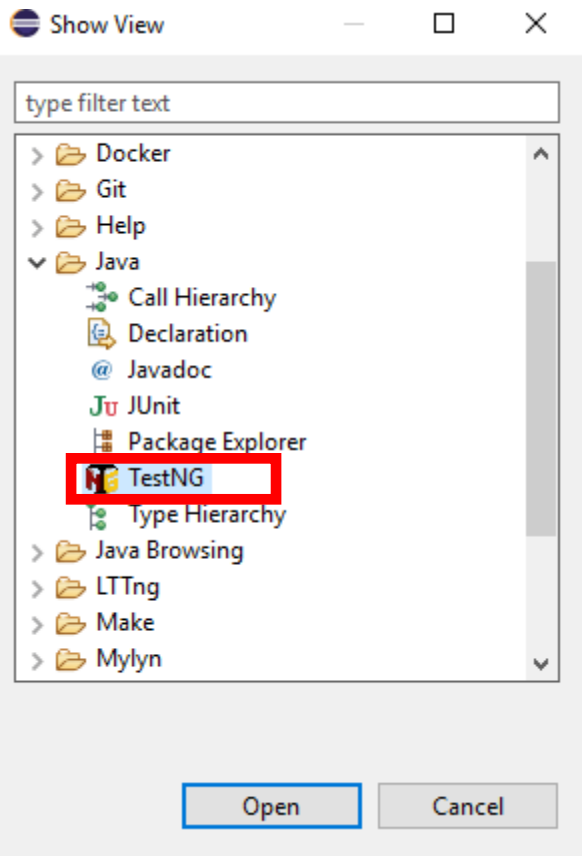
2. Click **show view**

3. Click **others**



Step 8: TestNG is present??

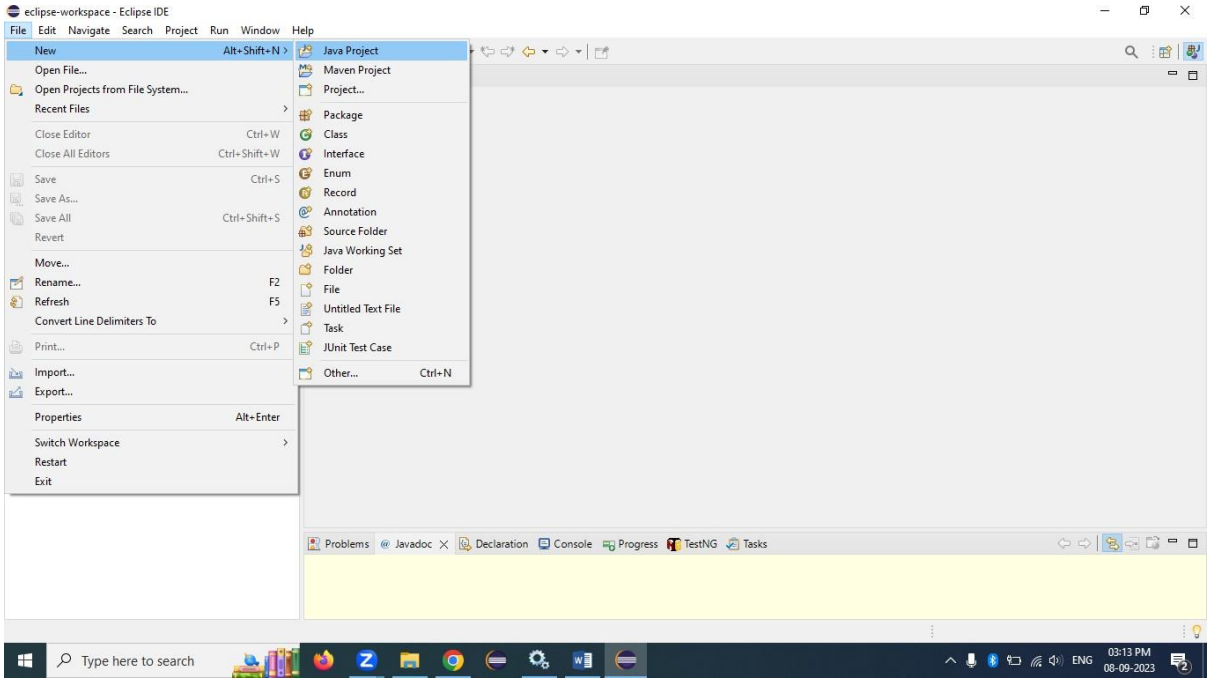
1. Click **Java Folder**
2. Inside **TestNG** is present



How to Create New Java project?

