

## Experiment 08 - Testing using Selenium

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements  LO4: Understand the importance of Selenium and Jenkins to test Software Applications

**Aim:** To Setup and Run Selenium Tests in Jenkins Using Maven.

## **Introduction:**

### **Selenium**

Selenium is a portable framework for testing web applications. Selenium is an open-source, test automation tool that has become an important automation tool in the software quality assurance world. This selenium testing tool consists of a different set of tools which include Selenium WebDriver, Selenium RC, Selenium IDE, and Selenium Grid, all of which have different features.

Selenium testing tool is a lightweight tool and is developer-friendly, commonly used for automating web applications. Test automation using selenium webdriver with java, automation testing can be used in any operating system environment such as Windows, Linux, and OS X and has been first developed by Jason Huggins in the year 2004. Cross-browser testing in selenium is an effective selenium testing method used by testers and this tool is also used for web application testing. This selenium testing tool is composed of several components that provide different features.

### **Components of Selenium**

There are various components of the selenium tool which provide different features that support multiple browsers, deliver parallel test capabilities and can be executed on multiple machines.

It has a suite of tools that cater to different needs of businesses and has the following components:

#### **1. Selenium Remote Control (RC)**

Selenium RC supports all major web browsers and this tool interprets commands to convert them into javascript and further these scripts are injected into the browser.

#### **2. Selenium IDE (Selenium Integrated Development Environment)**

Selenium IDE is a simple record and playback kind of tool which is available as an add-on for Firefox only.

#### **3. Selenium WebDriver**

Selenium WebDriver is one of the vital tools of the Selenium suite and it does not require any manual process like Selenium server and has direct communication between code and browser.

#### **4. Selenium Grid**

The Selenium grid is the last component of the selenium suite and is used for parallel testing and sometimes even supports distributive testing.

## **Common uses of selenium**

Selenium testing tool is commonly used to automate the testing across various web browsers. Cross-browser testing in selenium is most important as it supports various browsers such as Chrome, Mozilla, Firefox, Safari, and IE. Selenium tool can be easily used to automate browser testing across these browsers using Selenium WebDriver.

Selenium testing tool is best suited with an automated testing selenium suite for all web applications across browsers. It delivers good results for all tests as it is an integral part of automation testing.

The complete suite of Selenium software is mainly used to automate web browsers effectively. Selenium UI testing is another testing method that can be done with this tool.

Selenium testing tool is available free of charge and is used for frequent regression testing. Selenium testing enables rapid feedback to developers, as bugs are identified quickly and efficiently. This tool typically supports unlimited iterations of test case execution.

Selenium testing framework supports agile and extreme development methodologies and enables customized defect reporting. Selenium automation testing is used for finding defects that have been undetected by manual testing.

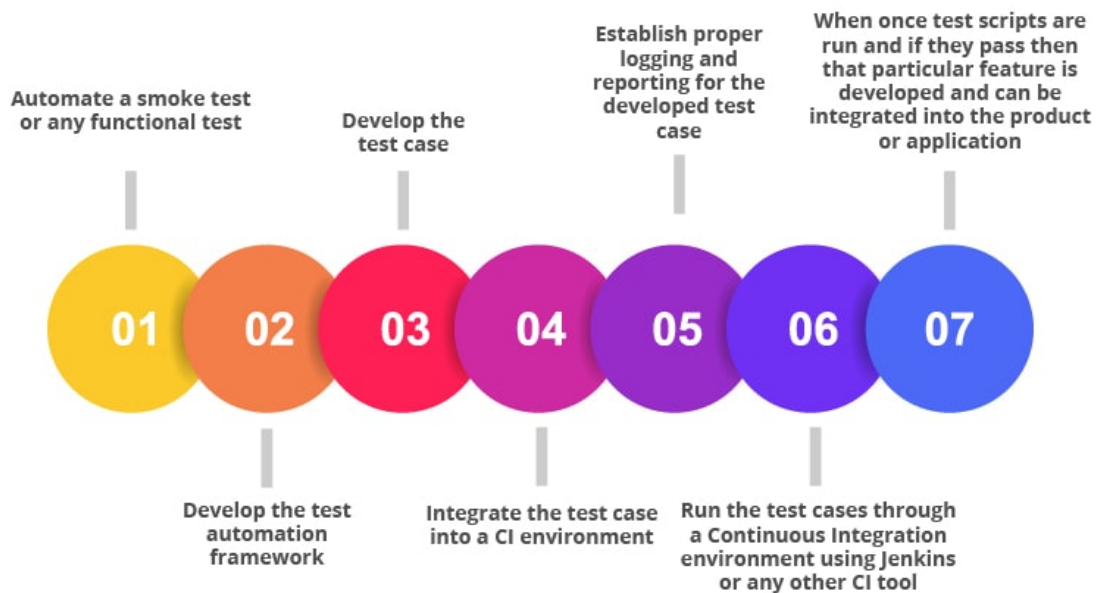
Selenium tool also supports database testing using SQL commands and can be used to conduct database testing.

## **Benefits of test automation**

There are many advantages businesses achieve by leveraging Test automation, and some of them are:

- Helps to find bugs at an early stage
- Testing becomes easy with pre-recorded and predefined actions
- It becomes simple and easy to compare test results to know the expected behaviour
- Testing can be done from any device and at any time even from remote locations
- Automated tests are reusable and are more accurate than manual test outcomes
- Testers can reuse the code on different versions of the software
- Shortens overall testing time when compared to manual testing
- Lowers overall testing costs
- Ensures faster time to market
- Generates quicker ROI

## Steps for Selenium testing



- Automate a smoke test or any functional test
- Develop the test automation framework
- Develop the test case
- Integrate the test case into a CI environment
- Establish proper logging and reporting for the developed test case
- Run the test cases through a Continuous Integration environment using Jenkins or any other CI tool
- When once test scripts are run and if they pass then that particular feature is developed and can be integrated into the product or application

