[**raman@techvisionit.com**](mailto:raman@techvisionit.com)

**user@951**

**Maven & Jenkins with Selenium: Complete Tutorial**

## What is Jenkins?

Jenkins is the leading open-source continuous integration tool developed by Hudson lab. It is cross-platform and can be used on Windows, Linux, Mac OS and Solaris environments. Jenkins is written in Java. Jenkin's chief usage is to monitor any job which can be SVN checkout, cron or any application states. It fires pre-configured actions when a particular step occurs in jobs.

## Why Jenkins and Selenium?

* Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to a new environment when the tests pass.
* Jenkins can schedule your tests to run at specific time.
* You can save the execution history and Test Reports.
* Jenkins supports Maven for building and Testing a project in continuous integration.

## Why Maven & Jenkins

Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefitsApache Maven provides support for managing the full lifecycle of a test project.

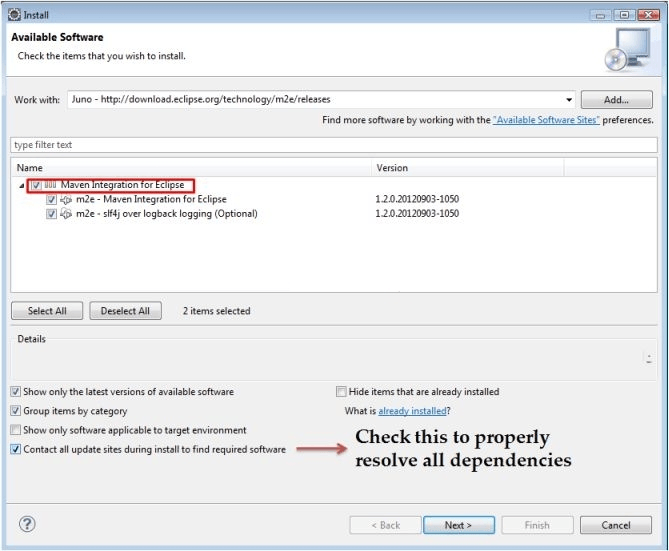
* Maven is used to define project structure, dependencies, build, and test management.
* Using pom.xml(Maven) you can configure dependencies needed for building testing and running code.
* Maven automatically downloads the necessary files from the repository while building the project.

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

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

**Step1)** In Eclipse IDE, select **Help** | **Install New Software** from Eclipse Main Menu.

**Step 2)** On the Install dialog, select **Work with** and m2e plugin as shown in the following screenshot:

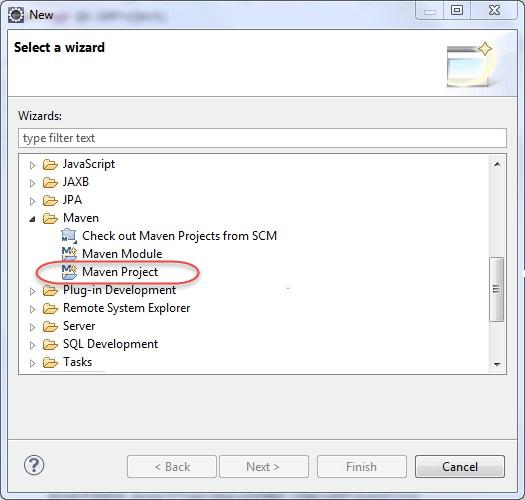
[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin1.png)

**Step 3)**Click on **Next** button and finish installation.  
  
**Configure Eclipse with Maven**

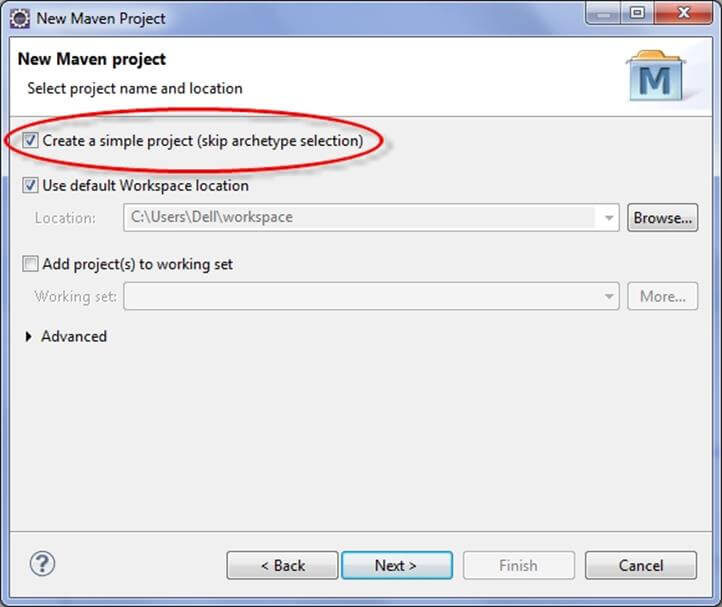
With m2e plugin is installed, we now need create Maven project.