Step 1: Creating new project

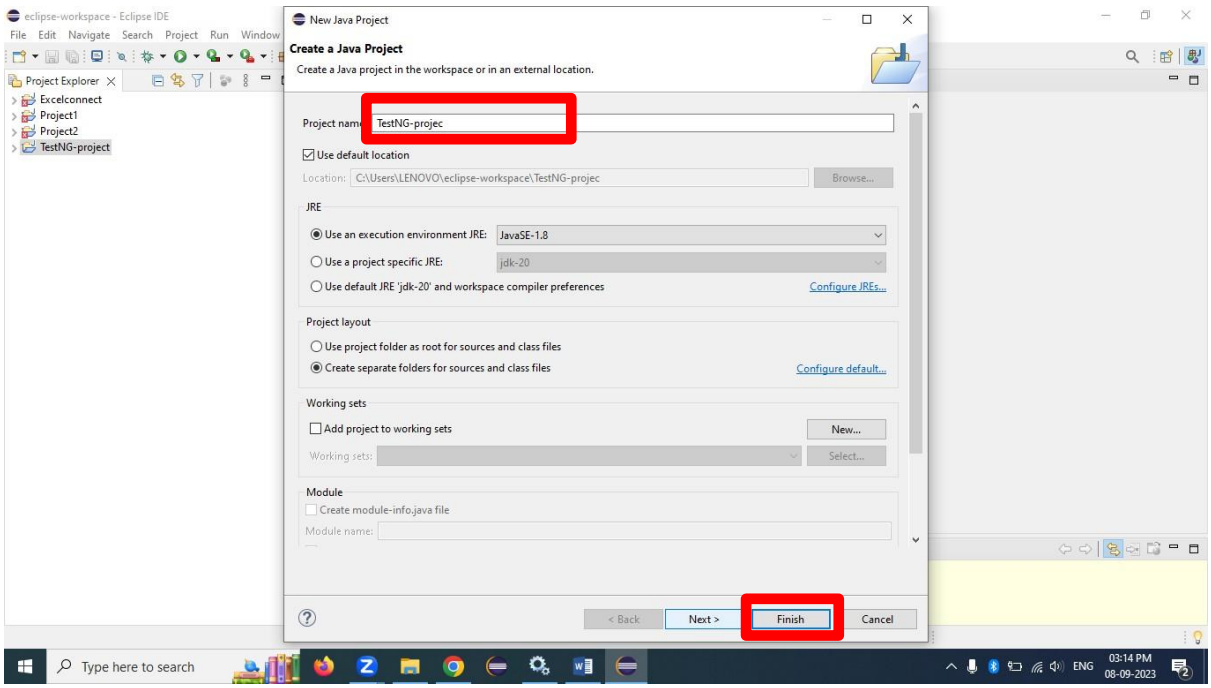
1. **File→ New→ Java project**



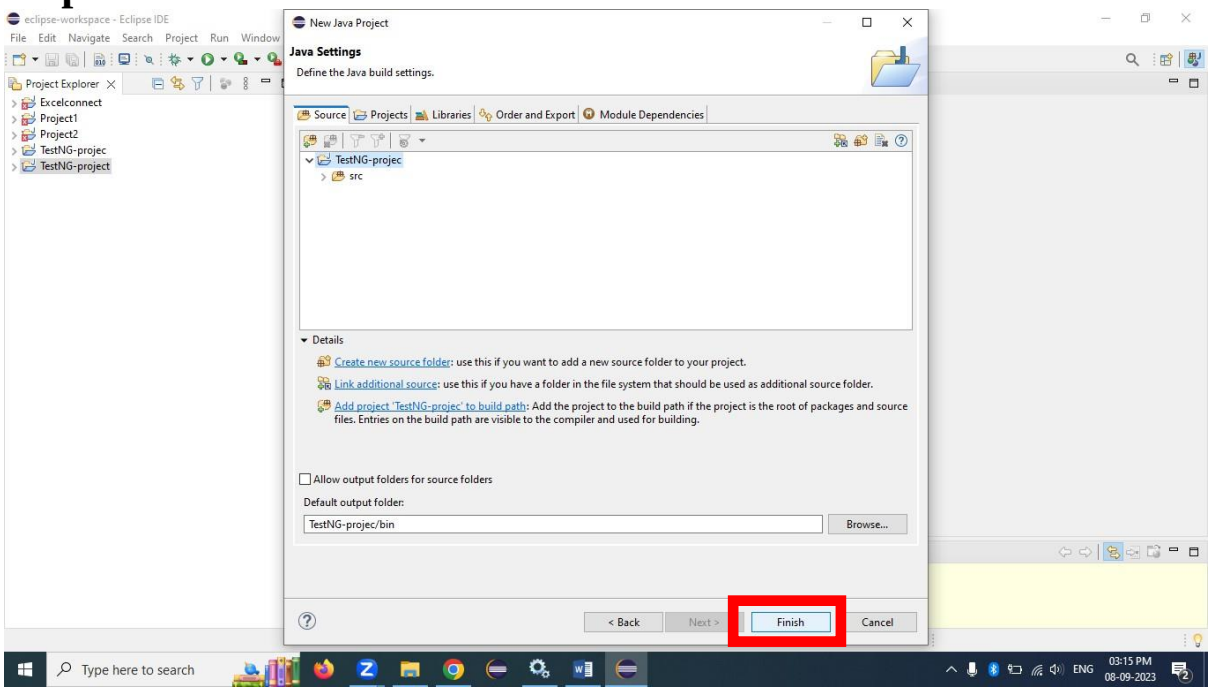
Step 2: You can name your project as your wish

1.I named as **TestNg-project**

2. Click **Next**



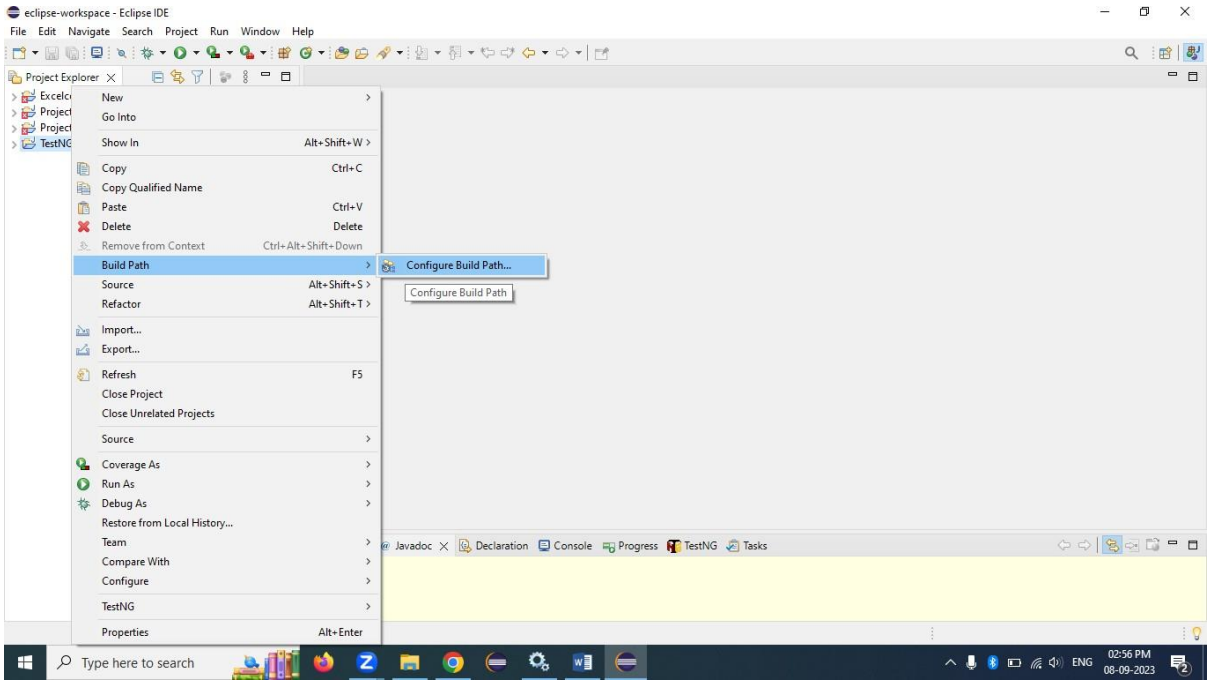
Step 3: Click **Finish**



How to configure Jar files insideTestNG?