## Configure Selenium in Jenkins:

### Steps to install TestNG and Maven on Eclipse and use it with Selenium

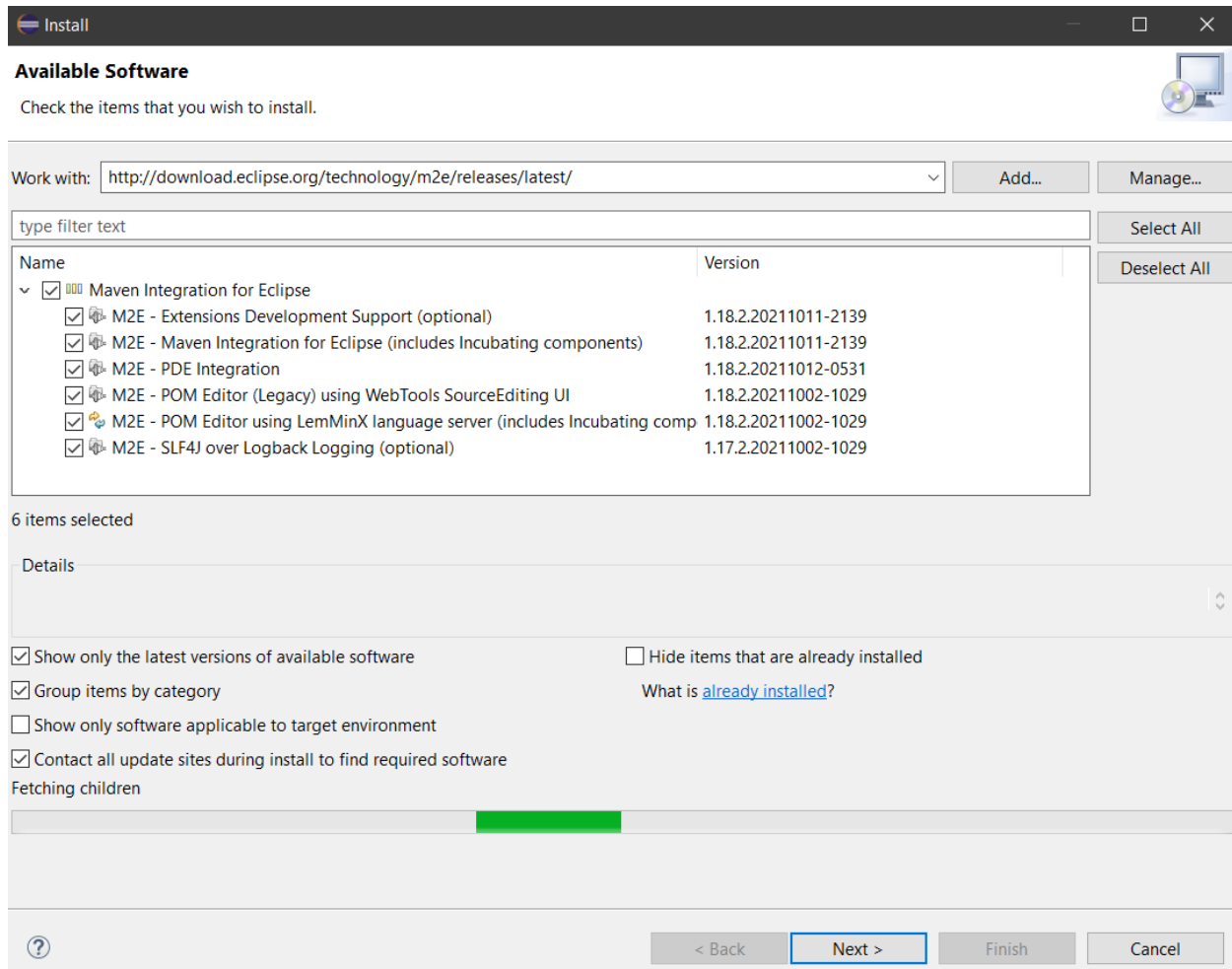
For this tutorial, we will use Eclipse (Juno) IDE for Java Developers to set up the Selenium WebDriver Project. Additionally, we need to add the m2eclipse plugin to Eclipse to facilitate the build process and create a pom.xml file.

Let's add the m2eclipse plugin to Eclipse with the following steps:

Step1) In Eclipse IDE, select Help -> Install New Software from Eclipse Main Menu.

Step 2) On the Install dialog, Enter the URL

<http://download.eclipse.org/technology/m2e/releases/>. Select Work with and m2e plugin as shown in the following screenshot:



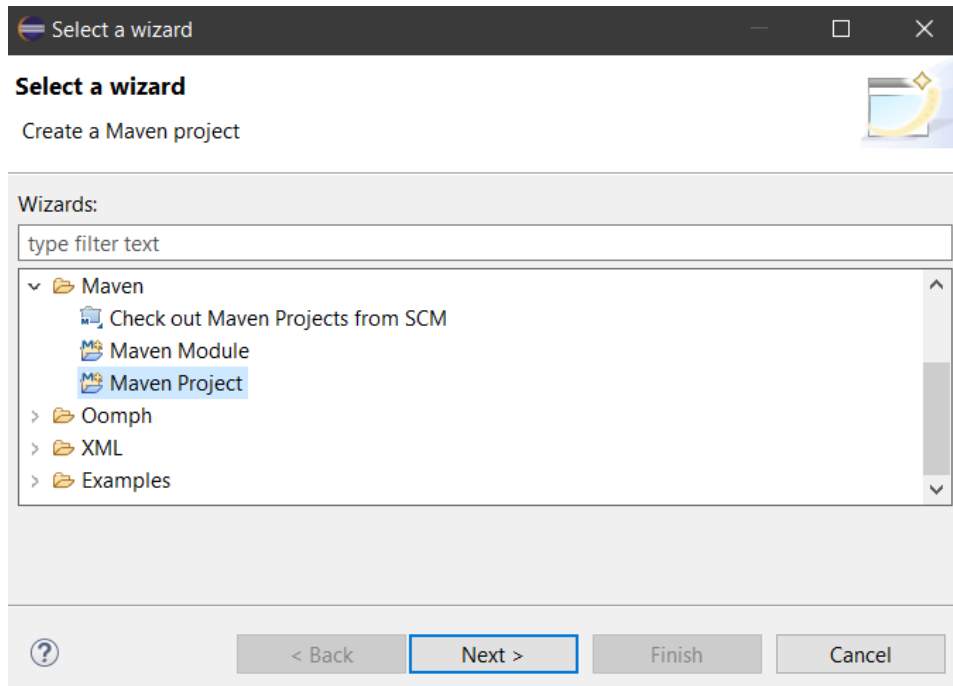
Step 3) Click on the Next button and finish the installation.