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

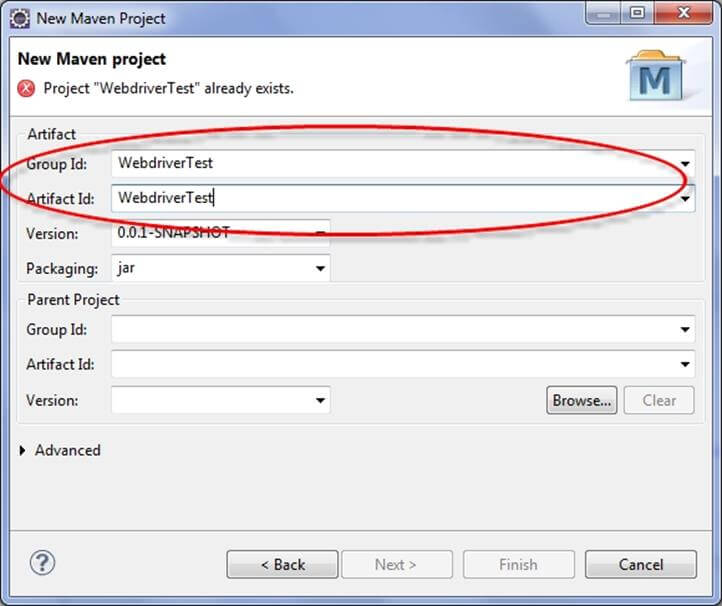
**Step 2)** On the **New** dialog, select **Maven** | **Maven Project** and click Next

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin2.png)

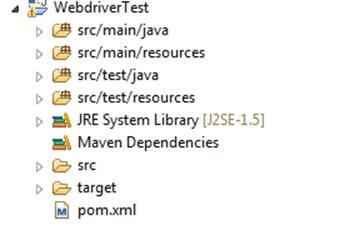
**Step 3)**On the **New Maven Project** dialog select the **Create a simple project** and click Next

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin3.jpg)

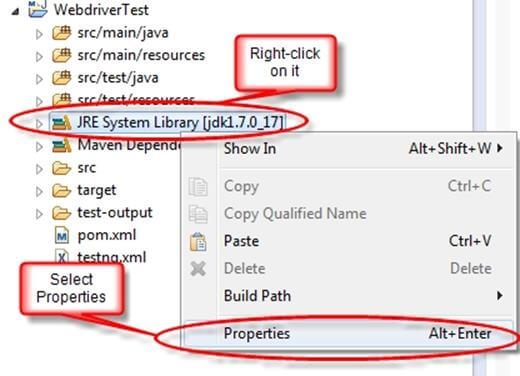
**Step 4)**Enter WebdriverTest in **Group Id**: and **Artifact Id**: and click finish

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin4.jpg)

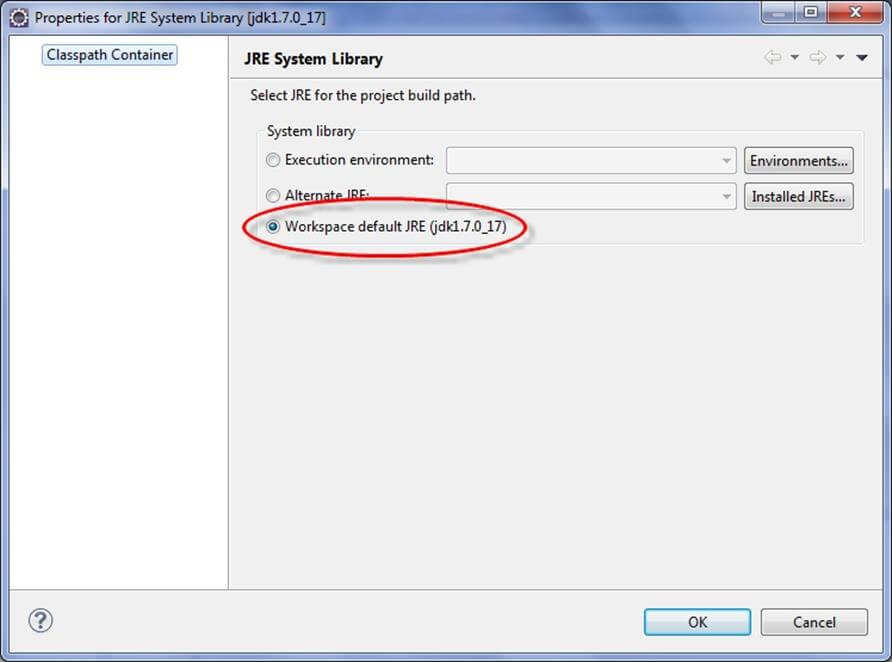
**Step 5)** Eclipse will create **WebdriverTest** with following structure:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin5.jpg)

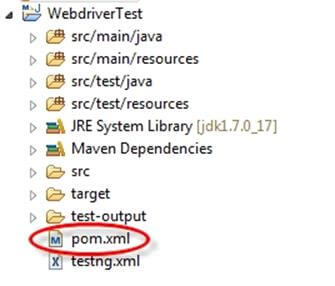
**Step 6)** Right-click on **JRE System Library** and select the **Properties** option from the menu.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin6.jpg)

