Maven & Jenkins with Selenium: Complete Tutoria

What is Jenkins?

Jenkins is the leading open-source continuous
developed by Hudson lab. It is cross-platform
Windows, Linux, Mac OS and Solaris environm
written in Java. Jenkin's chief usage is to monit
can be SVN checkout, cron or any application s
configured actions when a particular step occu

In this tutorial, we will learn

Important Features of Jenkins

Important Features of Jenkins

- Change Support: Jenkins generates the list of all changes done in repositories like SVN.
- Permanent links: Jenkins provides direct links to the latest build or failed build that can b
  communication
- Installation: Jenkins is easy to install either using direct installation file (exe) or war file to
  application server.
- Email integration: Jenkins can be configured to email the content of the status of the buil
- Easy Configuration: To configure various tasks on Jenkins is easy.
- TestNG test: Jenkins can be configured to run the automation test build on TestNG after e
- Multiple VMs: Jenkins can be configured to distribute the build on multiple machines.
- Project build: Jenkins documents the details of jar, version of jar and mapping of build ar
- Plugins: 3rd party plugin can be configured in Jenkins to use features and additional func

Why Jenkins and Selenium?

- Running Selenium tests in Jenkins allows you to run your tests every time your software
  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 bui
framework, it feels underpowered. Integrating Maven with Selenium provides following ber
Apache Maven provides support for managing the full lifecycle of a test project.

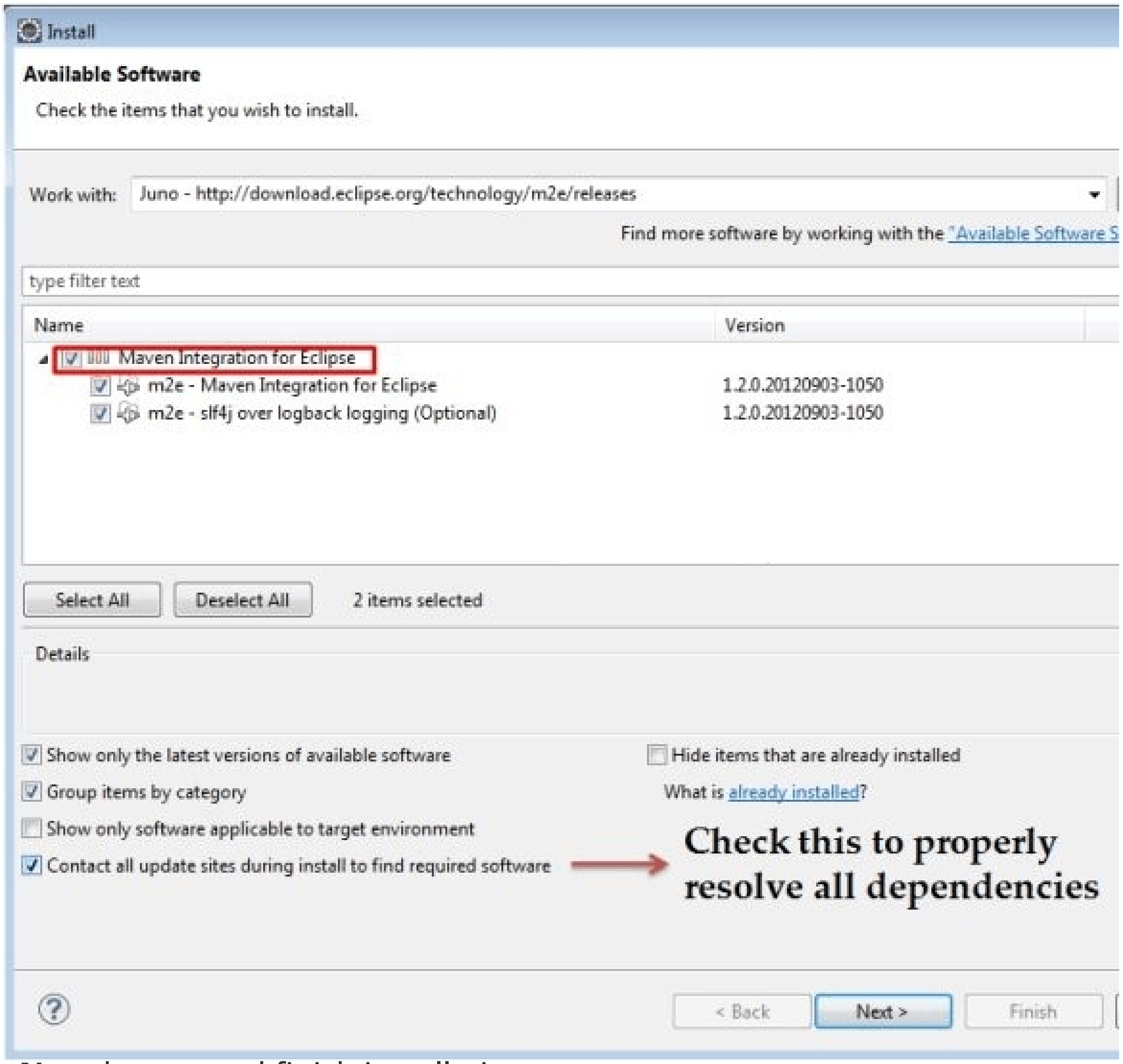
- Maven is used to define project structure, dependencies, build, and test management.
- Using pom.xml(Maven) you can configure dependencies needed for building testing and
- Maven automatically downloads the necessary files from the repository while building th

Steps to use install Maven and use it with TestNG Selenium

For this tutorial, we will use Eclipse (Juno) IDE for Java Developers to set up Selenium WebD
Additionally, we need add m2eclipse plugin to Eclipse to facilitate the build process and cre
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 sc


Install

Available Software

Check the items that you wish to install.

Work with: Juno - http://download.eclipse.org/technology/m2e/releases

Find more software by working with the "Available Software S

type filter text

Name	Version
☑ Maven Integration for Eclipse	
☑ m2e - Maven Integration for Eclipse	1.2.0.20120903-1050
☑ m2e - slf4j over logback logging (Optional)	1.2.0.20120903-1050

2 items selected

Details

☑ Show only the latest versions of available software

☐ Hide items that are already installed

☑ Group items by category

What is already installed?

☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

Check this to properly resolve all dependencies

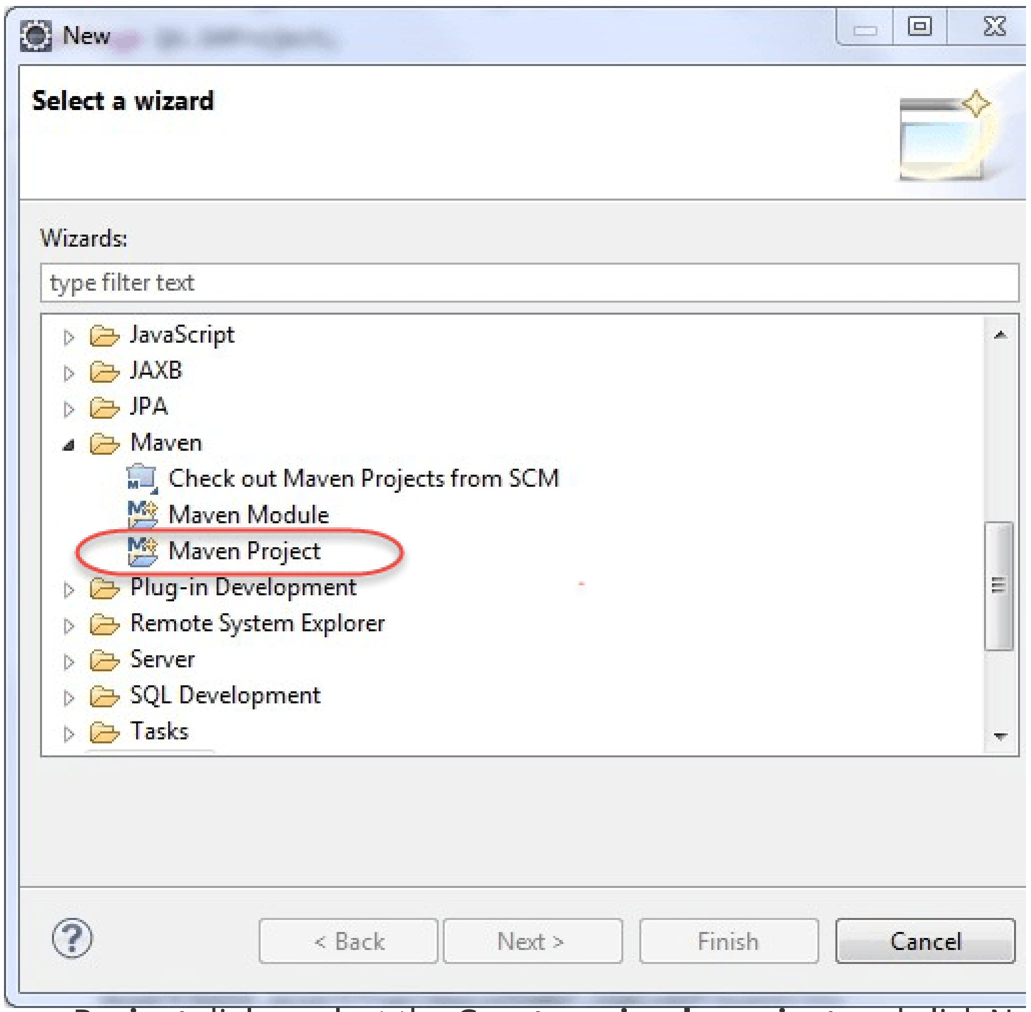
Step 3) Click on **Next** button and finish installation.

Configure Eclipse with Maven

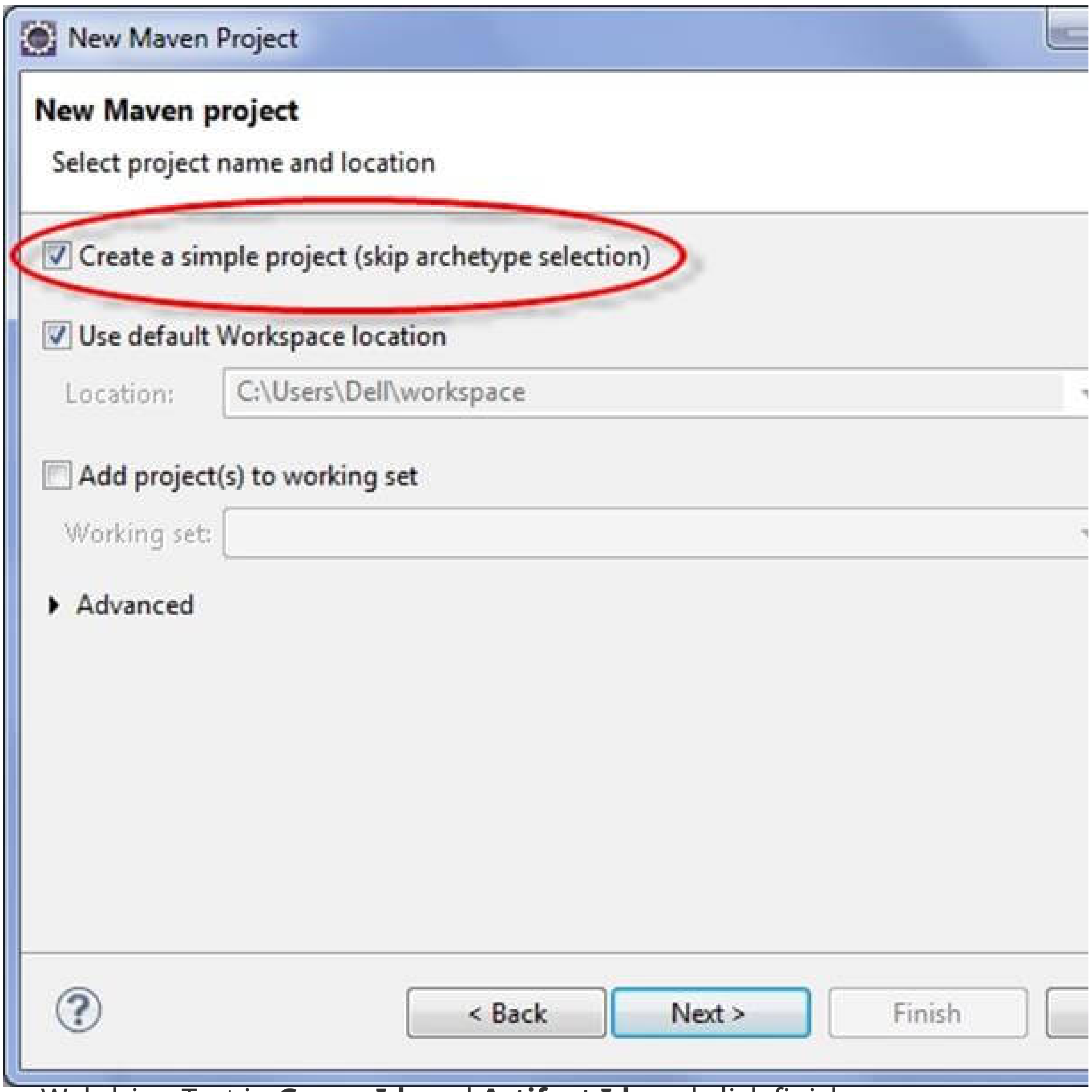
With m2e plugin is installed, we now need create Maven project.

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

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 Ne


New Maven Project

New Maven project

Select project name and location

[✓] Create a simple project (skip archetype selection)