## Configure Eclipse with Maven

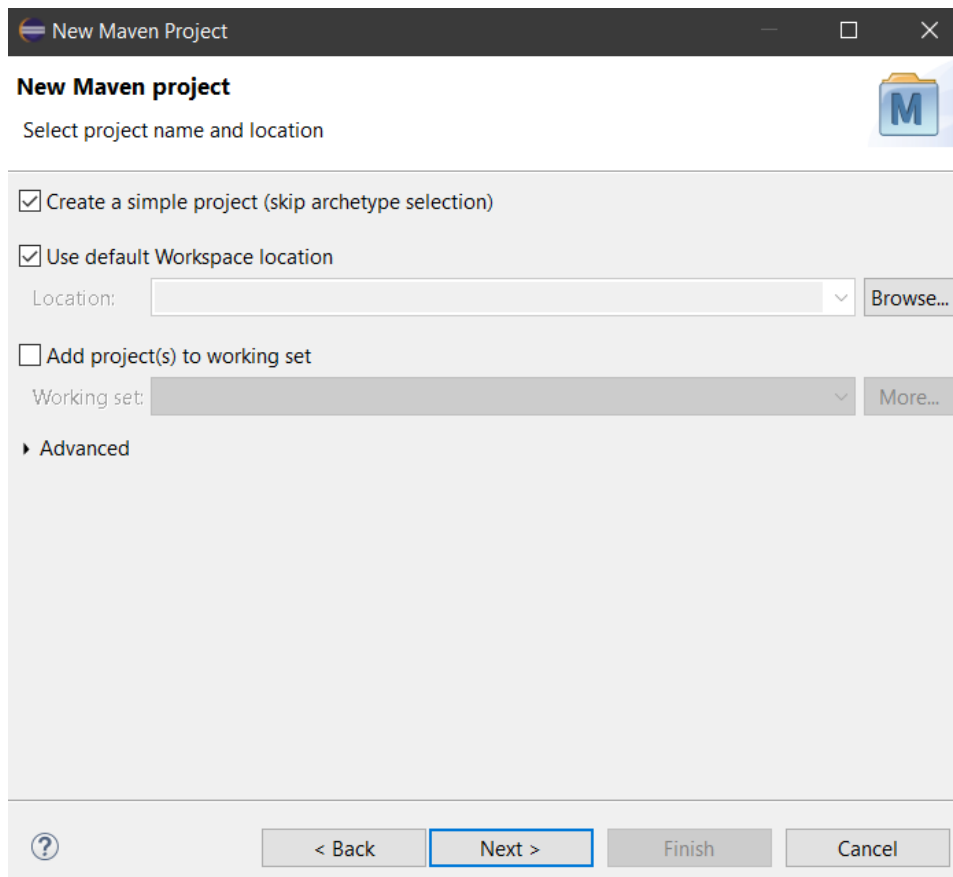
With the m2e plugin installed, we now need to create a Maven project.

Step 1) In Eclipse IDE, create a new project by selecting File | New | Other from the Eclipse menu.

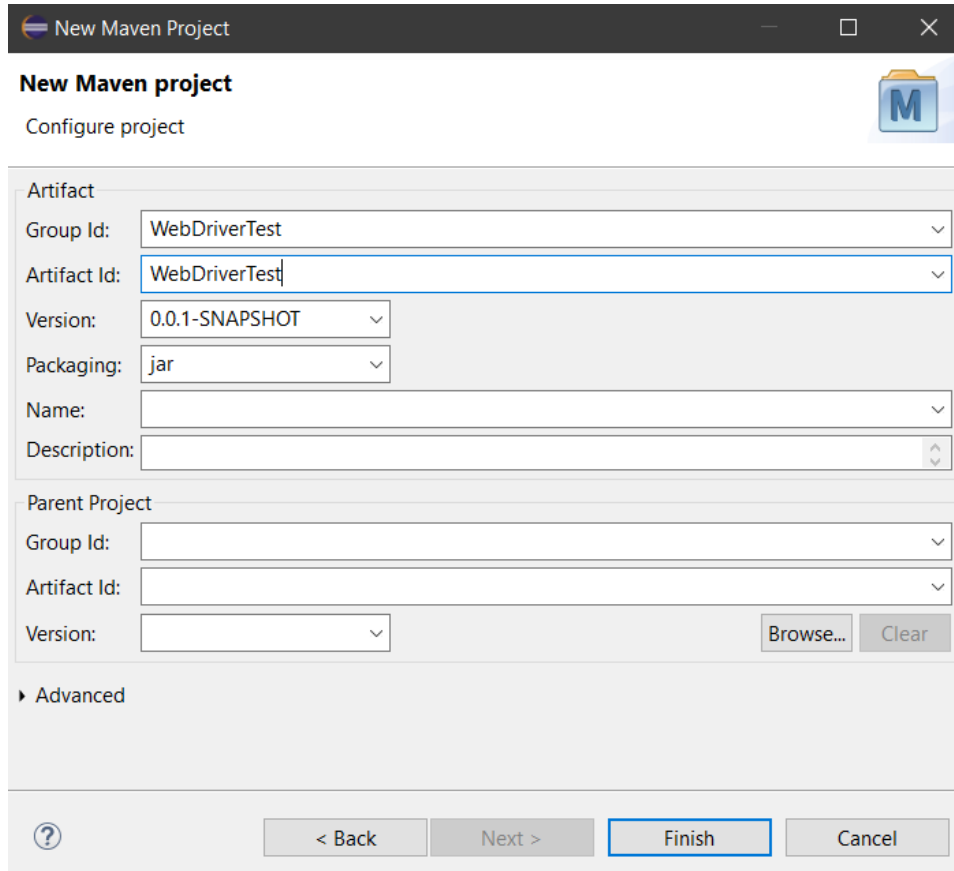
Step 2) On the New dialog, select Maven | Maven Project and click Next