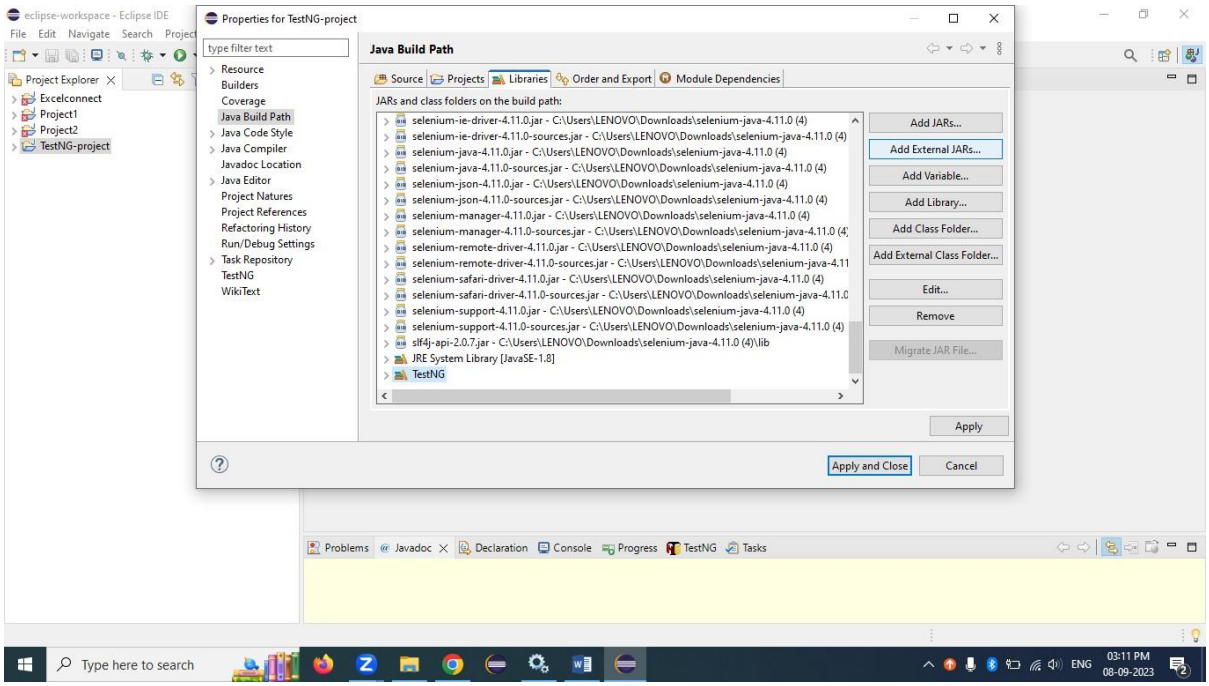
Step 1: Configure path

1. Right click on **TestNG-project** (Name of theproject)
2. Click **Build Path**
3. Click **Configure Build Path**



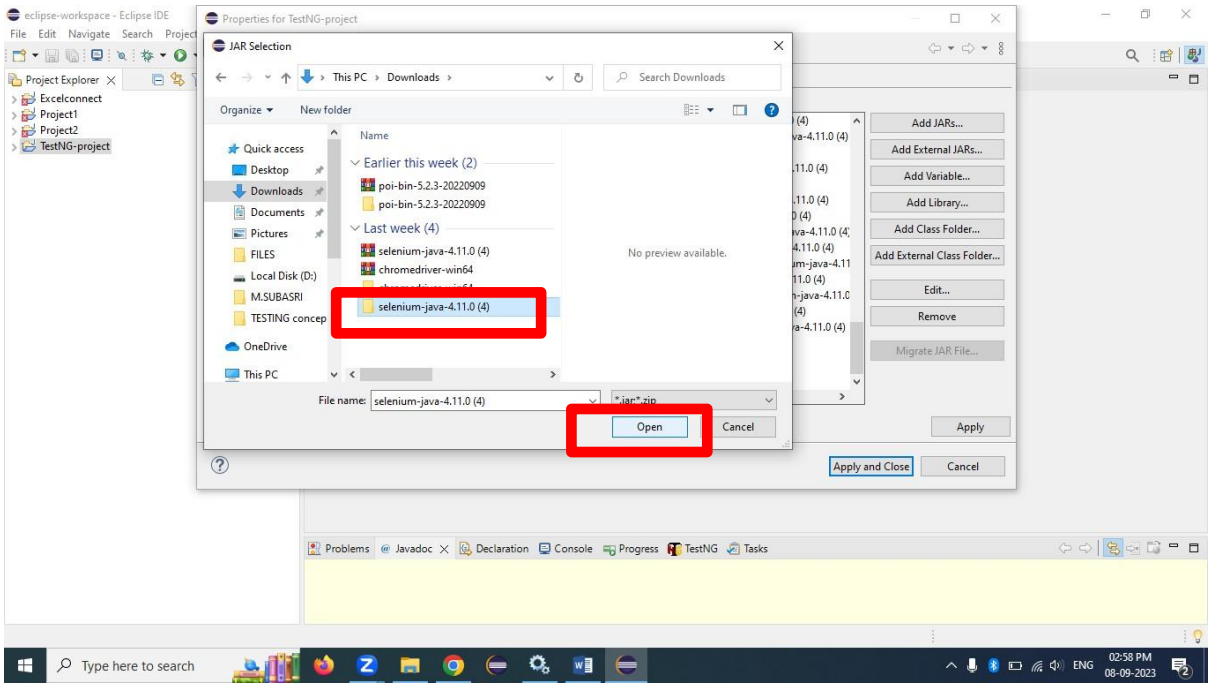
Step 2: Have to **configure Selenium insideTestNG**

1. Click **libraries**
2. Inside you will see **TestNG→Click** that
3. Click **Add External Jars**

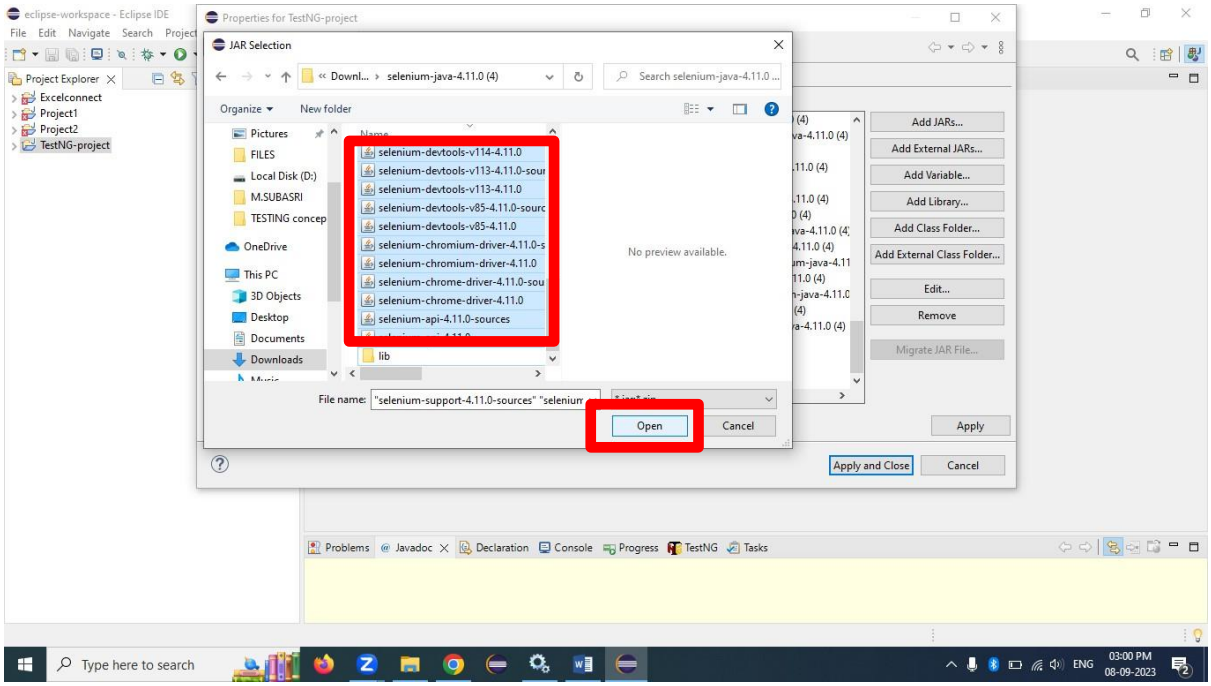


Step 3:The folder will open

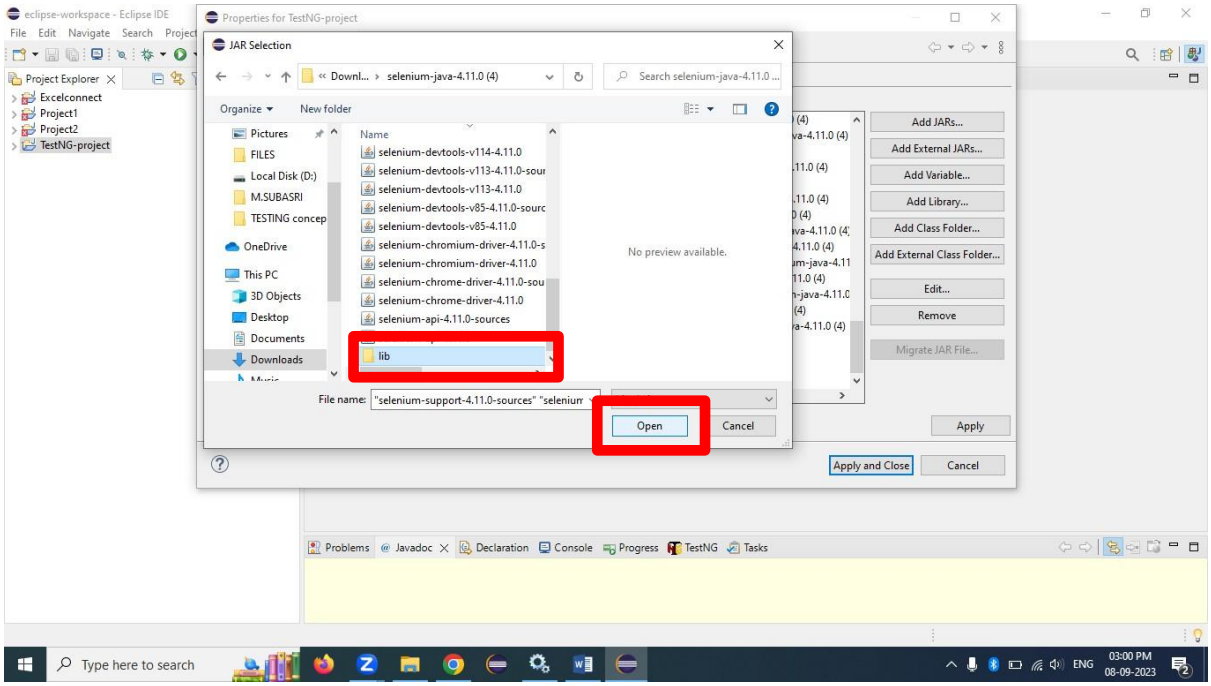
1. Click **Selenium folder** that you havedownloaded already.
2. If you not yet downloaded selenium, you candownload from here, https://www.selenium.dev/ (optional)
3. Click **open**