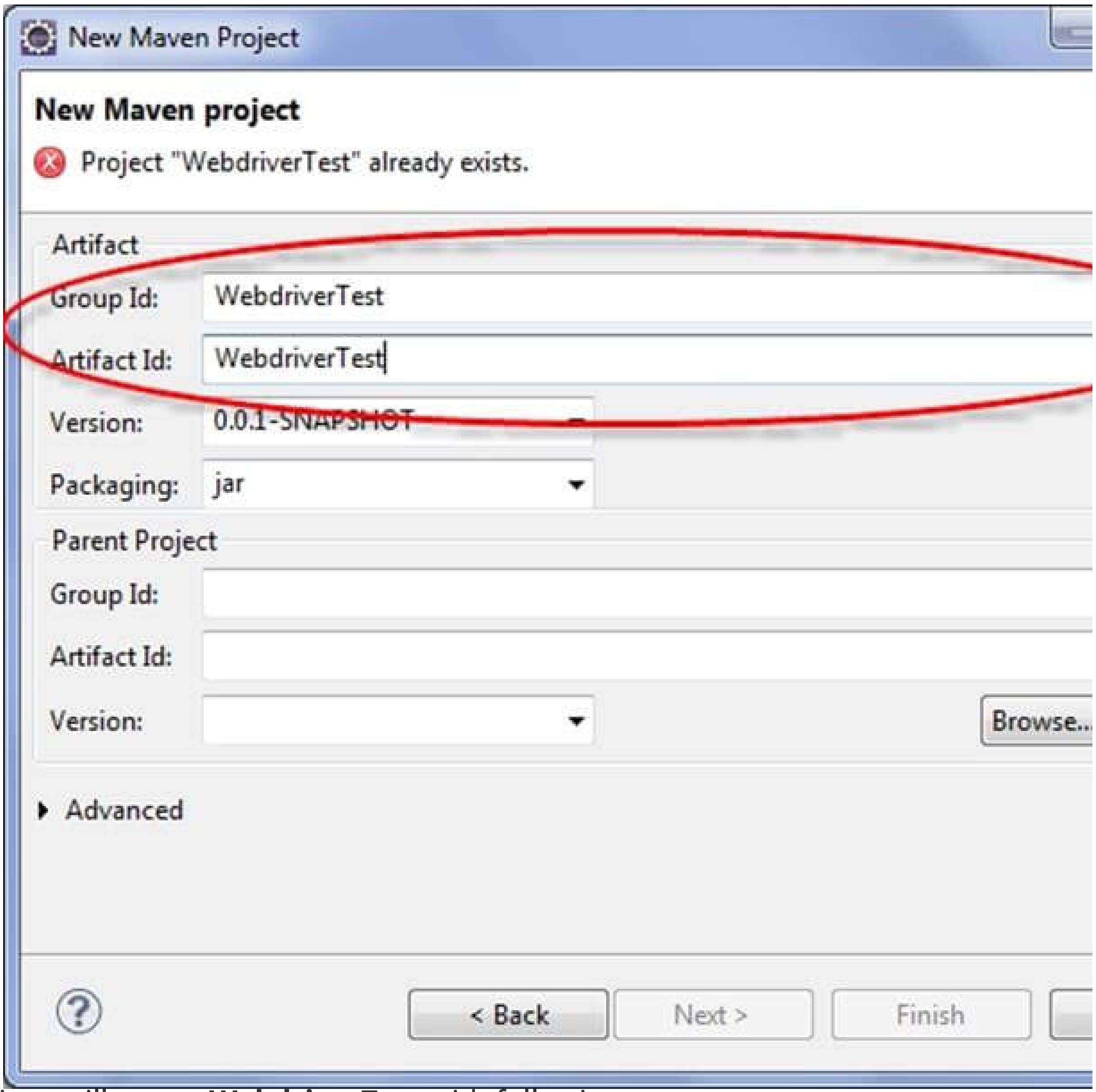
[✓] Use default Workspace location

Location: C:\Users\Dell\workspace

[ ] Add project(s) to working set

Working set:

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



New Maven Project

New Maven project

Project "WebdriverTest" already exists.

Artifact

Group Id: WebdriverTest

Artifact Id: WebdriverTest

Version: 0.0.1-SNAPSHOT

Packaging: jar

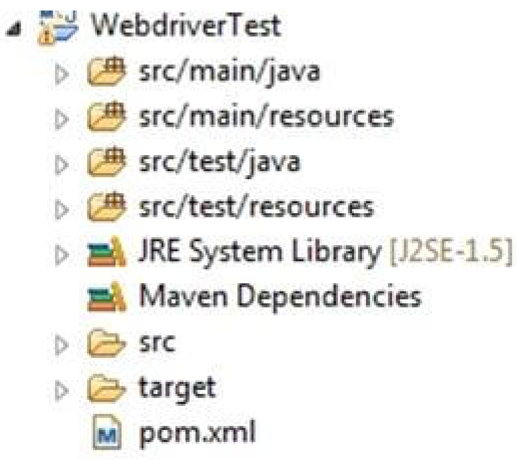
Parent Project

Group Id:

Artifact Id:

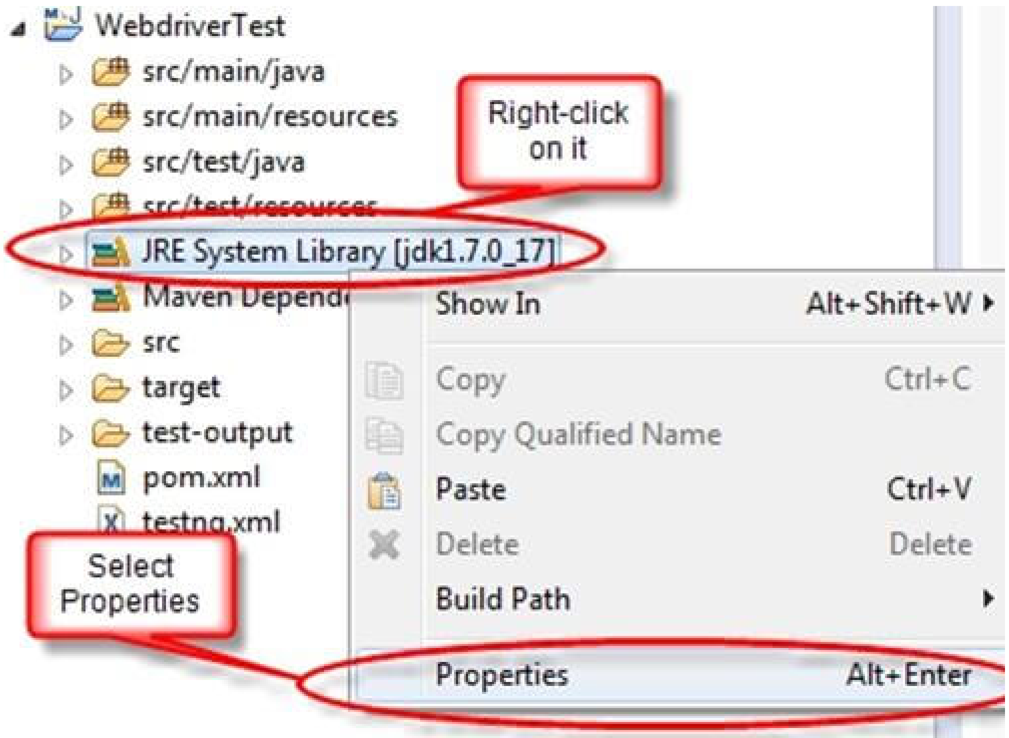
Version:

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

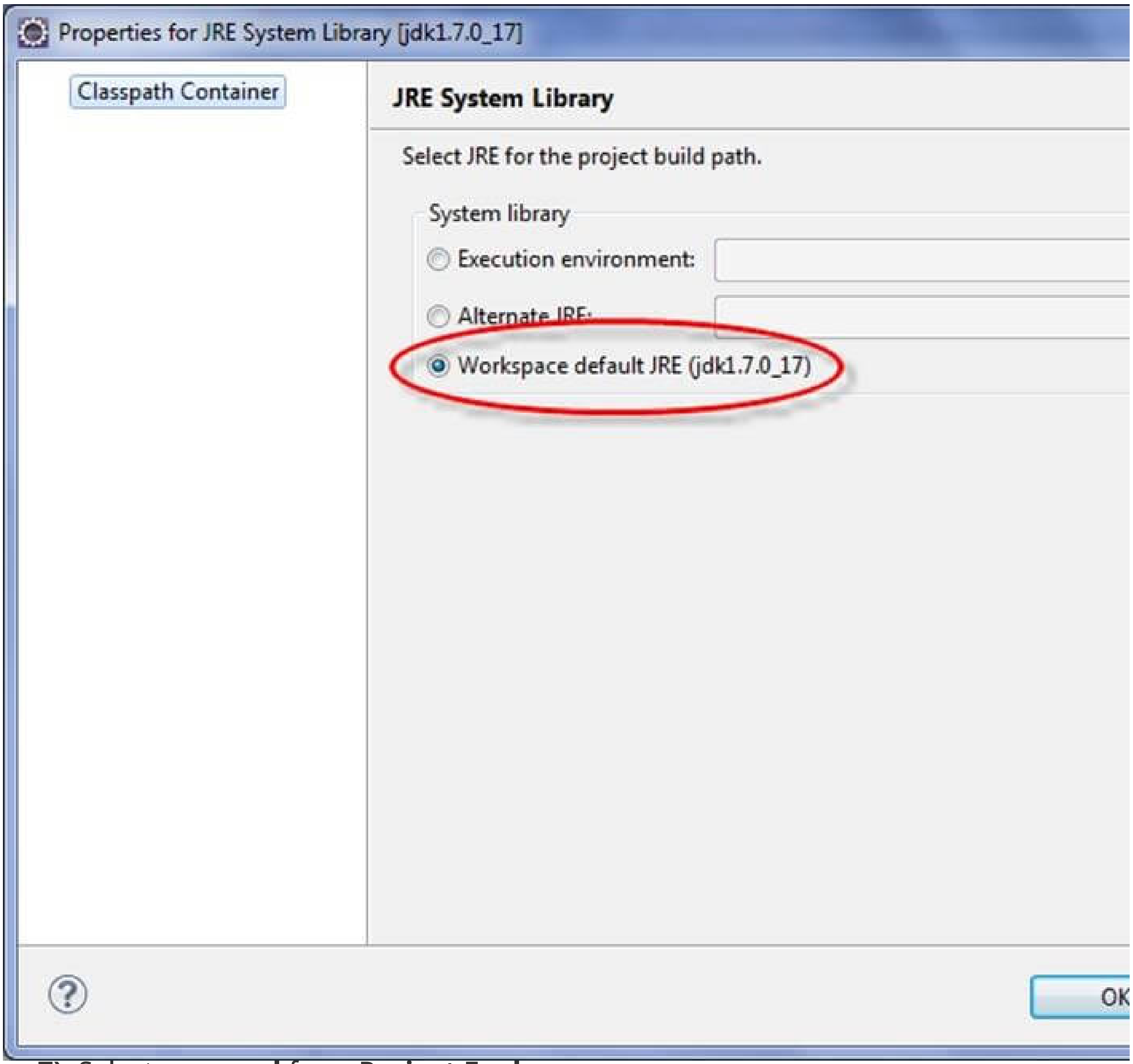


```
WebdriverTest
  src/main/java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
  target
  pom.xml
```