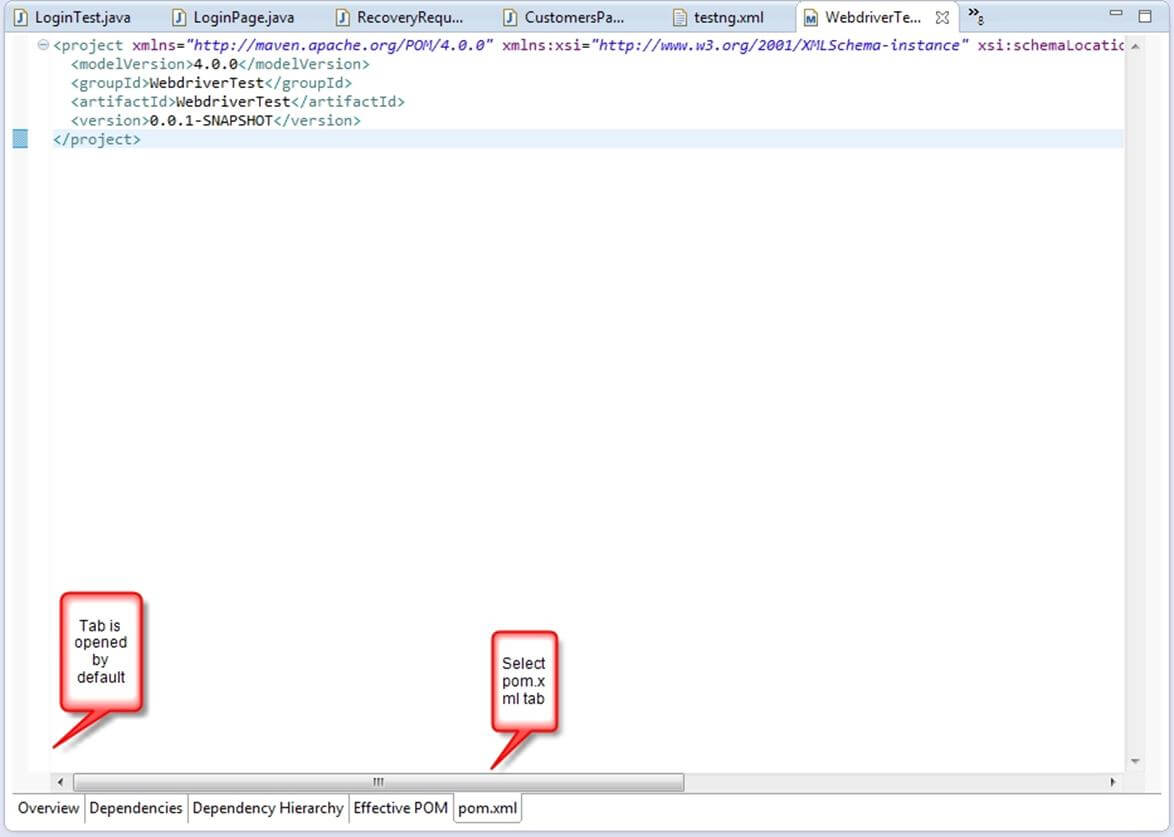
On the **Properties for JRE System Library** dialog box, make sure **Workspace default JRE** is selected and click OK

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin7.jpg)

**Step 7)**. Select **pom.xml** from **Project Explorer**..

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin8.jpg)

pom.xml file will Open in Editor section

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin9.jpg)

**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>2.45.0</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

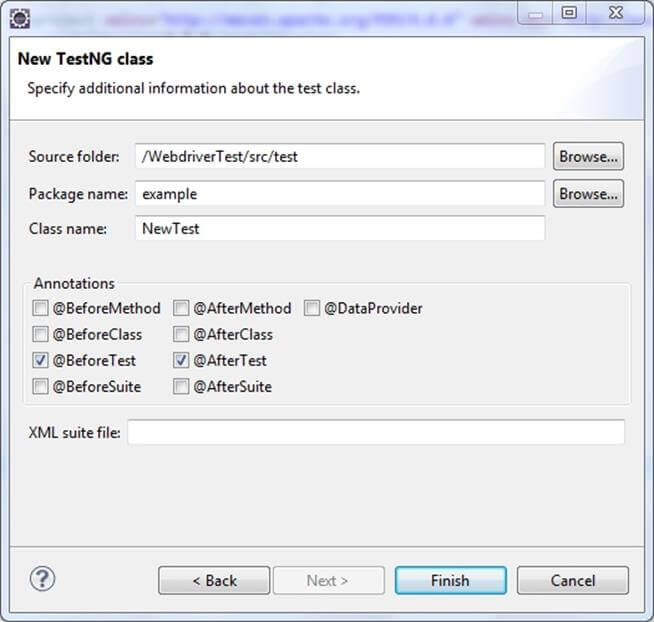
<version>6.8</version>

<scope>test</scope>

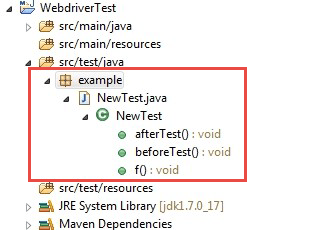
</dependency>

</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 as shown in the following screenshot:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin10.jpg)

**Step 10)**. Eclipse will create the NewTest class as shown in the following screenshot:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin11.png)

**Step 11)**Add the following code to the **NewTest** class:

This code will verify the title of Guru99 Selenium Page

package example;

import org.openqa.selenium.By;

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() {

driver = new FirefoxDriver();