Step 4: Select all jar files **except lib folder→Open**



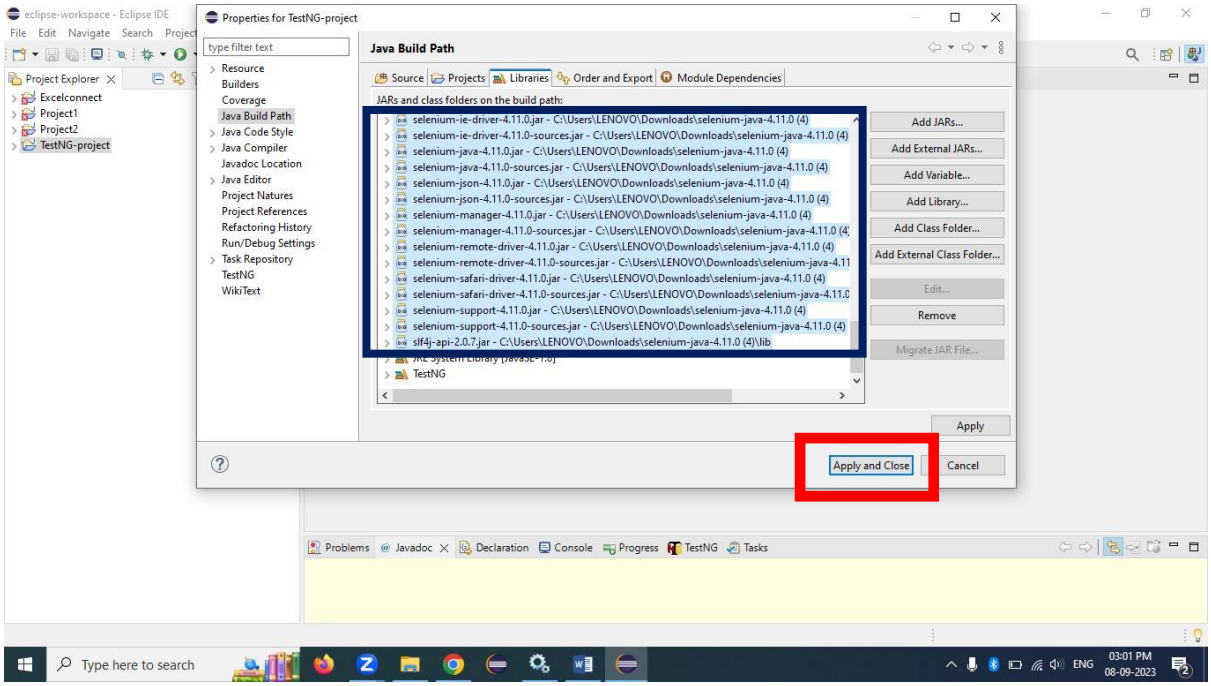
Step 5: Now **select lib folder→Open**



Step 6: All selenium jar files will be uploaded toTestNG.

Now all jar files will be configured, that isshown in blue color box in below image.