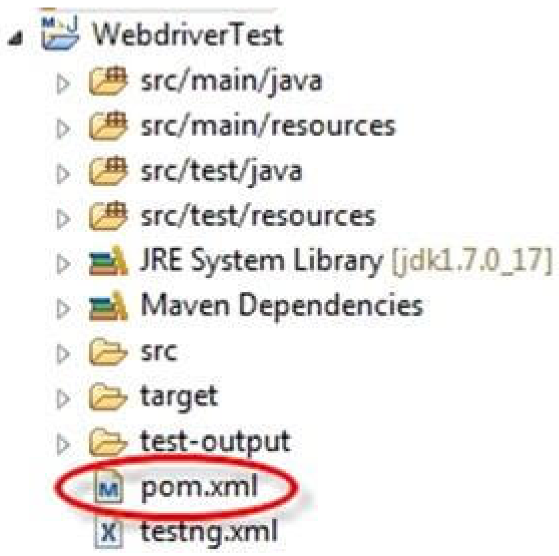
Step 6) Right-click on **JRE System Library** and select the **Properties** option from the menu



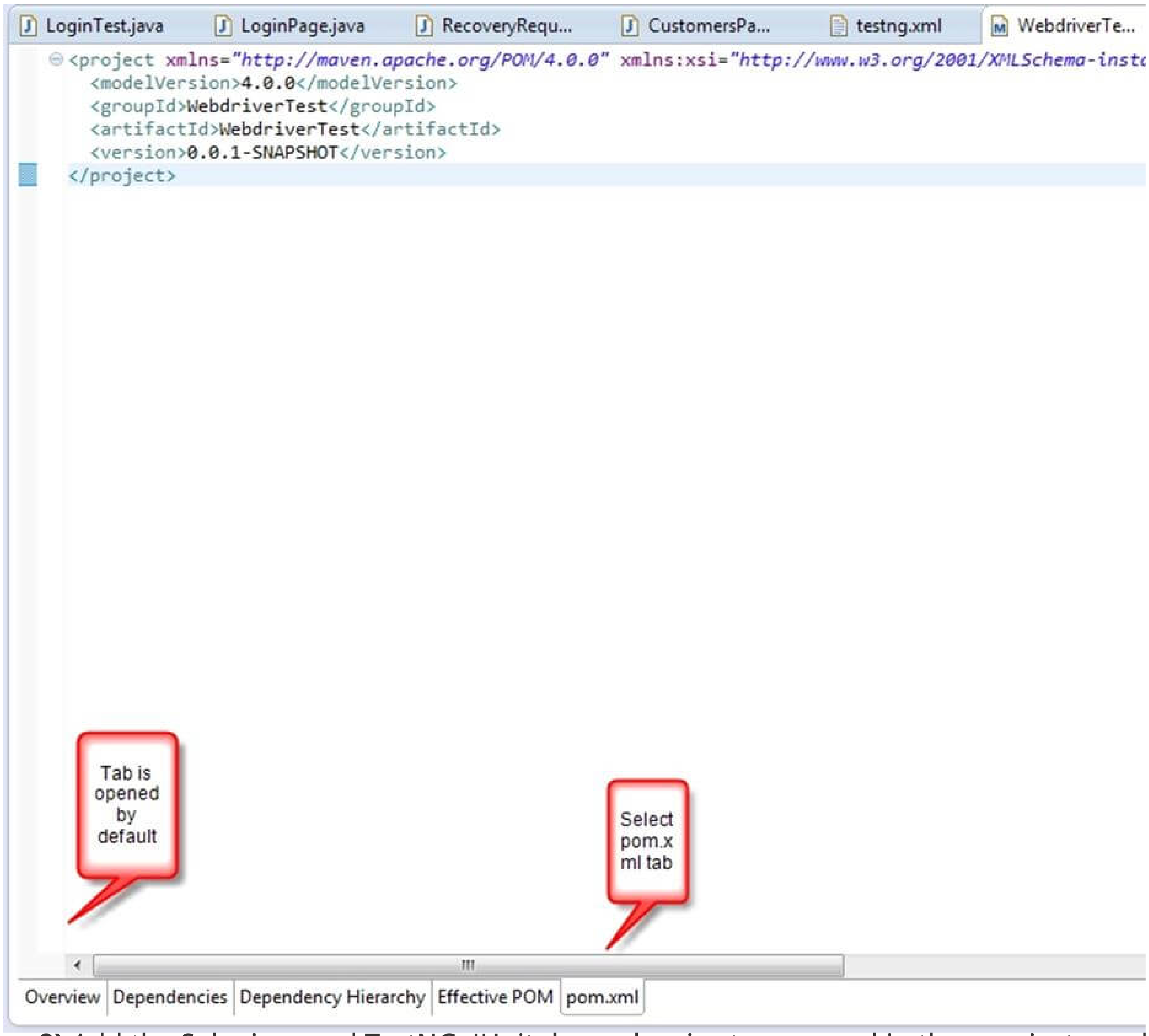
On the **Properties for JRE System Library** dialog box, make sure **Workspace default JRE** i
OK


Properties for JRE System Library [jdk1.7.0_17]
Classpath Container
JRE System Library
Select JRE for the project build path.
System library
Execution environment:
Alternate JRE:
Workspace default JRE (jdk1.7.0_17)
OK