Step 3) On the New Maven Project dialog select the Create a simple project and click Next



Step 4) Enter WebDriverTest in Group Id: and Artifact Id: and click finish



**New Maven Project**

Configure project

**Artifact**

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

**Parent Project**

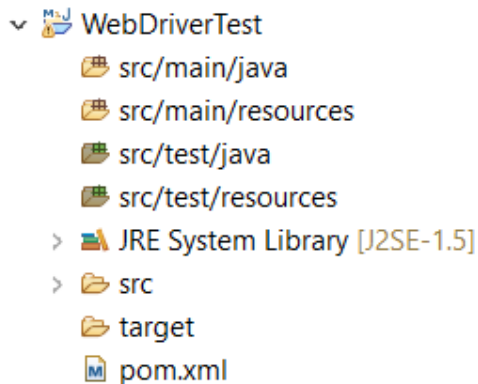
Group Id:

Artifact Id:

Version:

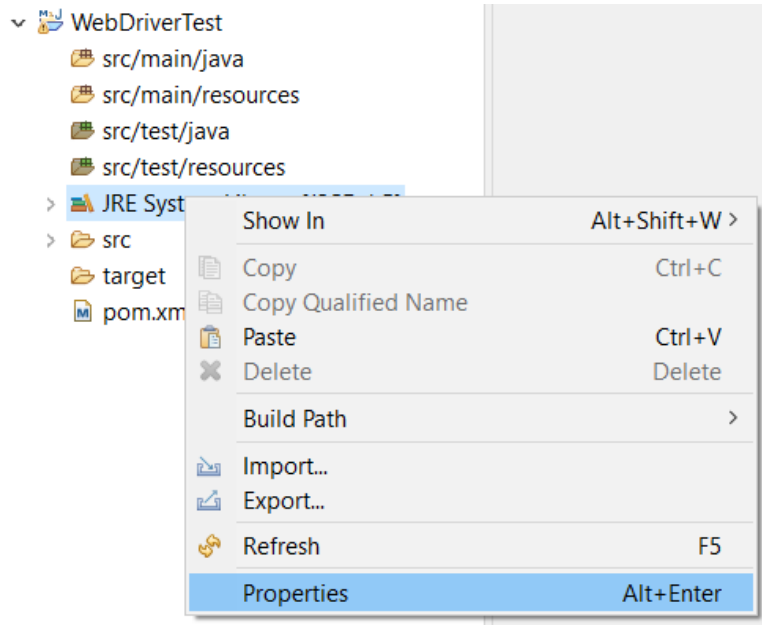
▶ **Advanced**

Step 5) Eclipse will create WebDriverTest with following structure:

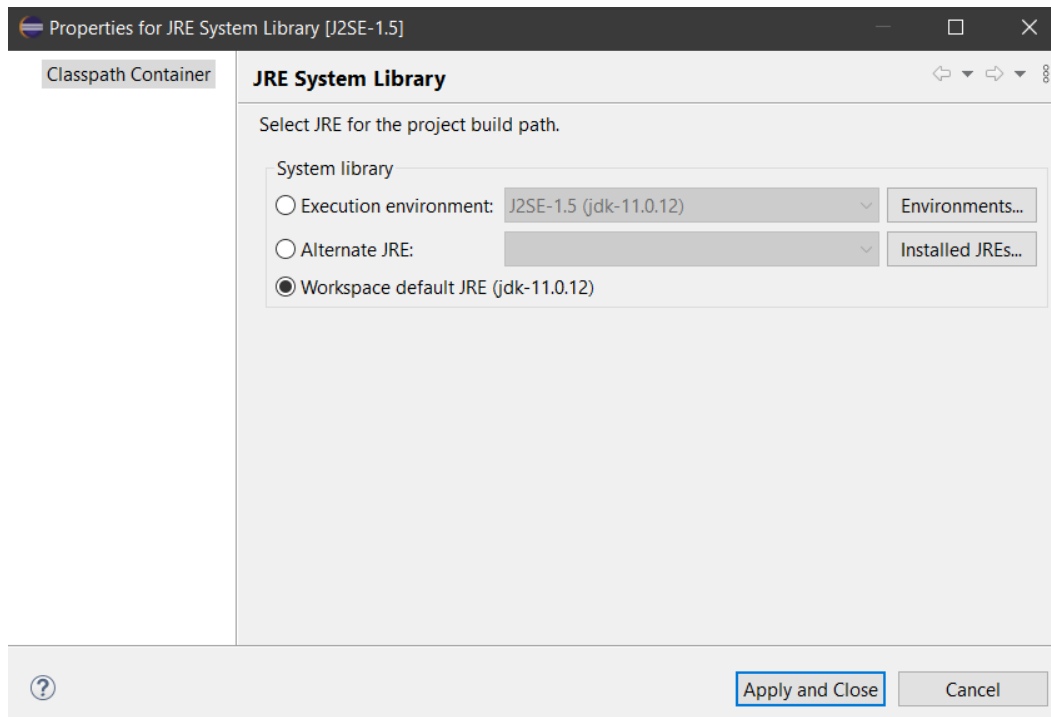


- ▼ **WebDriverTest**
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/resources
  - > JRE System Library [J2SE-1.5]
  - > src
  - target
  - pom.xml

Step 6) Right-click on JRE System Library and select the Properties option from the menu.