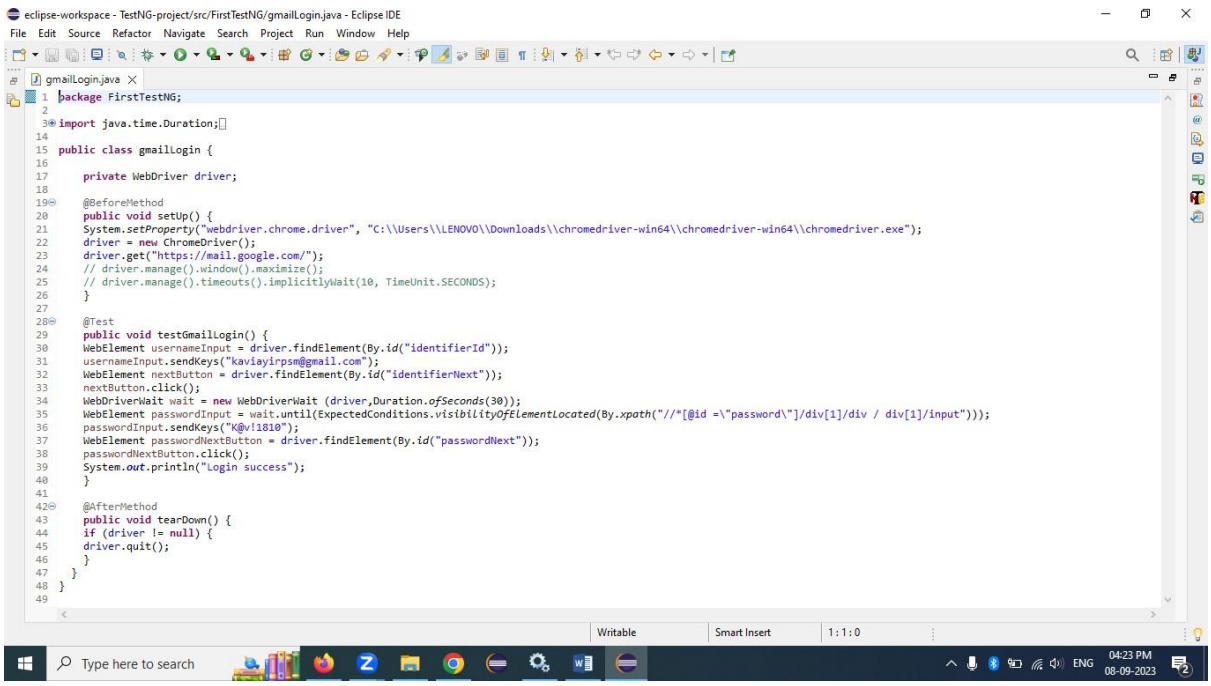
Click **Apply and Close**



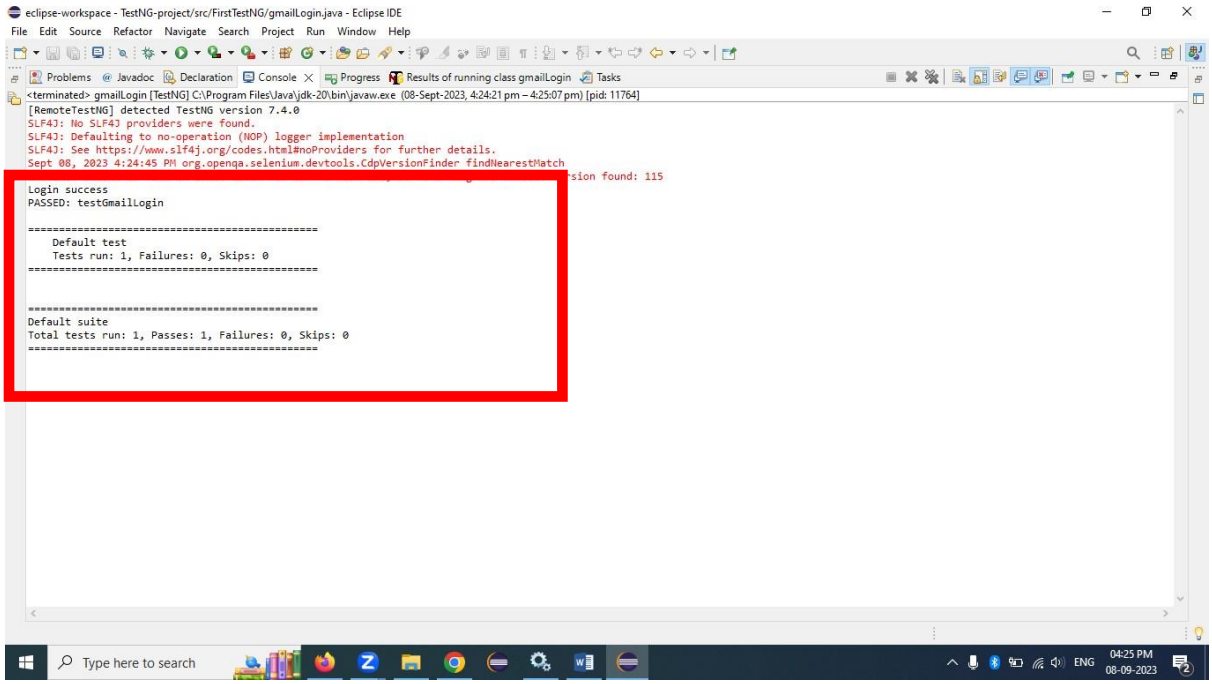
Sample code for G-mail account Login

```
package testNG;

import java.time.Duration;
import java.util.concurrent.TimeUnit;

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.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class gmail{
    private WebDriver driver;

    @BeforeClass
    public void setUp() {
      System.setProperty("webdriver.chrome.driver", "path to
chromeDriver");
        driver = new ChromeDriver();
        driver.get("https://mail.google.com/");
      //  driver.manage().window().maximize();
      //  driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
    }

    @Test
    public void testGmailLogin() {
        WebElement usernameInput =
driver.findElement(By.id("identifierId"));
        usernameInput.sendKeys("sample@gmail.com");
        WebElement nextButton =
driver.findElement(By.id("identifierNext"));
        nextButton.click();
        WebDriverWait wait = new WebDriverWait
```

```
(driver,Duration.ofSeconds(30));
        WebElement passwordInput =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//
*[@id =\"password\"]/div[1]/div / div[1]/input")));
        passwordInput.sendKeys("Pass");
        WebElement passwordNextButton =
driver.findElement(By.id("passwordNext"));
        passwordNextButton.click();
       System.out.println("Login success");
    }

    @AfterClass
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```



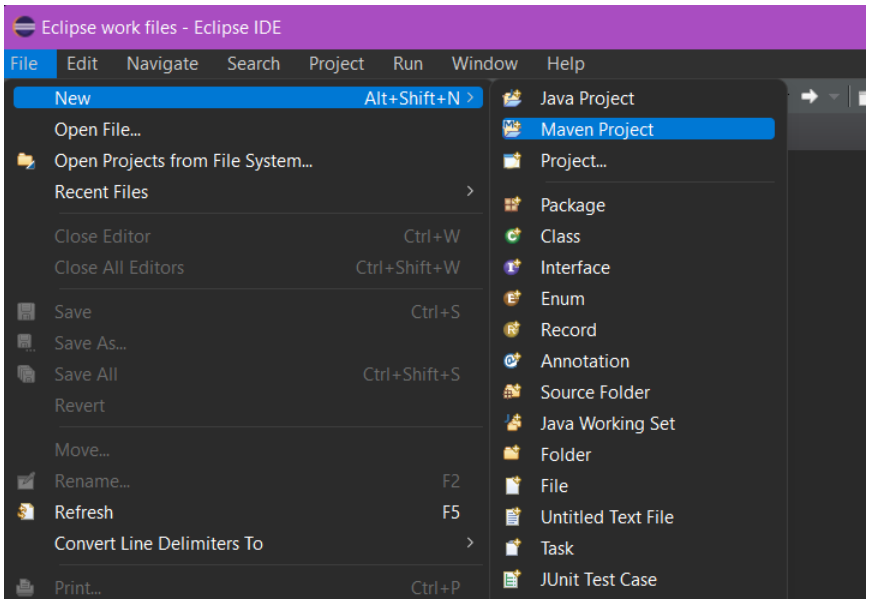
Output:


eclipse-workspace - TestNG-project/src/FirstTestNG/gmailLogin.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Progress Results of running class gmailLogin Tasks
<terminated> gmailLogin [TestNG] C:\Program Files\Java\jdk-20\bin\javaw.exe (08-Sept-2023, 4:24:21 pm – 4:25:07 pm) [pid: 11764]
[RemoteTestNG] detected TestNG version 7.4.0
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
Sept 08, 2023 4:24:45 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
sion found: 115
Login success
PASSED: testGmailLogin
===
Default test
Tests run: 1, Failures: 0, Skips: 0
===
===
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===
Type here to search
ENG
04:25 PM
08-09-2023

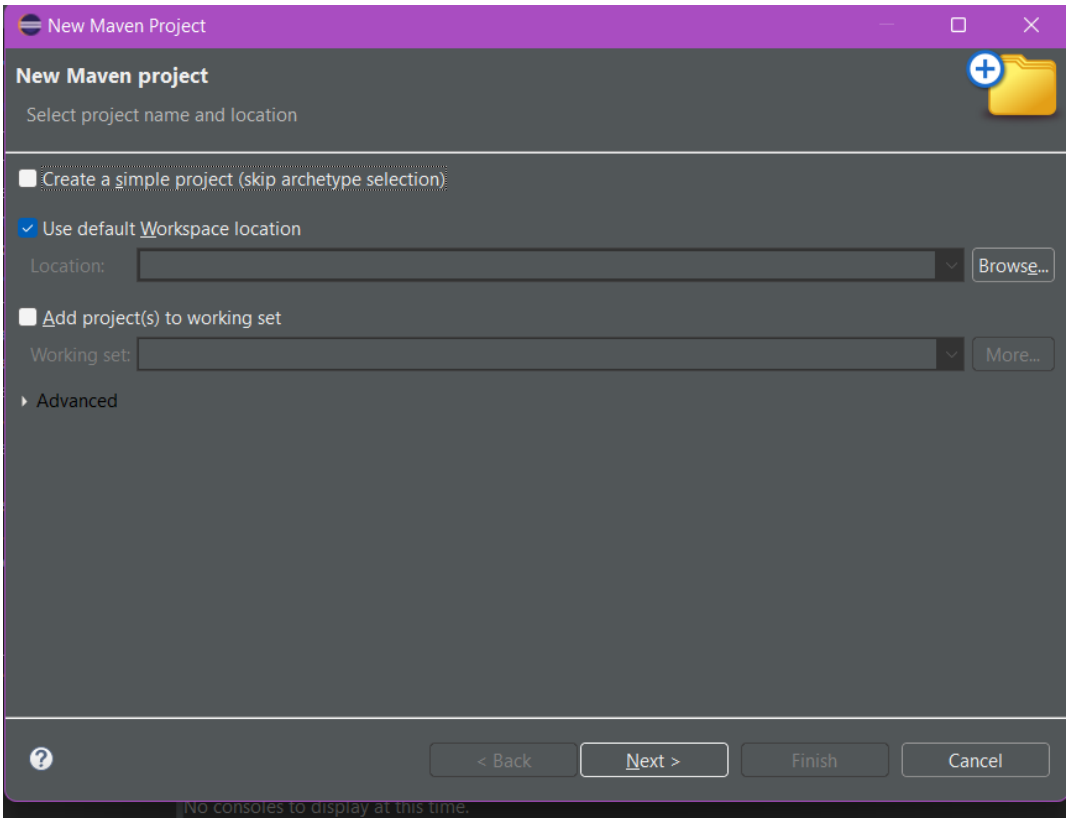
MAVEN

Create a New Maven Project:

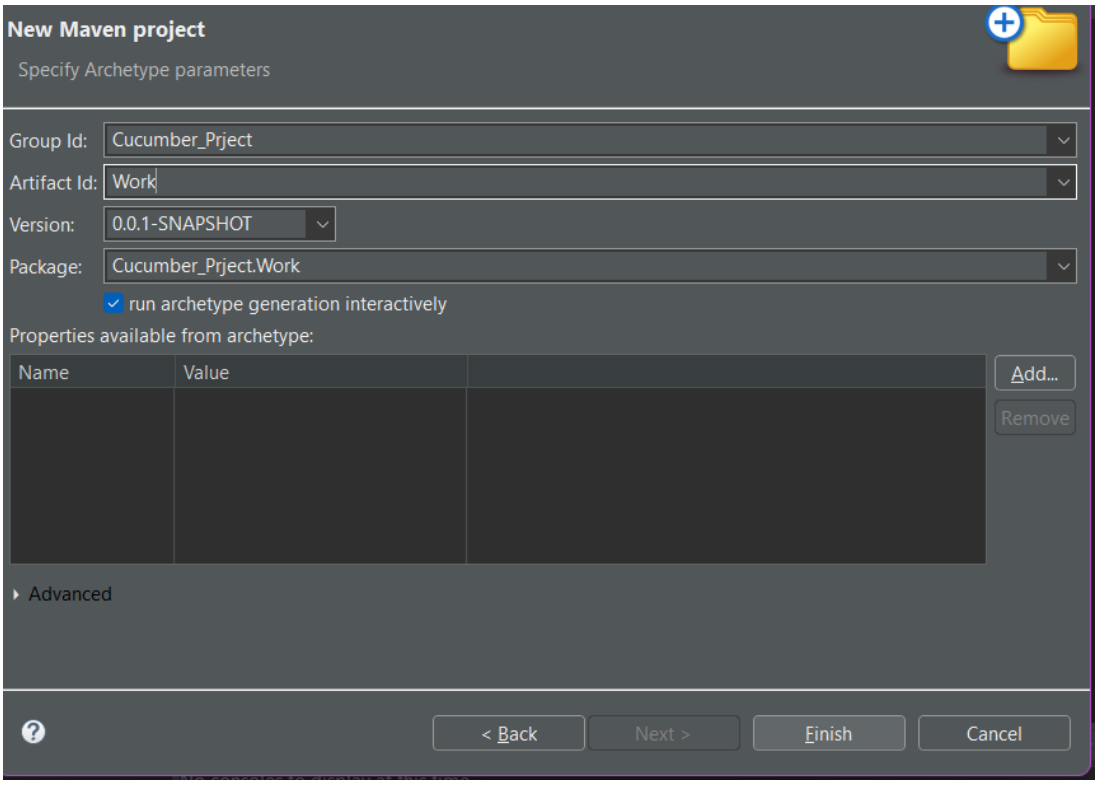
Open Eclipse and go to "File" -> "New" -> "Maven Project



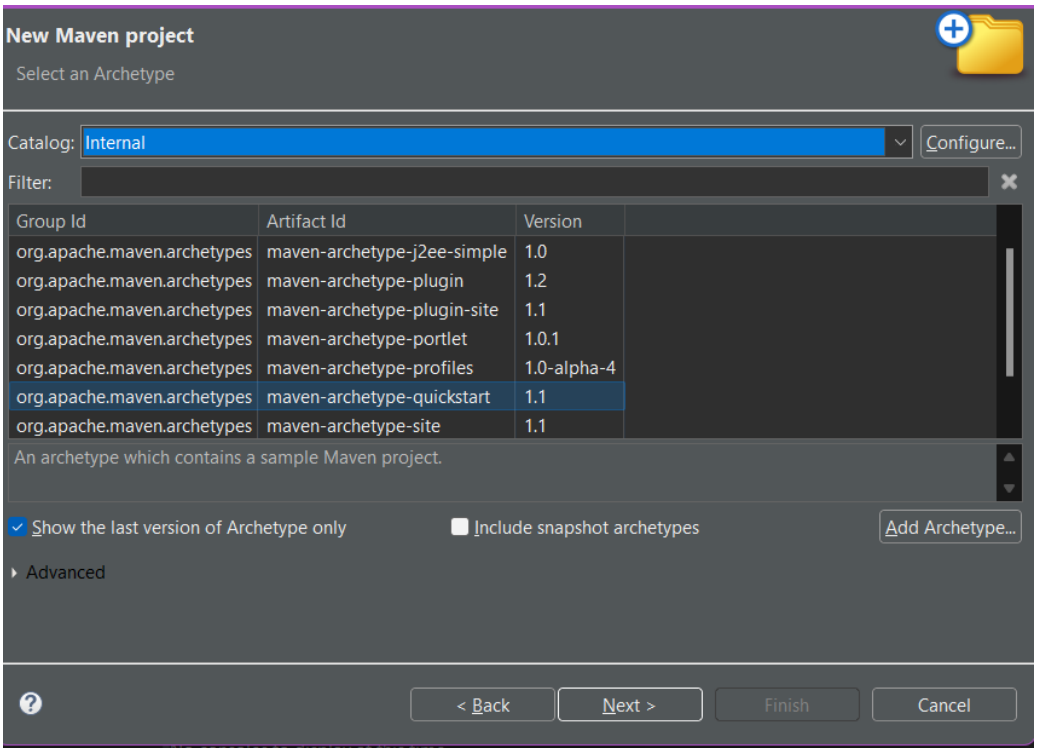
Click Next. (Use Default workspace or you can change the location)



Select "Catalog (Internal & choose ID maven- archetype-quickstart)" and click "Next."



Fill in the "Group Id" and "Artifact Id" for your project and click "Finish."



After Finish, it loading in the console and ***asking for Yes?***