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



pom.xml file will Open in Editor section



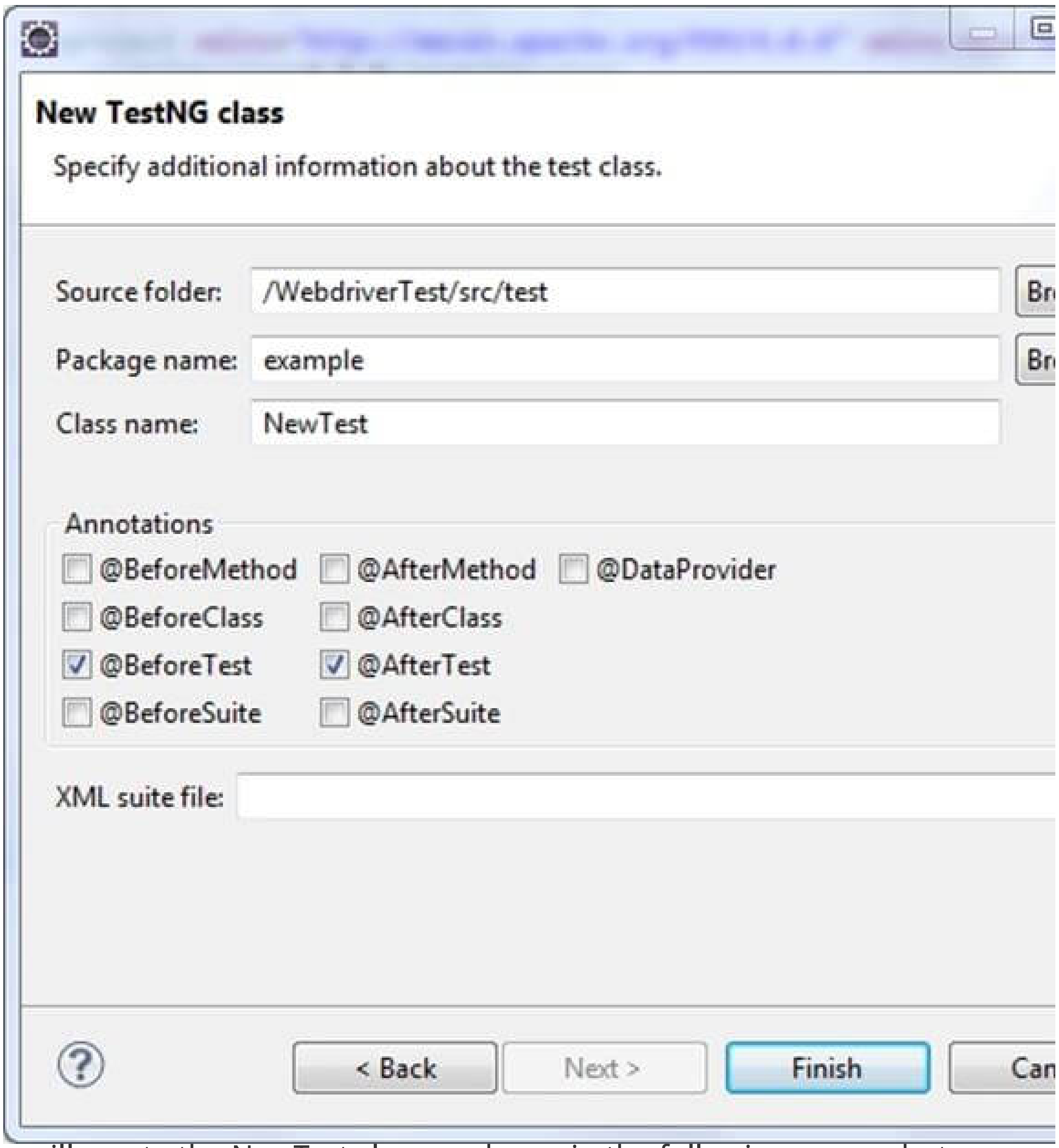
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
  <modelVersion>4.0.0</modelVersion>
  <groupId>WebdriverTest</groupId>
  <artifactId>WebdriverTest</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