}

@AfterTest

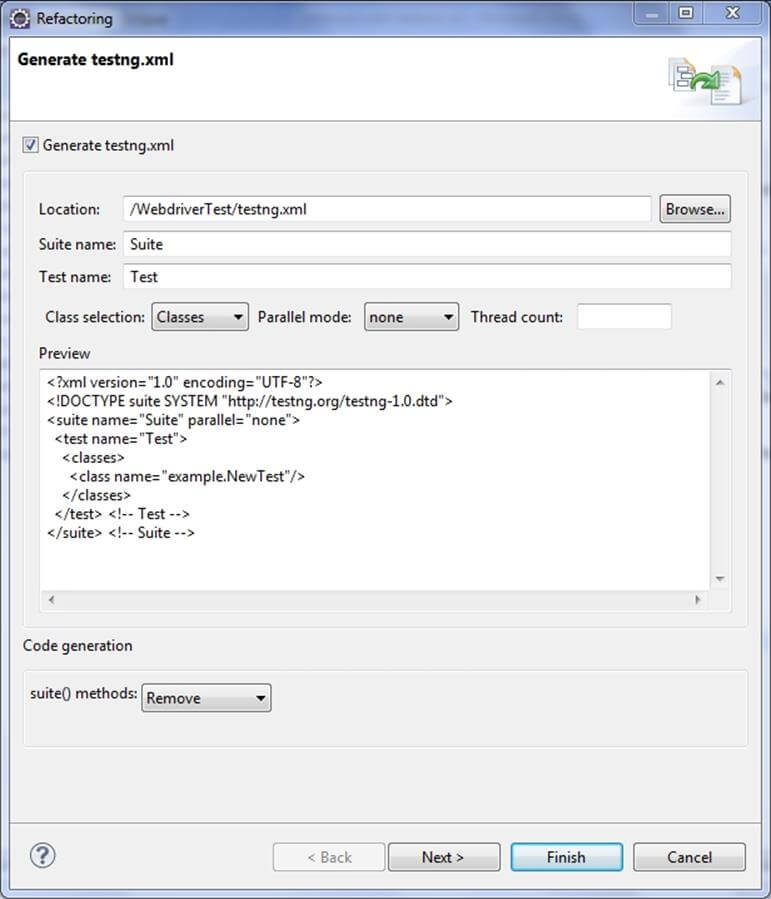
public void afterTest() {

driver.quit();

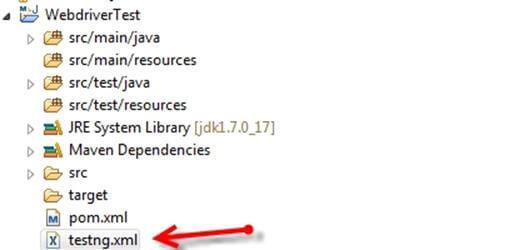
}

}

**Step 12)** 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** as shown in the following screenshot:

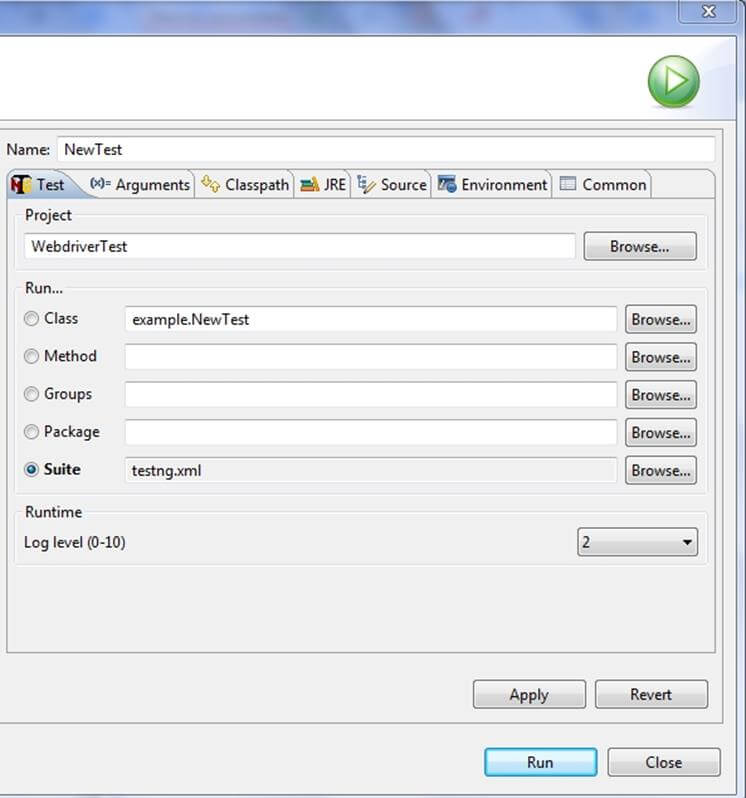
[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin12.jpg)

Update the project and make sure that file appears in the tree **Package Explorer** (right click on the project - Refresh).

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin13.jpg)

**Step 13)**Now you need to run 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

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin14.jpg)

Make sure that build finished successfully.

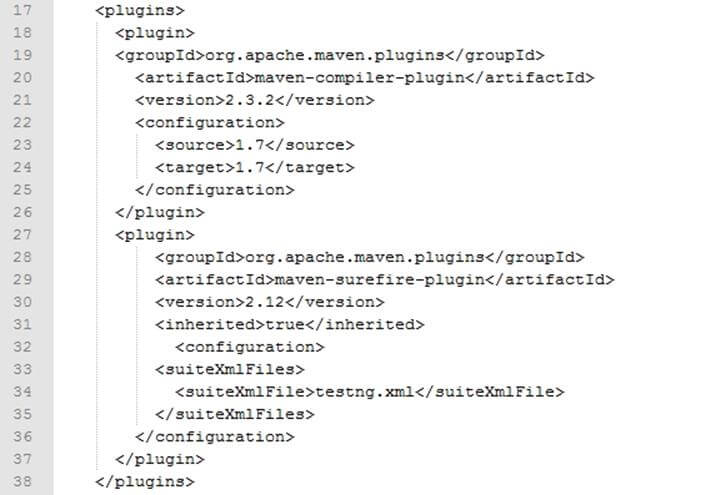
**Step 14)**. Additionally, we need to add