```
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [org.apache.mave
[INFO] Using property: groupId = Cucumber_Prject
[INFO] Using property: artifactId = Work
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = Cucumber_Prject.Work
Confirm properties configuration:
groupId: Cucumber_Prject
artifactId: Work
version: 0.0.1-SNAPSHOT
package: Cucumber_Prject.Work
 Y: :
```

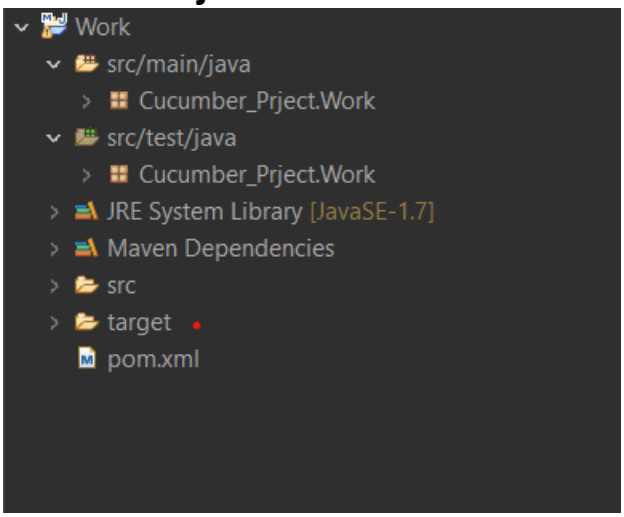
Type '**Y**' click enter