Step 8).Add the Selenium and TestNG, JUnit dependencies to pom.xml in the <project> nod

```
<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
click on the **Finish** button as shown in the following screenshot:



New TestNG class

Specify additional information about the test class.

Source folder: /WebdriverTest/src/test

Package name: example

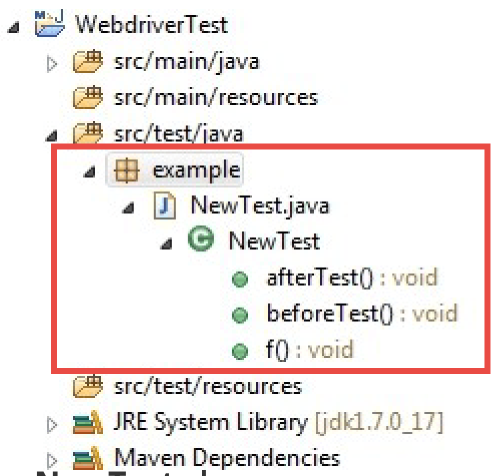
Class name: NewTest

Annotations

☐ @BeforeMethod	☐ @AfterMethod	☐ @DataProvider
☐ @BeforeClass	☐ @AfterClass	
✓ @BeforeTest	✓ @AfterTest	
☐ @BeforeSuite	☐ @AfterSuite	

XML suite file:

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

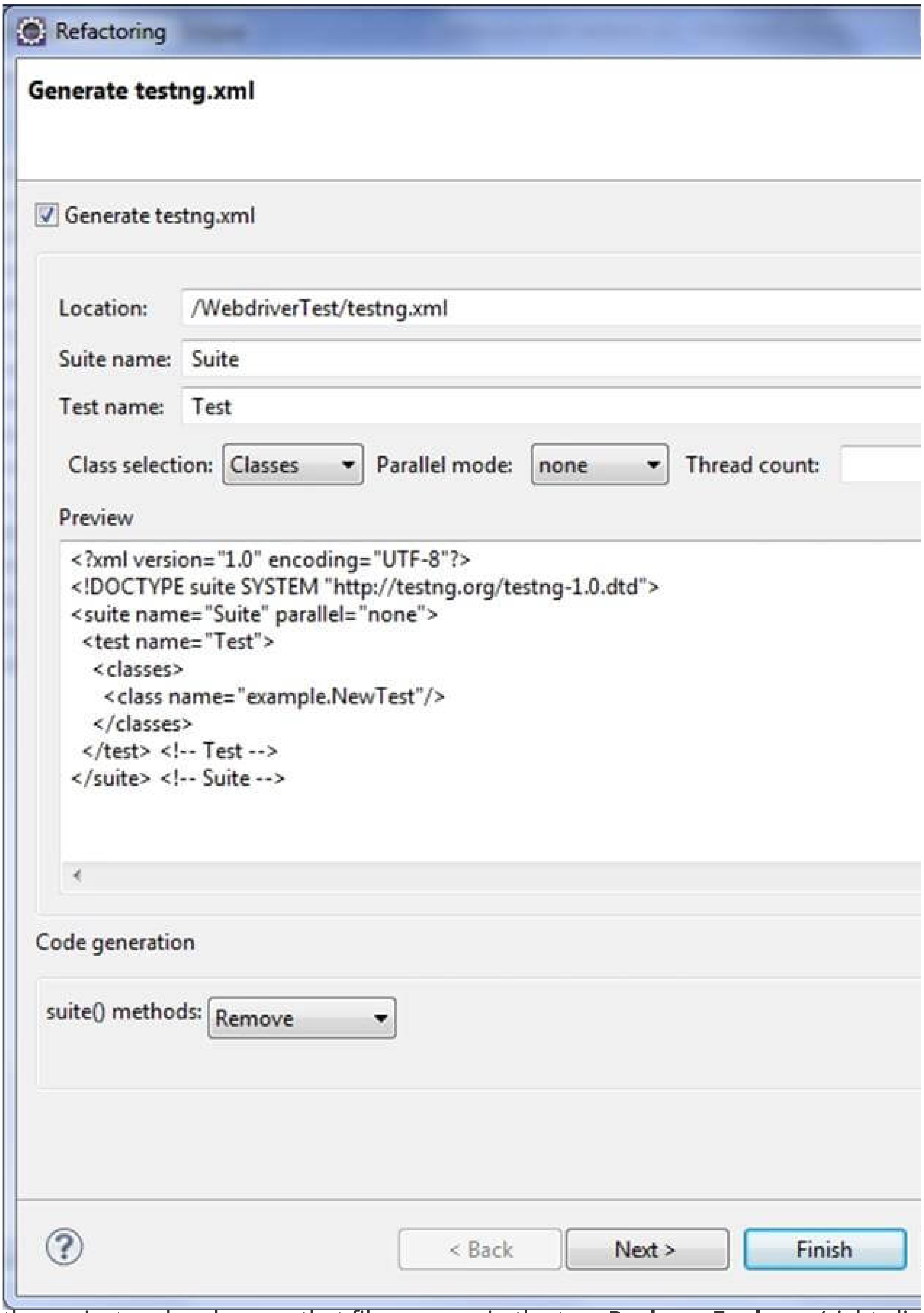


Step 11) Add the following code to the **NewTest** class:

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://www.guru99.com/selenium-tutorial.html");
                        String title = driver.getTitle();
                        Assert.assertTrue(title.contains("Free Selenium Tutorials"));
                }
                @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
in the following screenshot:



☑ Generate testng.xml

Location: /WebdriverTest/testng.xml
Suite name: Suite
Test name: Test
Class selection: Classes Parallel mode: none Thread count:

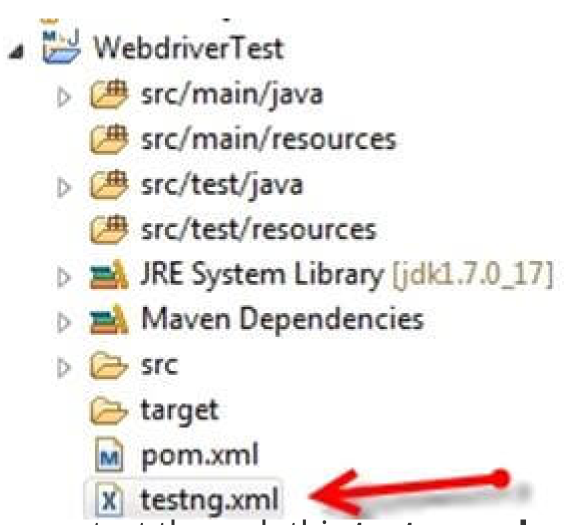
Preview

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

Code generation

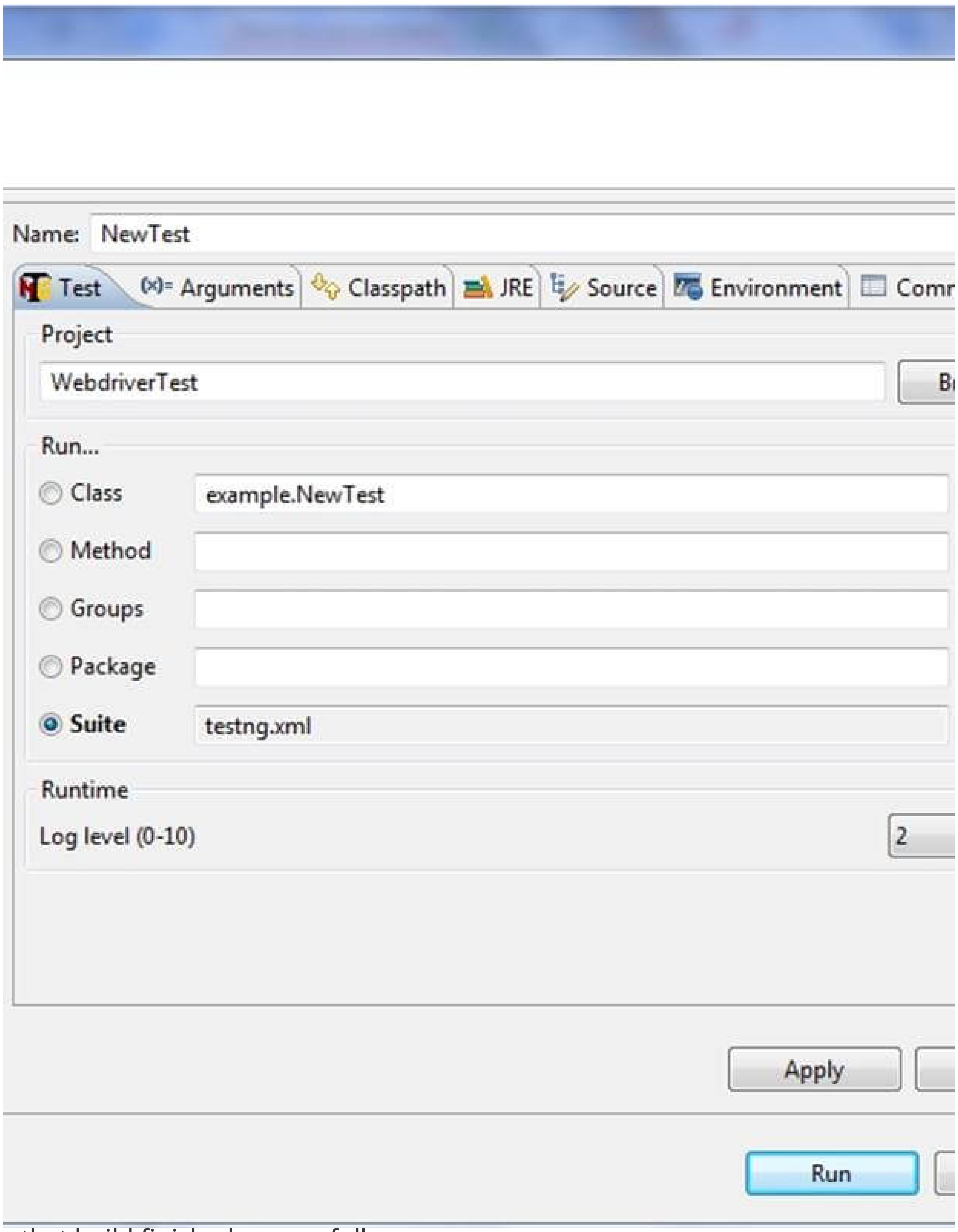
suite() methods: Remove

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



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 f
testng.xml and click Run


Name: NewTest

Test | Arguments | Classpath | JRE | Source | Environment | Comr

Project

WebdriverTest

Run...

Class: example.NewTest

Method:

Groups:

Package:

Suite (selected): testng.xml

Runtime

Log level (0-10): 2

Make sure that build finished successfully.

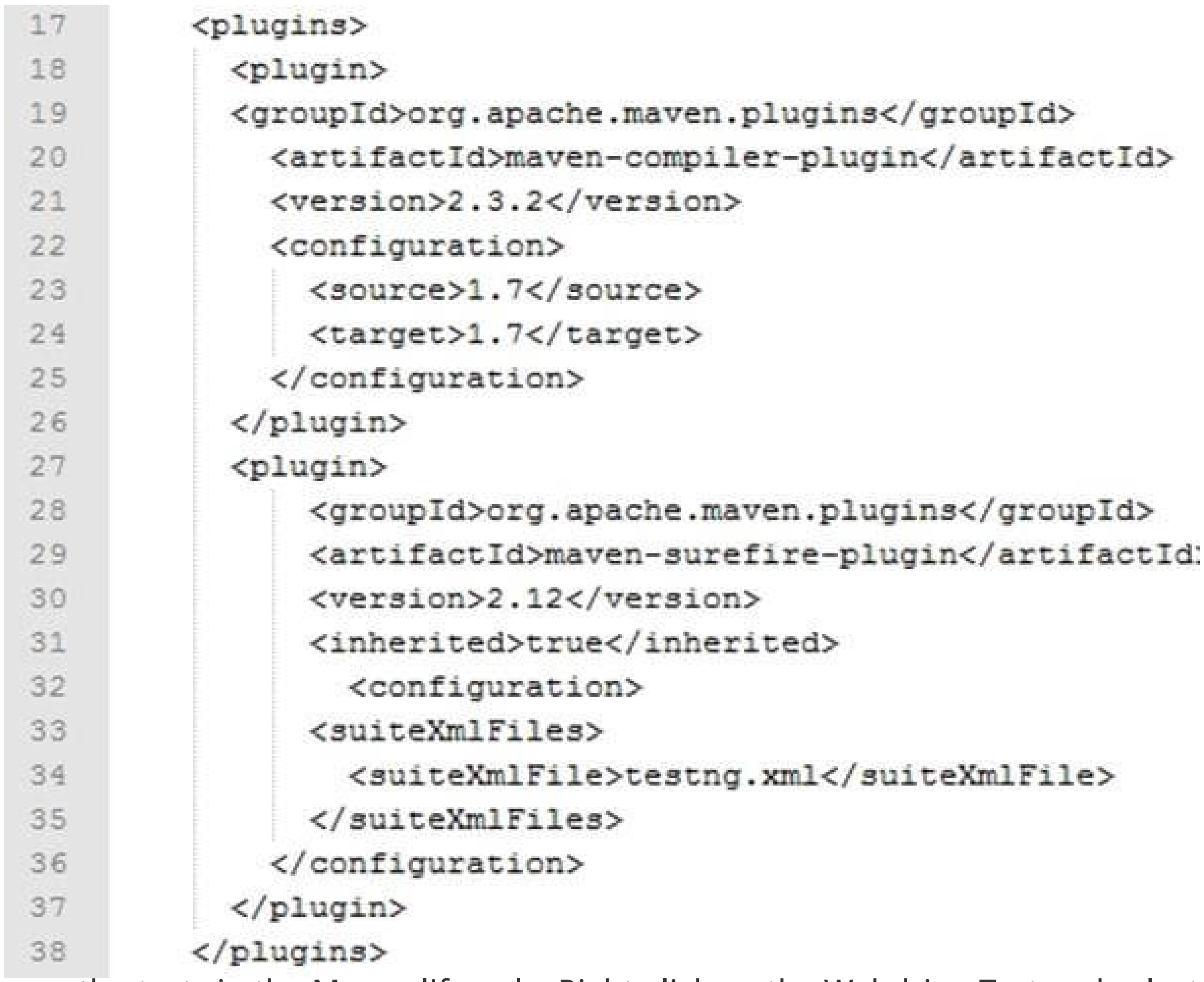
Step 14). Additionally, we need to add

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

to pom.xml**.**

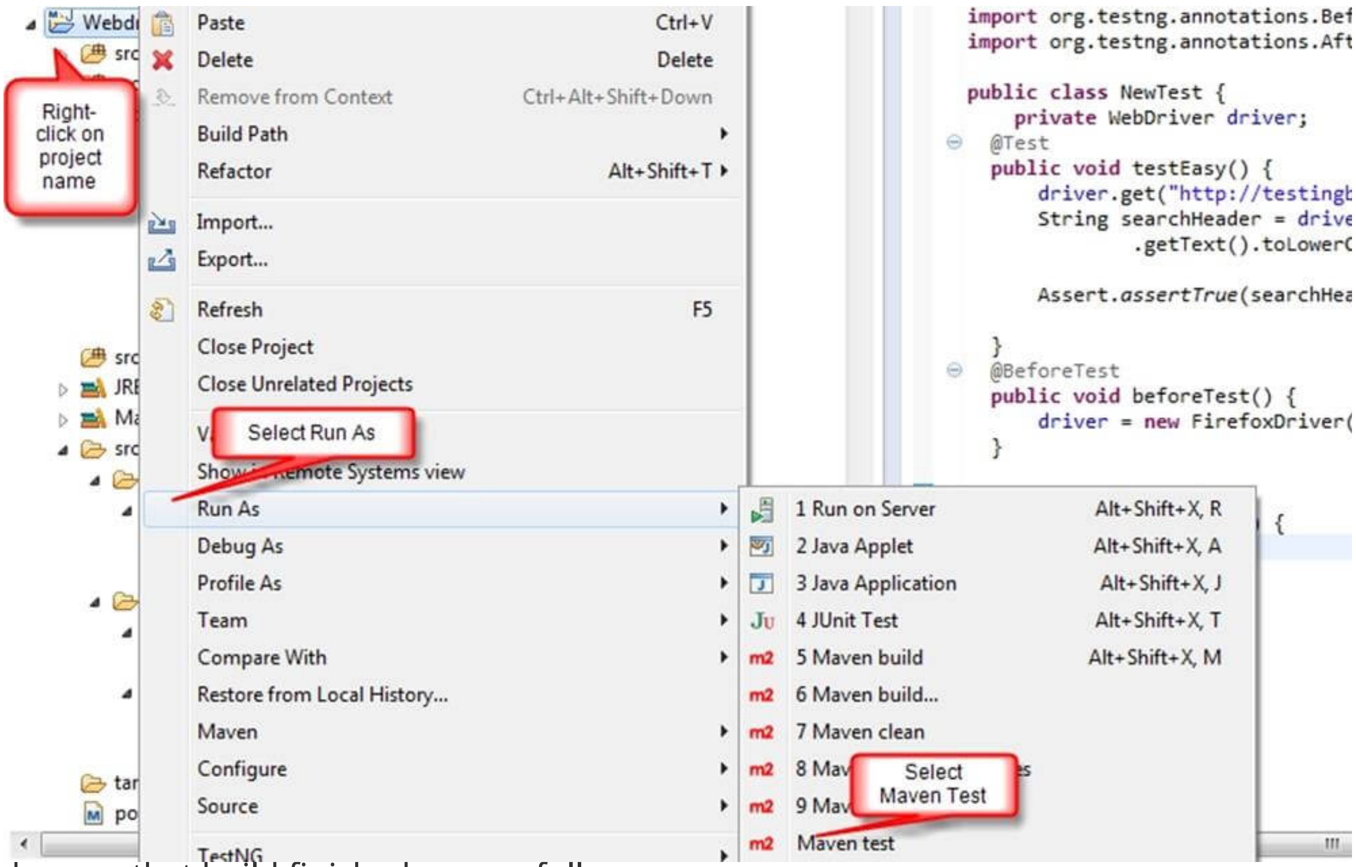
The maven-surefire-plugin is used to configure and execute tests. Here plugin is used to co testing.xml for TestNG test and generate test reports.

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