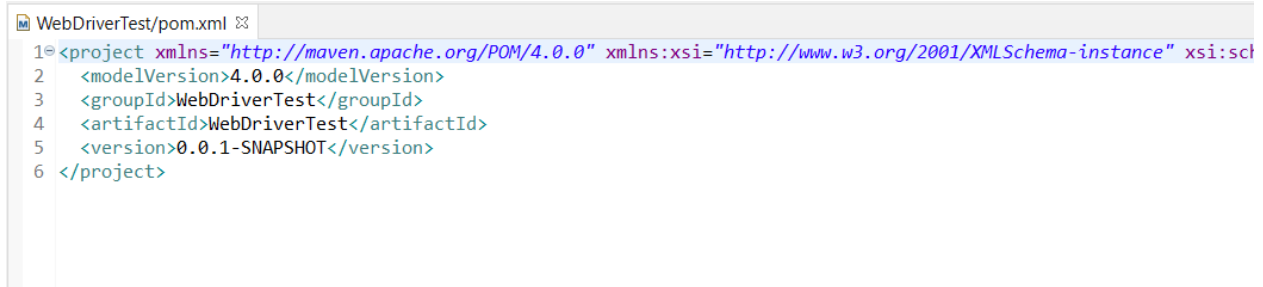


On the Properties for JRE System Library dialog box, make sure Workspace default JRE is selected and click OK



Step 7) Select pom.xml from Project Explorer. pom.xml file will Open in Editor section

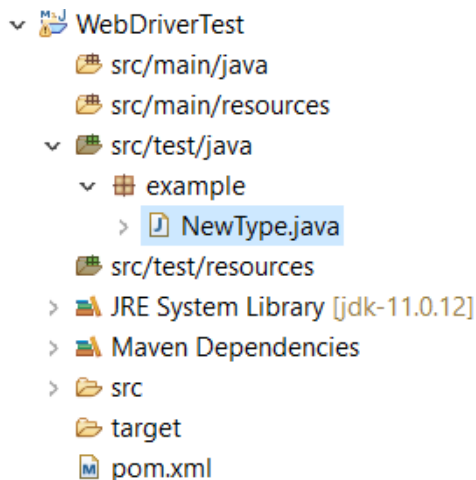




Step 8) Add the Selenium, Maven, TestNG, Junit dependencies to pom.xml in the <project> node:

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.12.0</version>
  </dependency>
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.4.0</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Step 9) Create a New TestNG Class. Enter Package name as “example” and “NewTest” in the Name: textbox and click on the Finish button. Eclipse will create the NewTest class as shown.



Step 10) Add the following code to the NewTest class:

This code will verify the title of Selenium Page

package example;

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
public class NewTest {

    private WebDriver driver;

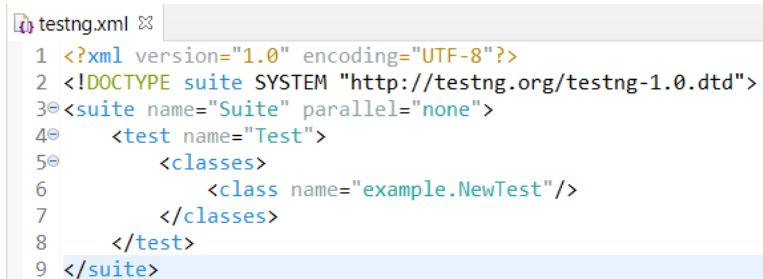
    @Test
    public void testEasy() {
        driver.get("http://demo.guru99.com/test/guru99home/");
        String title = driver.getTitle();
        Assert.assertTrue(title.contains("Demo Guru99 Page"));
    }

    @BeforeTest
    public void beforeTest() {
        System.setProperty("webdriver.gecko.driver",
"PATH\\T0\\geckodriver.exe");
        driver = new FirefoxDriver();
    }