```
Problems  Javadoc  Declaration  Console ×  Progress  TestNG
C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (20-Sep-2023, 10:07:27 am) [pid: 10724]
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-a
[INFO] Using property: groupId = Cucumber_Prject
[INFO] Using property: artifactId = Work
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = Cucumber_Prject.Work
Confirm properties configuration:
groupId: Cucumber_Prject
artifactId: Work
version: 0.0.1-SNAPSHOT
package: Cucumber_Prject.Work
 Y: : Y
```

Project Build Successful

```
<terminated> C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (13-Sep-2023, 2:50:50 pm) [pid: 25668]
[INFO] Parameter: groupId, Value: Cucuwork
[INFO] Parameter: artifactId, Value: Cucumberwork
[INFO] Parameter: packageName, Value: Cucuwork.Cucumberwork
[INFO] Parameter: version, Value: 0.0.1-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\kavi1\OneDrive\Desktop\Eclipse work files\Cuc
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:40 min
[INFO] Finished at: 2023-09-13T14:52:34+05:30
[INFO] ------------------------------------------------------------------------
```

Maven Project File Structure

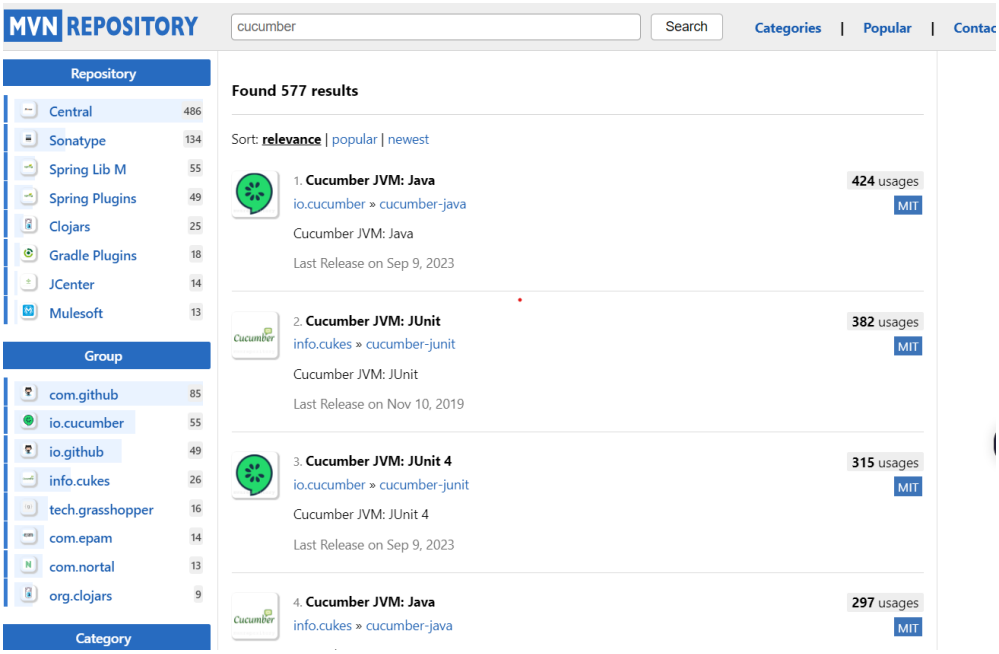


Cucumber Framework

MAVEN DEPENDENCY

Go to Maven Repository -> https://mvnrepository.com/

Search for cucumber framework



Click the dependency you need

Cucumber JVM: Java

Cucumber JVM: Java

License	MIT
Categories	Testing Frameworks & Tools
Tags	testing
Ranking	#1096 in MvnRepository (See Top Artifacts) #48 in Testing Frameworks & Tools
Used By	424 artifacts

Central (112)

	Version	Vulnerabilities	Repository	Usages	Date
7.14.x	7.14.0		Central	22	Sep 09, 2023
7.13.x	7.13.0		Central	37	Jul 07, 2023
7.12.x	7.12.1		Central	29	Jun 02, 2023
	7.12.0		Central	31	Apr 29, 2023
7.11.x	7.11.2		Central	37	Mar 23, 2023
	7.11.1		Central	36	Jan 31, 2023
	7.11.0		Central	38	Jan 12, 2023
7.10.x	7.10.1		Central	20	Dec 16, 2022
	7.10.0		Central	12	Dec 11, 2022
7.9.x	7.9.0		Central	30	Nov 01,

Click the Version based on the latest version or usages.
Copy the content and paste to the POM.xml

Cucumber JVM: Java » 7.13.0

Cucumber JVM: Java

License	MIT
Categories	Testing Frameworks & Tools
Tags	testing
Date	Jul 07, 2023
Files	pom (8 KB) jar (721 KB) View All
Repositories	Central
Ranking	**#1096 in MvnRepository (See Top Artifacts)** **#48 in Testing Frameworks & Tools**
Used By	**424 artifacts**

Note: There is a new version for this artifact

New Version	7.14.0

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-
java -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.13.0</version>
</dependency>
```

☑ Include comment with link to declaration

Copied to clipboard!

Use same procedure for other dependency like selenium, apache poi, testNG, etc.,

Add Cucumber Dependencies to the pom.xml:

Open the pom.xml file and add the Cucumber dependencies for Java and JUnit:

```
<dependencies>

    <!-- Selenium WebDriver -->
   <dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>selenium-java</artifactId>
   <version>2.47.1</version>
   </dependency>


    <!-- Cucumber Dependencies -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.0.0</version> <!-- Use the latest version available -->
</dependency>

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.0.0</version> <!-- Use the same version as cucumber-java -->
</dependency>


<!-- JUnit Dependency -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version> <!-- Use the latest version available -->
    <scope>test</scope>
</dependency>

        <dependency>
         <groupId>org.apache.poi</groupId>
         <artifactId>poi</artifactId>
         <version>5.0.0</version> <!-- Use the latest version -->
    </dependency>

       <!-- Apache POI for Excel -->
     <dependency>
         <groupId>org.apache.poi</groupId>
         <artifactId>poi-ooxml</artifactId>
         <version>3.17</version>
     </dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.8.0</version>
    <scope>test</scope>
</dependency>

  </dependencies>
</project>
```

Example

```
project-root
│
├── src
│   ├── main
│   │   └── java
│   │       └── Feature
│   │           └── fblogin.feature
│   └── test
│       └── java
│           ├── stepdefinitions
│           │   └── StepDefinitions.java
│           └── runners
│               └── TestRunner.java
└── pom.xml
```

- src/main/java/Feature/fblogin.feature: This is where your Cucumber feature file will reside, defining the behavior in a Gherkin syntax.

- src/test/java/stepdefinitions/StepDefinitions.java: This Java class will contain the step definitions that map Gherkin steps to Java code.

- src/test/java/runners/TestRunner.java: This Java class will act as the JUnit runner to execute your Cucumber tests.

- pom.xml: The Maven POM (Project Object Model) file containing project configuration and dependencies.