```
    <plugins>
      <plugin>
      <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.3.2</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</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>
```

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


Right-click on project name
Select Run As
Select Maven Test
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Show in Remote Systems view
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
Maven
Configure
Source
TestNG
1 Run on Server Alt+Shift+X, R
2 Java Applet Alt+Shift+X, A
3 Java Application Alt+Shift+X, J
4 JUnit Test Alt+Shift+X, T
5 Maven build Alt+Shift+X, M
6 Maven build...
7 Maven clean
Maven test

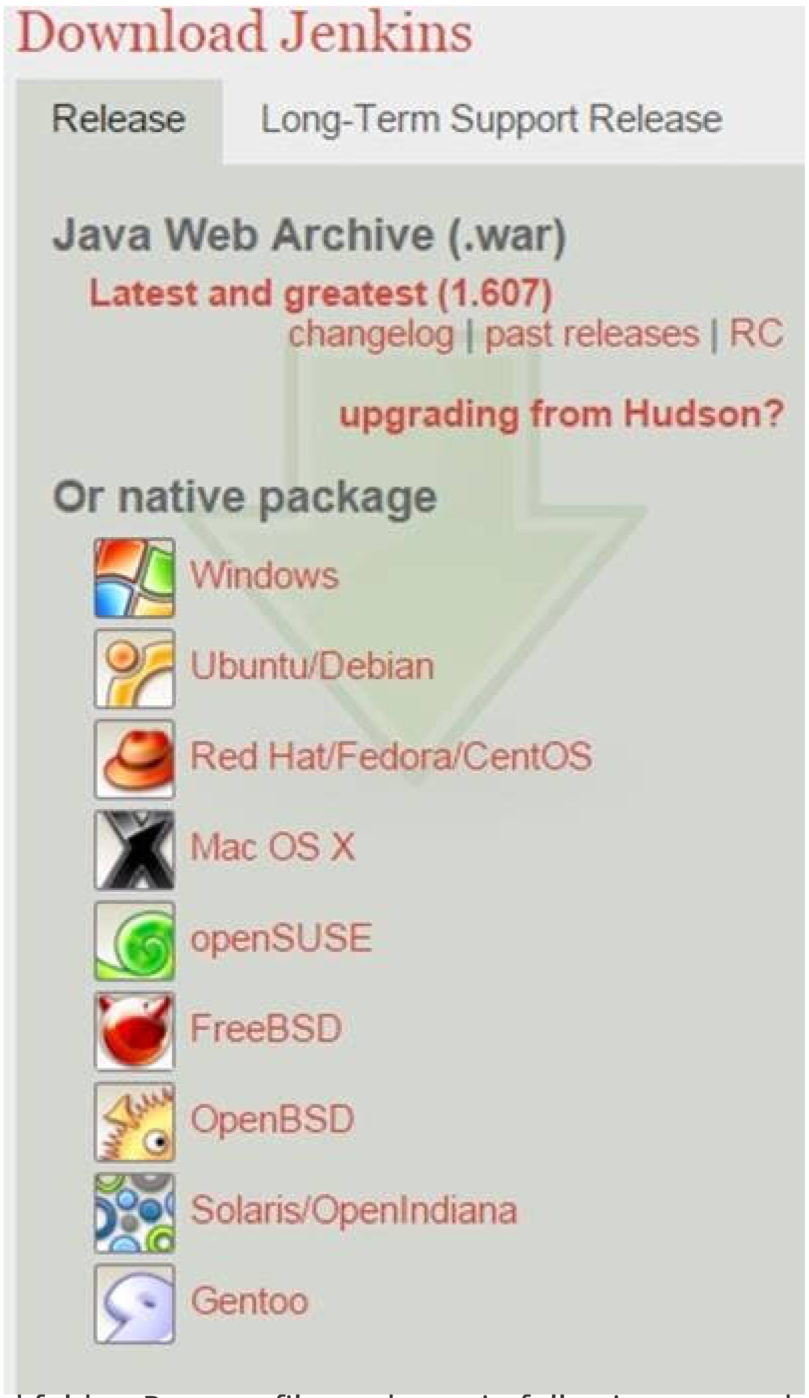
```
import org.testng.annotations.Bef
import org.testng.annotations.Aft