    @AfterTest
    public void afterTest() {
        driver.quit();
    }
}
```

Step 11) Right-click on the WebdriverTest and select TestNG | Convert to TestNG.

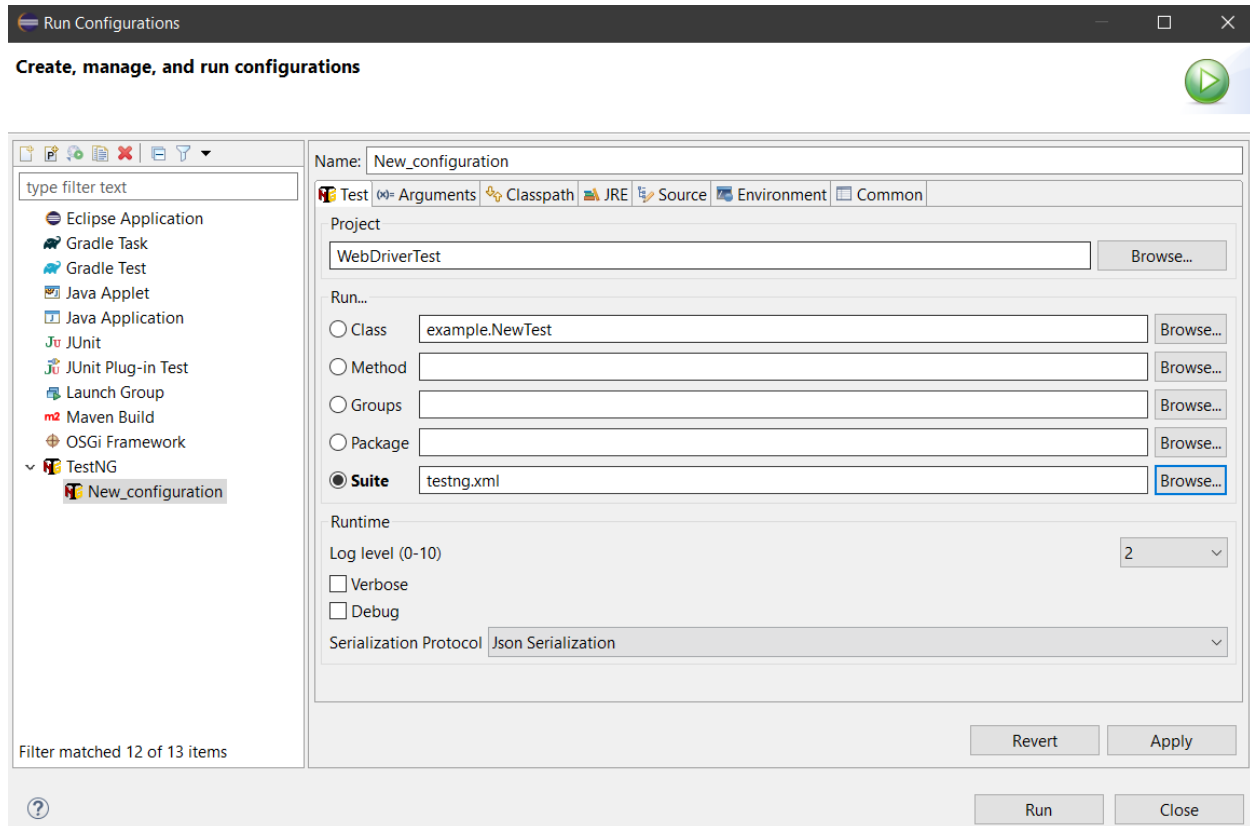
Eclipse will create testng.xml which says that you need to run only one test with the name NewTest.

The image shows a screenshot of the Eclipse IDE with a file named 'testng.xml' open. The XML content is as follows:

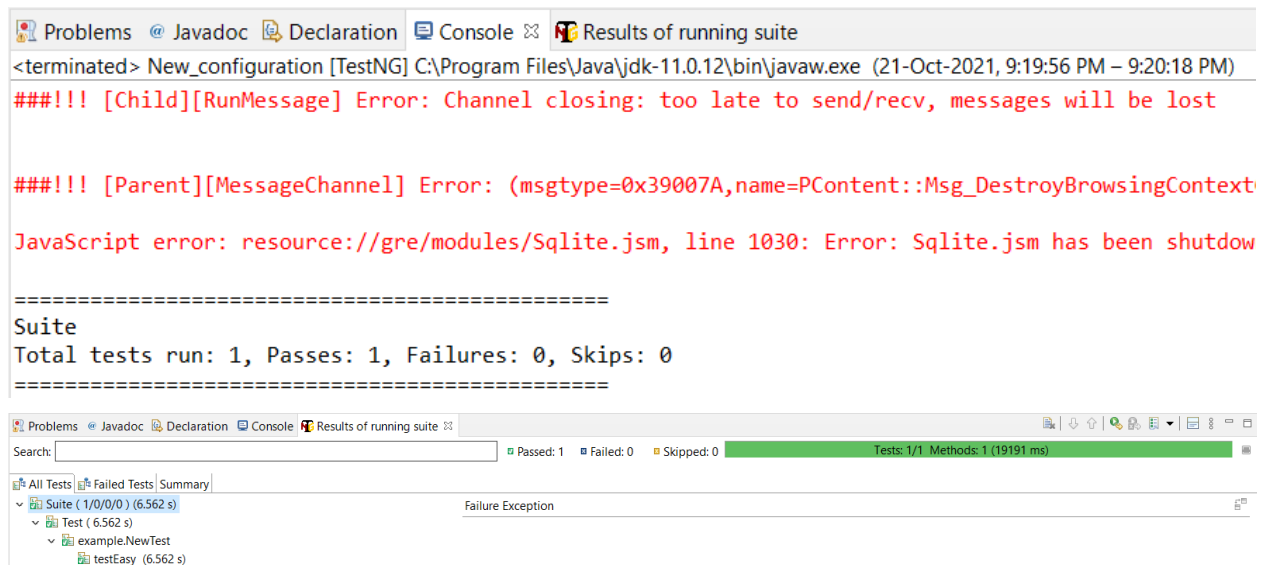
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite" parallel="none">
4   <test name="Test">
5     <classes>
6       <class name="example.NewTest"/>
7     </classes>
8   </test>
9 </suite>
```

Step 12) Now you need to run the test through this testng.xml.

So, go to the Run Configurations and create a new launch TestNG, select the project and field Suite as testng.xml and click Run.



Make sure that the build is finished successfully.



Step 14) Additionally, we need to add

1. maven-compiler-plugin
2. maven-surefire-plugin

3. testng.xml  
to pom.xml.

The maven-surefire-plugin is used to configure and execute tests. Here plugin is used to configure the testing.xml for TestNG test and generate test reports.

The maven-compiler-plugin is used to help in compiling the code and using the particular JDK version for compilation. Add all dependencies in the following code snippet, to pom.xml in the

<plugin> node:

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.12</version>

<inherited>>true</inherited>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

<pluginManagement>

<plugins>

<!--This plugin's configuration is used to store Eclipse m2e settings only. It has no influence on the Maven build itself.-->