Fblogin.feature

```
Feature: demo

  Scenario: Login functionality exists

Given I have open the browser

When I open Facebook website

Then Login button should exits
```

StepDefinition.java

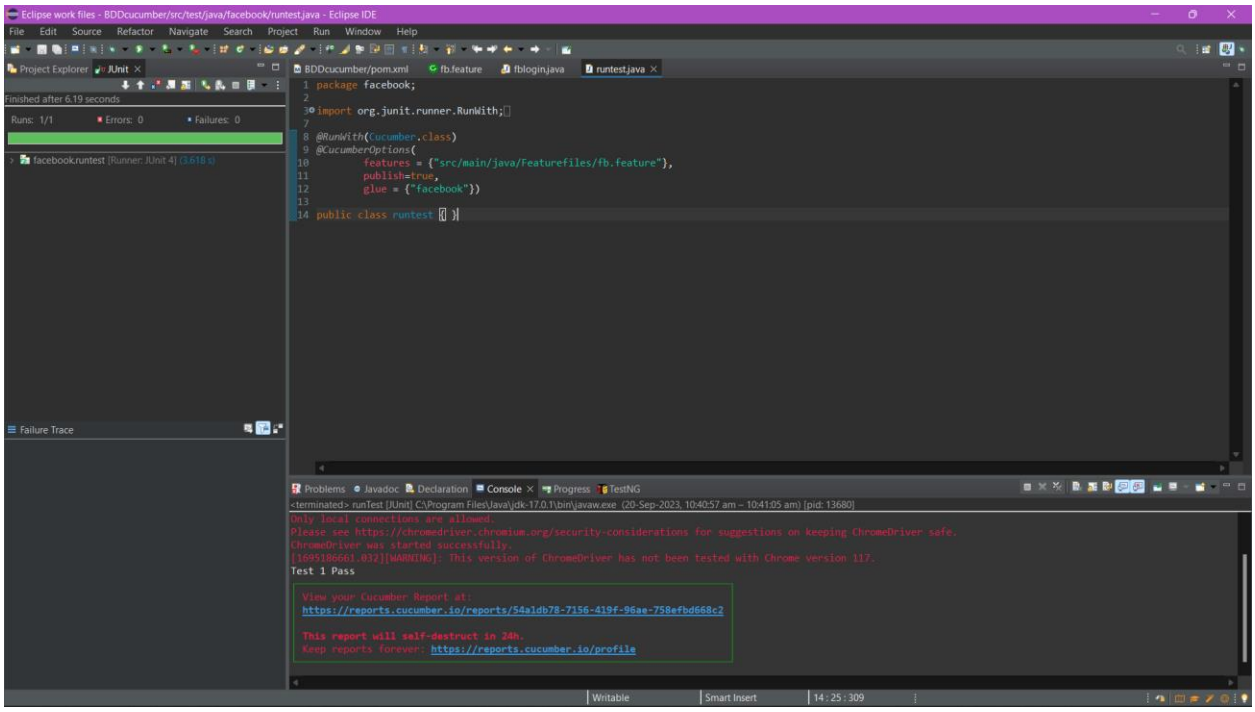
```
package facebook;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.WebElement;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
import io.cucumber.java.en.Then;

public class StepDefnition {
  WebDriver driver = null;
        @Given("^I have open the browser$")
  public void openBrowser() {
        System.setProperty("webdriver.chrome.driver", "path to chrome
Driver”);
    driver = new ChromeDriver();
  }
        @When("^I open Facebook website$")
  public void goToFacebook() {
    driver.navigate().to("https://www.facebook.com/");
  }
  @Then("^Login button should exits$")
  public void loginButton() {
    if(driver.findElement(By.name("login")).isEnabled()) {
    System.out.println("Test 1 Pass");
    } else {
      System.out.println("Test 1 Fail");
    }
    driver.close();
  }
}
```

TestRunner.java

```
import org.junit.runner.RunWith;
```

```
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
@RunWith(Cucumber.class)
@CucumberOptions(
		features = {"src/main/java/Featurefiles/fblogin.feature"},
		publish=true,
		glue = {" stepdefinitions "}) // StepDefinition package name
public class RunnerTest{ }
```

Output:



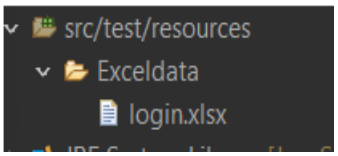
Links to refer

https://www.edureka.co/community/53904/login-test-for-gmail-with-cucumber-and-selenium-webdriver

https://www.numpyninja.com/post/how-to-read-data-from-excel-sheet-in-bdd-cucumber-framework

ExcelHandling Example

Create one folder in src/test/resources



Add excel sheet with data.

Feature File

```
Feature: Gmail Login

  Scenario: User Login with valid username and password from Excel
    Given I navigate to the Gmail login page
    When I enter Gmail username and password from Excel
    And I click the Gmail login button
    Then I should be logged into Gmail
```

StepDefinition

```
package Excelhandle;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
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 java.io.FileInputStream;
import java.io.IOException;
```

```
public class writeDataSteps {

    private WebDriver driver;
    private String username;
    private String password;

    @Given("I navigate to the Gmail login page")
    public void iNavigateToGmailLoginPage() {
        System.setProperty("webdriver.chrome.driver", "path to chrome");
        driver = new ChromeDriver();
        driver.get("https://mail.google.com/");
    }

    @When("I enter Gmail username and password from Excel")
    public void iEnterGmailUsernameAndPasswordFromExcel() {
        String excelFilePath = "src/test/resources/Exceldata/login.xlsx";
        int sheetIndex = 0; // Assuming data is in the first sheet

        try (FileInputStream fis = new FileInputStream(excelFilePath);
             Workbook workbook = new XSSFWorkbook(fis)) {

            Sheet sheet = workbook.getSheetAt(sheetIndex);
            Row row = sheet.getRow(1); // Assuming data is in the first row

            username = row.getCell(0).getStringCellValue();
            password = row.getCell(1).getStringCellValue();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @When("I click the Gmail login button")
    public void iClickGmailLoginButton() {
        WebElement emailField = driver.findElement(By.id("identifierId"));
        emailField.sendKeys(username);
```

```
        WebElement nextButton = driver.findElement(By.id("identifierNext"));
        nextButton.click();
    }

    @Then("I should be logged into Gmail")
    public void iShouldBeLoggedInToGmail() {
        System.out.println("Move to password field");
        WebDriverWait wait = new WebDriverWait(driver, 30);
        WebElement passwordInput = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id =\"password\"]/div[1]/div / div[1]/input")));
        WebElement passwordField = driver.findElement(By.xpath("//*[@id =\"password\"]/div[1]/div / div[1]/input"));
        passwordField.sendKeys(password);

        WebElement nextButton = driver.findElement(By.id("passwordNext"));
        nextButton.click();
    }
}
```

RunnerClass

```
package Excelhandle;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/main/java/Featurefiles/write_data.feature",
    glue = {"Excelhandle"},
    publish=true,
    plugin = {"pretty", "html:target/cucumber-reports"}
)
```

```
public class writeDataRunner {
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×  Progress  Results of running class draganddrop
<terminated> writeDataRunner (1) [JUnit] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (20-Sep-2023, 1:12:49 pm – 1:13:05 pm) [pid: 22968]

Scenario: User Login with valid username and password from Excel # src/main/java/Featurefiles/write_data.feature:3
Starting ChromeDriver 117.0.5938.62 (25a7172909a4cba7355365cf424d7d7eb35231f4-refs/branch-heads/5938@{#1146}) on port 28480
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
  Given I navigate to the Gmail login page                    # Excelhandle.writeDataSteps.iNavigateToGmailLoginPage()
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the consol
  When I enter Gmail username and password from Excel         # Excelhandle.writeDataSteps.iEnterGmailUsernameAndPasswordFromExcel()
  And I click the Gmail login button                          # Excelhandle.writeDataSteps.iClickGmailLoginButton()
Move to password field
  Then I should be logged into Gmail                          # Excelhandle.writeDataSteps.iShouldBeLoggedInToGmail()

  View your Cucumber Report at:
  https://reports.cucumber.io/reports/338bb03e-7733-42b4-9e76-c49007d32ee7

  This report will self-destruct in 24h.
  Keep reports forever: https://reports.cucumber.io/profile
```