1. maven-compiler-plugin
2. maven-surefire-plugin
3. testng.xml

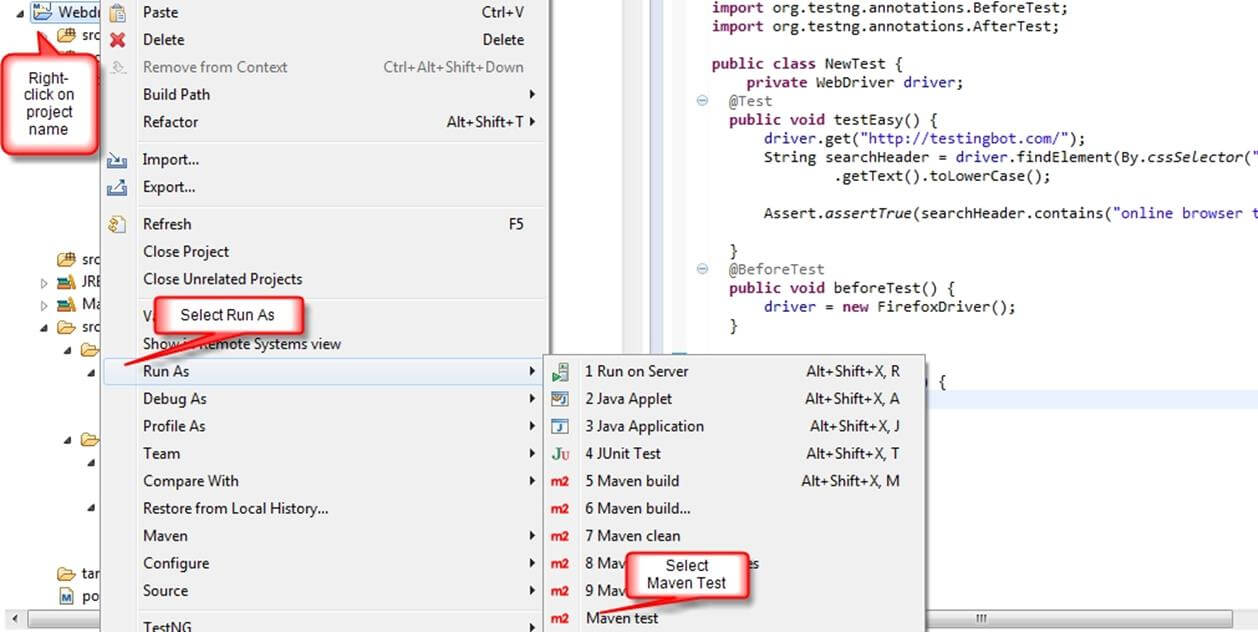
to pom.xml**.**

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:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin15.jpg)

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

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin16.jpg)

Make sure that build finished successfully.

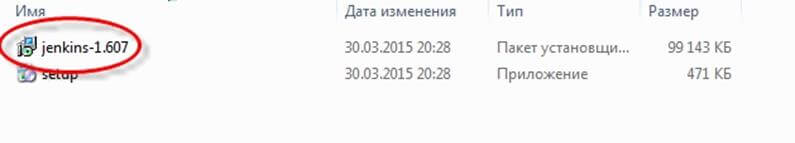
## Steps to Install Jenkins and configure it to Run Maven with TestNg Selenium

**Installation**

**Step 1)** Go to http://jenkins-ci.org/and download correct package for your OS. Install Jenkins.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin17.jpg)

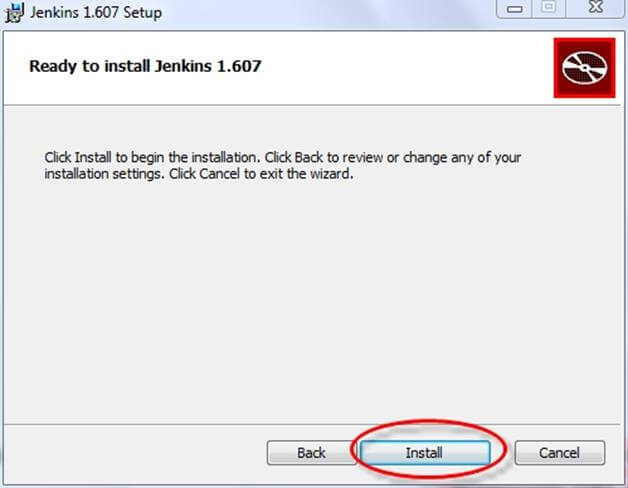
**Step 2)**Unzip Jenkins to specified folder. Run exe file as shown in following screenshot:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin18.jpg)

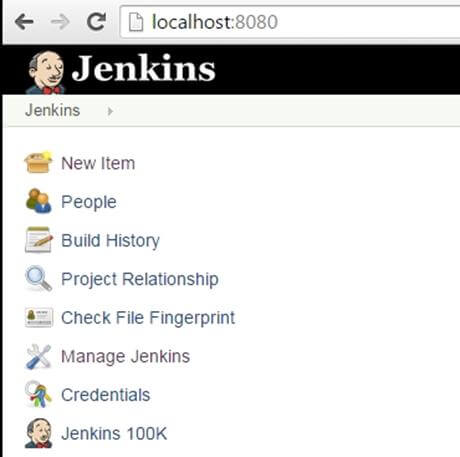
**Step 3)**In **Jenkins 1.607 Setup** window click on **Next** button.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin19.jpg)