<plugin>

<groupId>org.eclipse.m2e</groupId>

<artifactId>lifecycle-mapping</artifactId>

<version>1.0.0</version>

<configuration>

<lifecycleMappingMetadata>

<pluginExecutions>

<pluginExecution>

<pluginExecutionFilter>

<groupId>

```

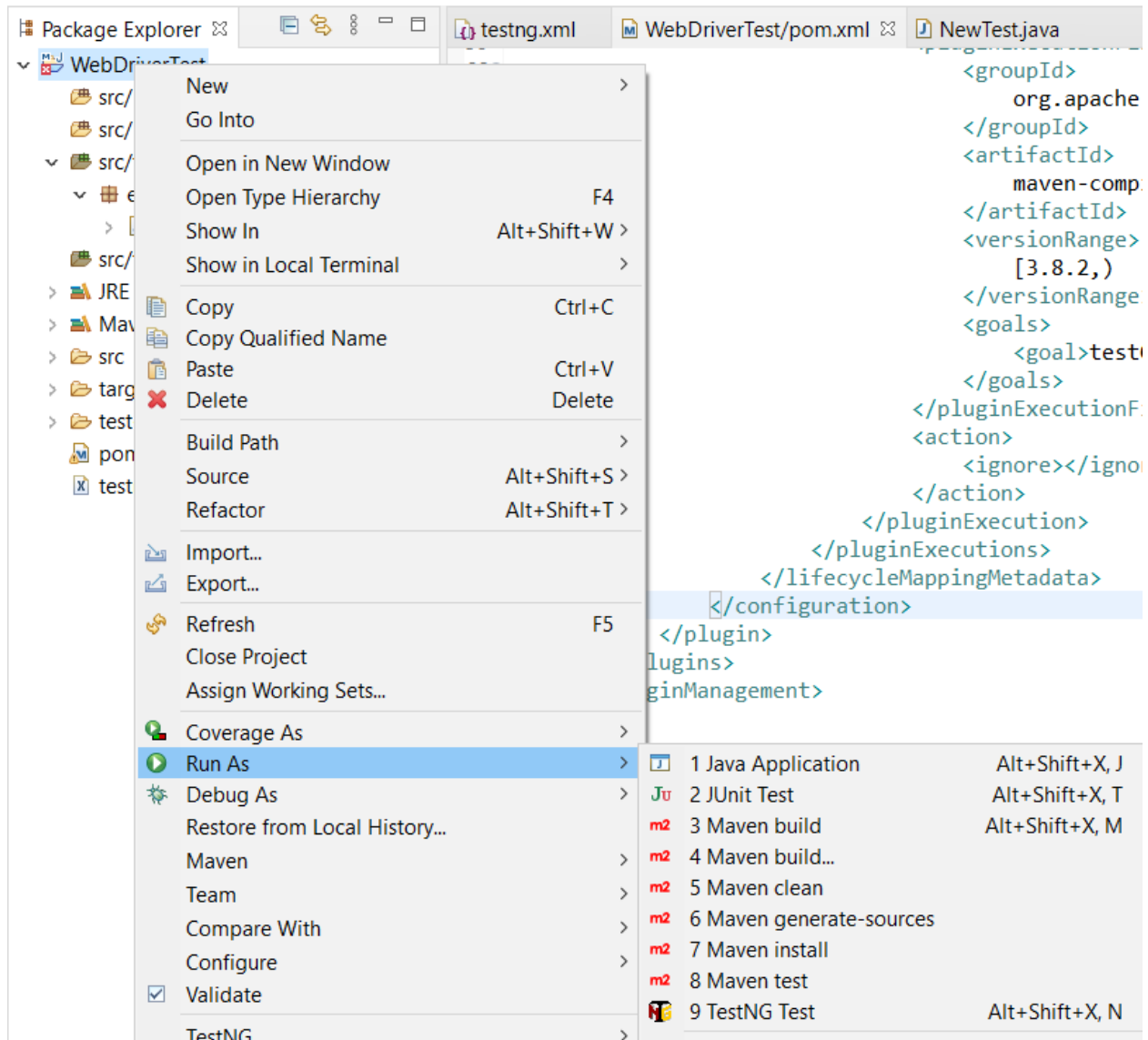
org.apache.maven.plugins
                                </groupId>
                                <artifactId>

maven-compiler-plugin
                                </artifactId>
                                <versionRange>
                                    [3.8.2,)
                                </versionRange>
                                <goals>

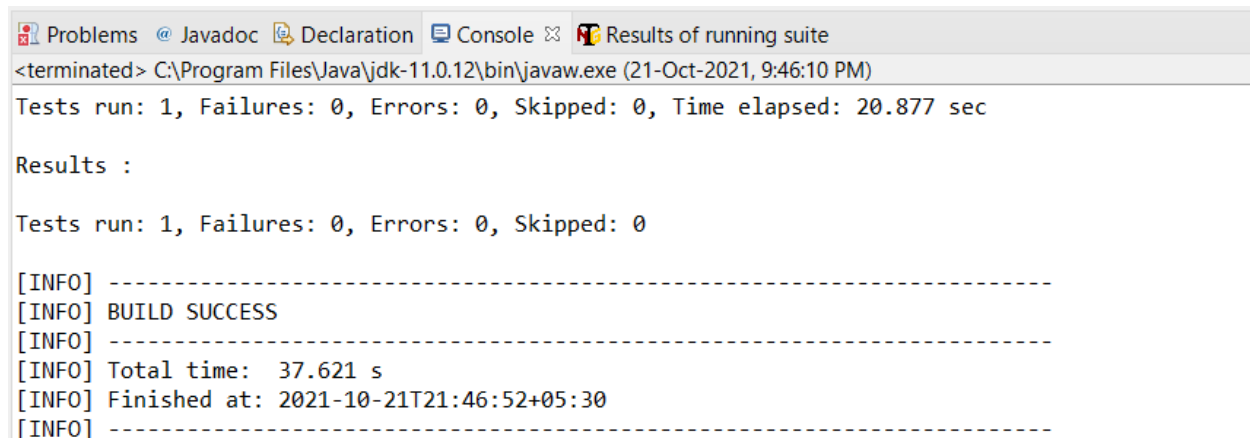
<goal>testCompile</goal>
                                </goals>
                                </pluginExecutionFilter>
                                <action>
                                    <ignore></ignore>
                                </action>
                                </pluginExecution>
                                </pluginExecutions>
                                </lifecycleMappingMetadata>
                                </configuration>
                                </plugin>
                                </plugins>
                                </pluginManagement>
</build>

```

Step 15) To run the tests in the Maven lifecycle, Right-click on the WebdriverTest and select Run As | Maven test. Maven will execute tests from the project.