**Step 4)**Click on **Install** button in the end.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin20.jpg)

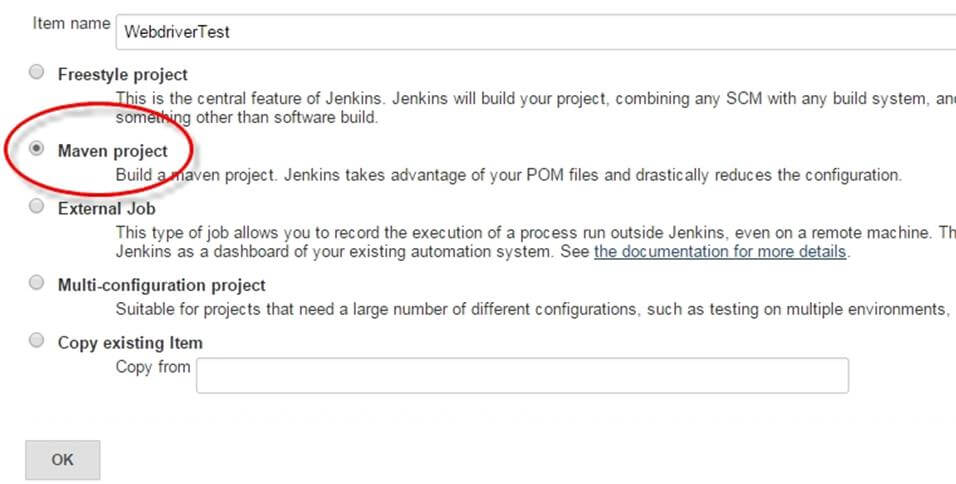
**Step 5)**Once installation is done, navigate to the Jenkins Dashboard (http://localhost:8080 by default) in the browser window.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin21.jpg)

**Step 6)**Click on the **New Item** link to create a CI job.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin22.jpg)

**Step 7)**Select the Maven project radio button as shown in the following screenshot:

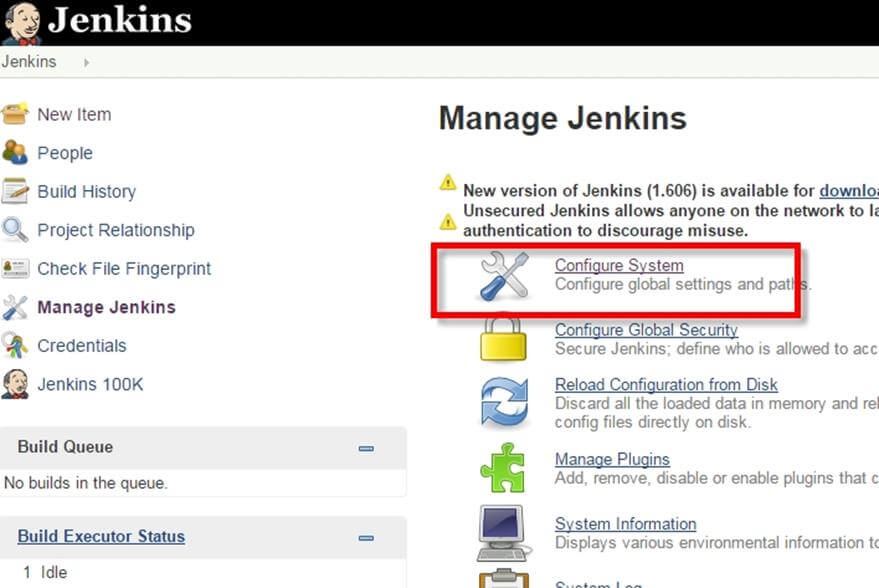
[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin23.jpg)

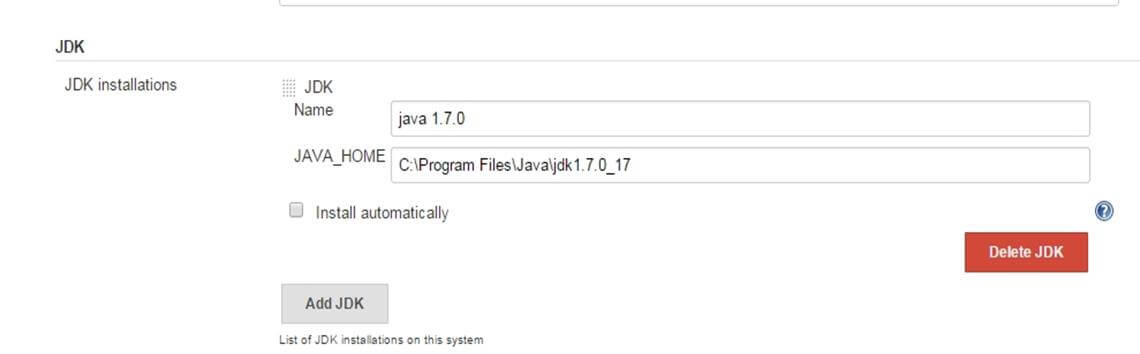
Using the Build a **Maven Project** option, Jenkins supports building and testing Maven projects.

**Step 6)**Click on OK button. A new job with name "WebdriverTest" is created in Jenkins Dashboard.

[Maven & Jenkins with Selenium: Complete Tutorial](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin24.jpg)

**Step 7)** Go to **Manage Jenkins** => **Configure System** as shown in the following screenshot.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin25.jpg)

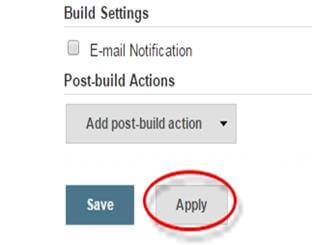
Click on JDK installations and configure JDK as in the following screenshot:  
  
[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin26.jpg)

**Step 8)**Go to the **Build** section of new job.

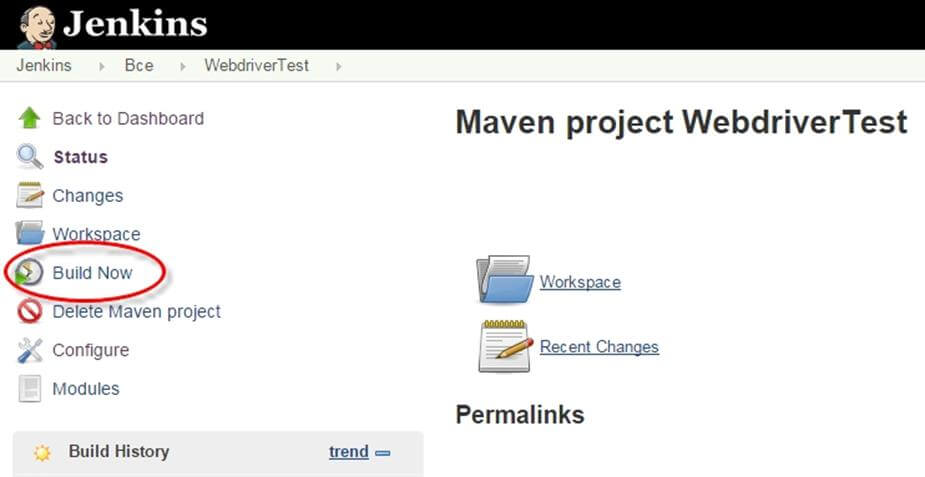
* In the **Root POM** textbox, enter full path to pom.xml
* In Goals and options section, enter "clean test"

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin27.jpg)

**Step 9)**Click on **Apply** button.

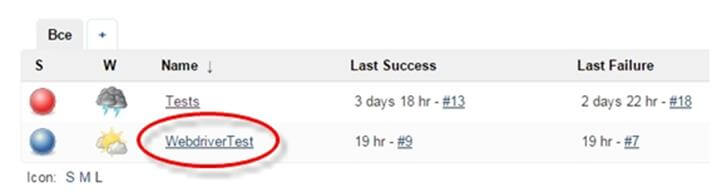
[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin28.jpg)

**Step 10)**On the WebdriverTest project page, click on the **Build Now** link.

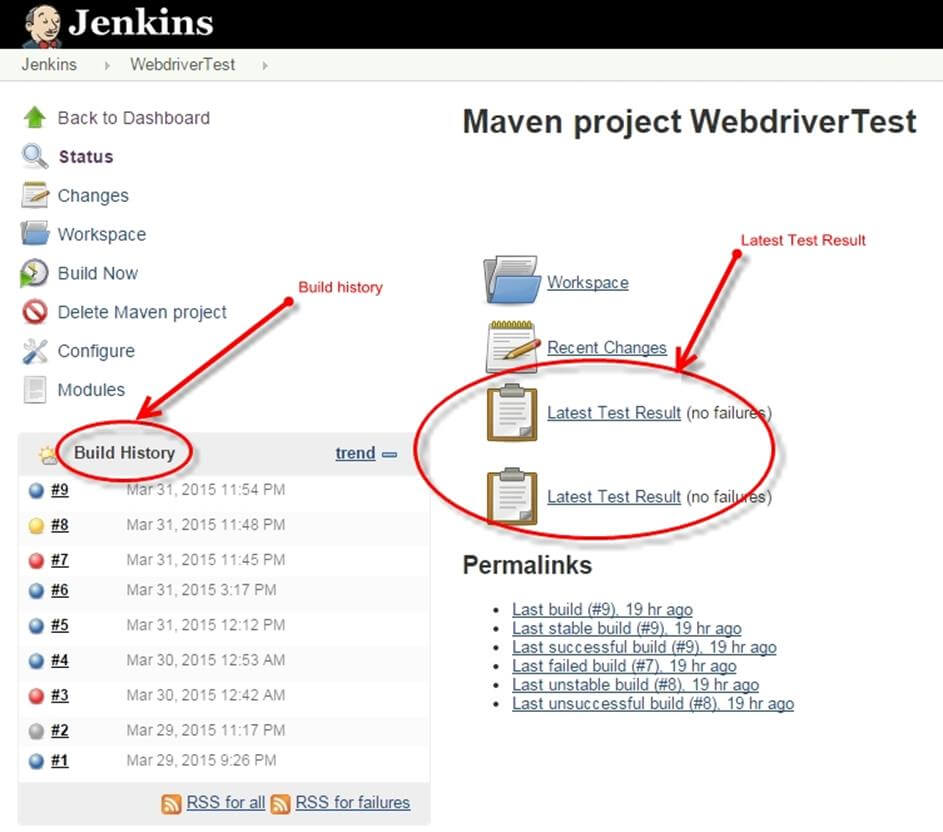
[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin29.jpg)

Maven will build the project. It will then have TestNG execute the test cases.

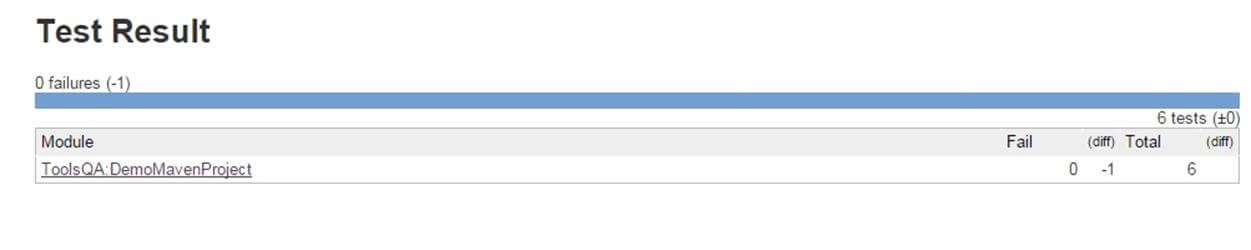
**Step 11)**Once the build process is completed, in Jenkins Dashboard click on the **WebdriverTest** project

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin30.jpg)

**Step 12)**The WebdriverTest project page displays the build history and links to the results as shown in the following screenshot:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin31.jpg)

**Step 13)**Click on the "Latest Test Result" link to view the test results as shown in the following screenshot:

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin32.jpg)

**Step 14)**. Select specific build, and you will see the current status by clicking on "**console output**".

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin33.jpg)

## Scheduling Jenkins for automatic execution.

Scheduling builds(Selenium Tests) is one of the important features of Jenkins where it automatically triggers the build, based on defined criteria. Jenkins provides multiple ways to trigger the build process under the Build Trigger configuration.