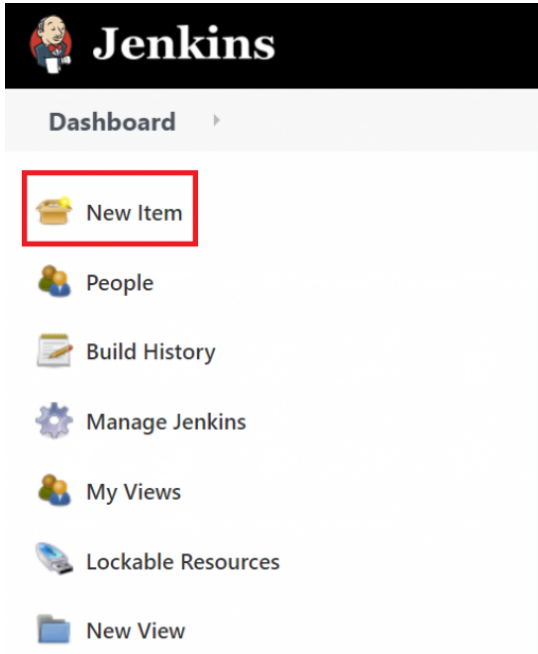
Make sure that build finished successfully.



## Setup Test

### Setting up Selenium Project on Jenkins

In the Jenkins dashboard, click New Item to create a new project. Specify the name of the project and click the Maven Project option. Click Ok.



The screenshot shows the Jenkins Dashboard. The 'New Item' button is highlighted with a red rectangle. Below the dashboard, the 'Enter an item name' form is shown with 'WebdriverTest' entered in the text field. Below the form, several project types are listed: Freestyle project, Maven project, Pipeline, Multi-configuration project, and Folder. The 'Maven project' option is selected. At the bottom, there is an 'OK' button.

**Jenkins**

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

**Enter an item name**

WebdriverTest

» Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining used for something other than software build.

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastica

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suita and/or organizing complex activities that do not easily fit in free-style job type

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such builds, etc.

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things to namespace, so you can have multiple things of the same name as lon

OK

Go to the Build section of your new job.

- In the Root POM textbox, enter full path to pom.xml
- In Goals and options section, enter "clean test"
- Click on Save



**Build**

Root POM ?

C:\Users\vkris\eclipse-workspace\WebDriverTest\pom.xml

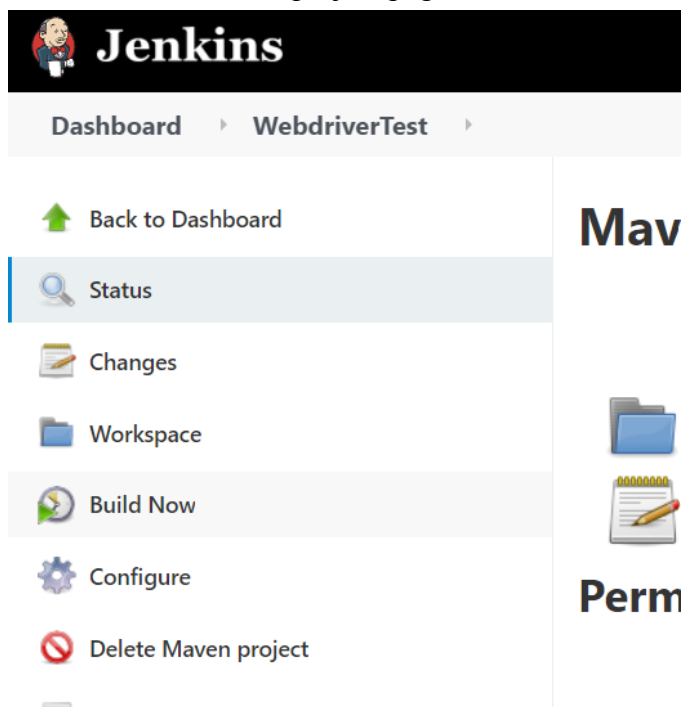
Goals and options ?

clean test

Advanced...

### Setup test:

On the WebdriverTest project page, click on the Build Now link.



Maven will build the project. It will then have TestNG execute the test cases. Once the build process is completed, check the Console Output.



## Console Output

```

Started by user V Krishnasubramaniam
Running as SYSTEM
Building on master in workspace C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\WebdriverTest
Parsing POMs
Discovered a new module WebDriverTest:WebDriverTest WebDriverTest
Modules changed, recalculating dependency graph
Established TCP socket on 29302
[WebDriverTest] $ "C:\Program Files\Java\jdk-11.0.12\bin\java" -cp C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven35-agent-1.13.jar;C:\apache-maven-3.8.2\boot\plexus-classworlds-2.6.0.jar;C:\apache-maven-3.8.2\conf\logging-jenkins.maven3.agent.Maven35Main C:\apache-maven-3.8.2 C:\Windows\System32\config\systemprofile\AppData\Local\Jenkins\war\WEB-INF\lib\remoting-4.10.jar
C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven35-interceptor-1.13.jar
C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven3-interceptor-commons-1.13.jar 29302
<==[JENKINS REMOTING CAPACITY]==>channel started
Executing Maven: -B -f C:\Users\vkris\workspace\WebDriverTest\pom.xml clean test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< WebDriverTest:WebDriverTest >-----
[INFO] Building WebDriverTest 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.12/maven-surefire-plugin-2.12.pom
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.12/maven-surefire-plugin-2.12.pom (10 kB at 6.1 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/2.12/surefire-2.12.pom
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/2.12/surefire-2.12.pom (11 kB at 27 kB/s)

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[JENKINS] Recording test results
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:16 min
[INFO] Finished at: 2021-10-21T22:15:44+05:30
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving C:\Users\vkris\workspace\WebDriverTest\pom.xml to WebDriverTest/WebDriverTest/0.0.1-SNAPSHOT/WebDriverTest-0.0.1-SNAPSHOT.pom
channel stopped
Finished: SUCCESS

```

## Conclusion

Thus, we have learnt about Selenium, its benefits and how to use it with a Maven project in Jenkins for automated testing.