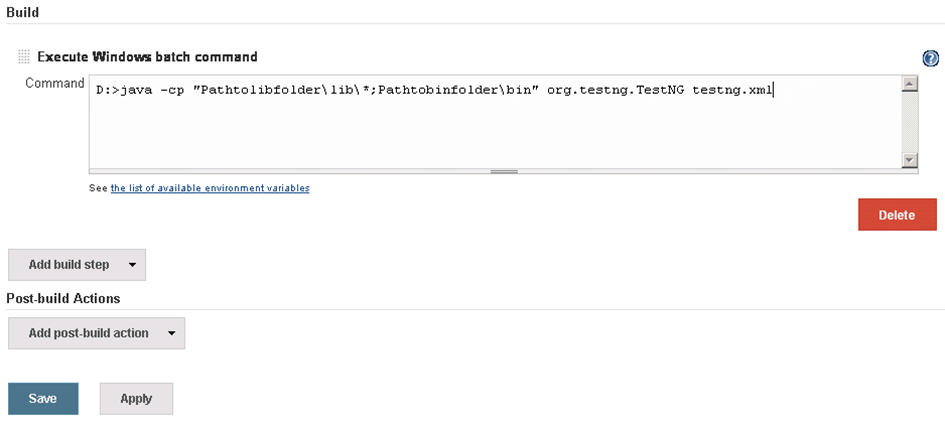
For example:   
Enter 0 23 \* \* \* in the Schedule textbox as shown in the following screenshot. This will trigger the build process every day at 11 p.m.

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin34.jpg)

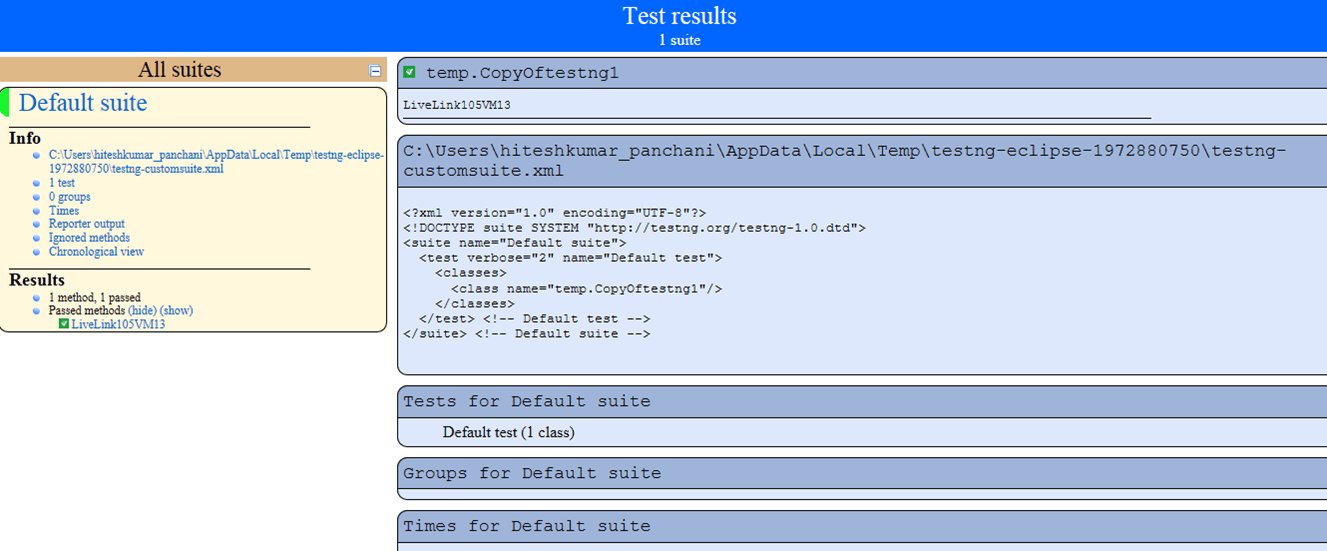
## Using Jenkings without Maven

To run pure TestNg script in Jenkins, enter the following in build

**D:>java -cp "Pathtolibfolder\lib\\*;Pathtobinfolder\bin" org.testng.TestNG testng.xml**

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin35.png)

* Click on Save button.
* Note: The actual path of lib and bin folder need to add in above command.
* After saving the command, Jenkins will build project in predefined time, and this command will be run using TestNG.
* Result will be stored in custom report HTML file that can be sent via email with a Jenkin configuration
* Output of the code will be

[](https://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin36.png)

## Benefits of using Jenkins

1. Early issue finding – Bug can be detected in early phase of the software development
2. Automatic integration – no separate effort required to integrate all changes
3. Installer – a deployable system available at any point of development
4. Records – part build records maintained
5. Support and Plugins: One of the reasons for Jenkin's popularity is the availability of large community support. Also, lots of ready-made plugins are available which help you expand its functionality.