public class NewTest {
    private WebDriver driver;
  @Test
  public void testEasy() {
      driver.get("http://testingb
      String searchHeader = drive
              .getText().toLowerC

      Assert.assertTrue(searchHea

  }
  @BeforeTest
  public void beforeTest() {
      driver = new FirefoxDriver(
  }
```

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 Jenkin


Download Jenkins

Release | Long-Term Support Release

Java Web Archive (.war)

Latest and greatest (1.607)

changelog | past releases | RC

upgrading from Hudson?

Or native package

Windows

Ubuntu/Debian

Red Hat/Fedora/CentOS

Mac OS X

openSUSE

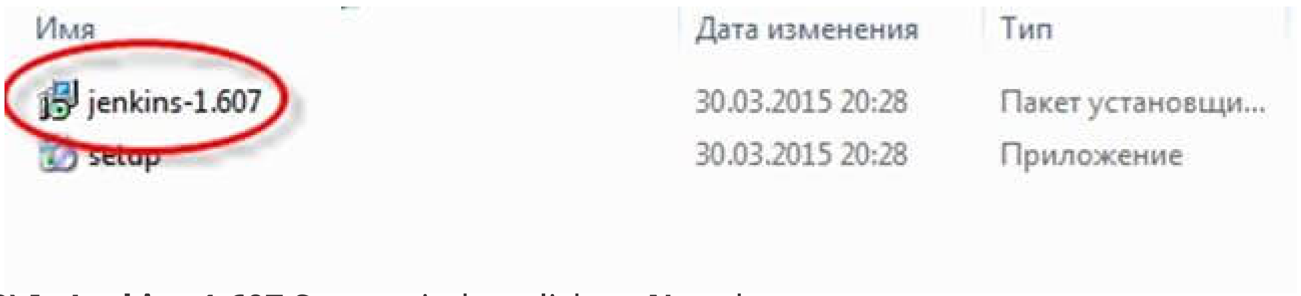
FreeBSD

OpenBSD

Solaris/OpenIndiana

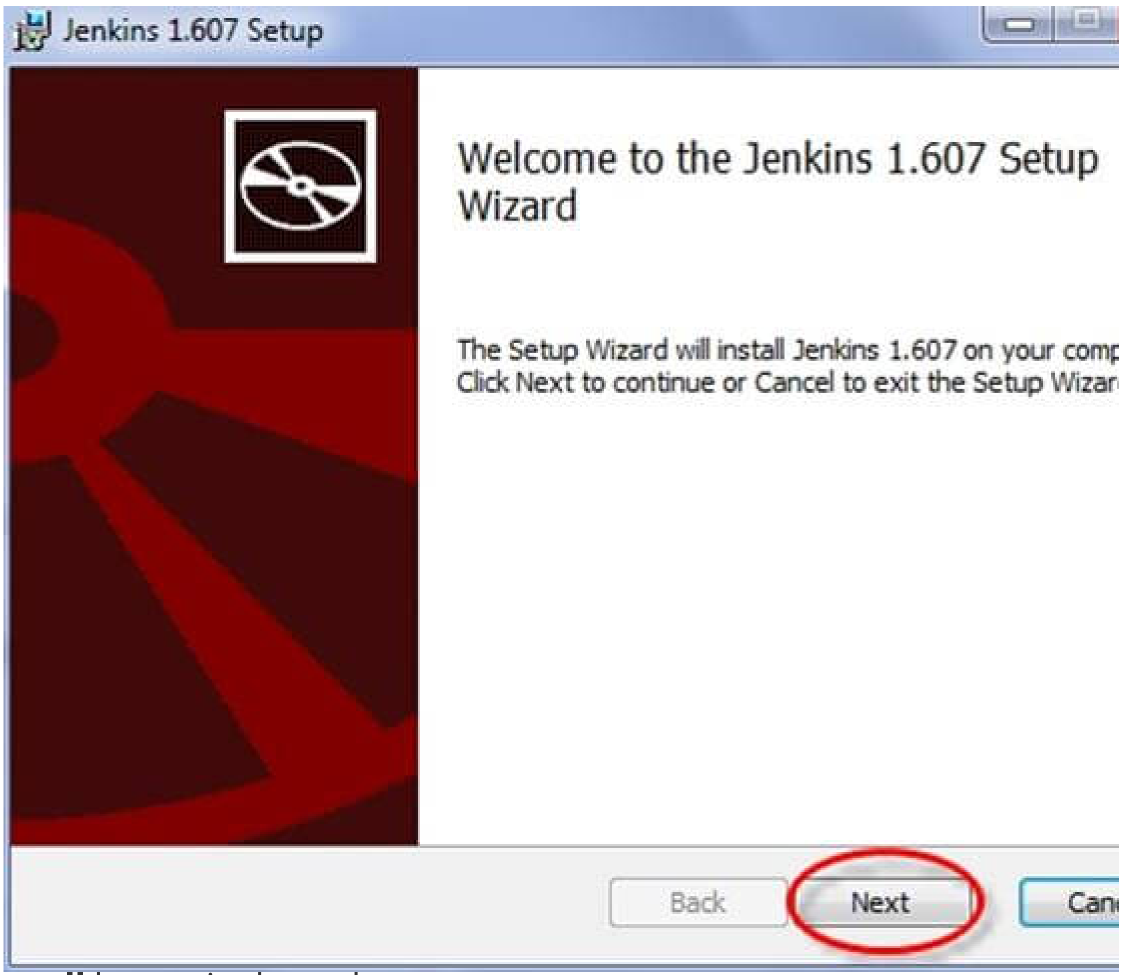
Gentoo

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



Имя	Дата изменения	Тип
jenkins-1.607	30.03.2015 20:28	Пакет установщи...
setup	30.03.2015 20:28	Приложение

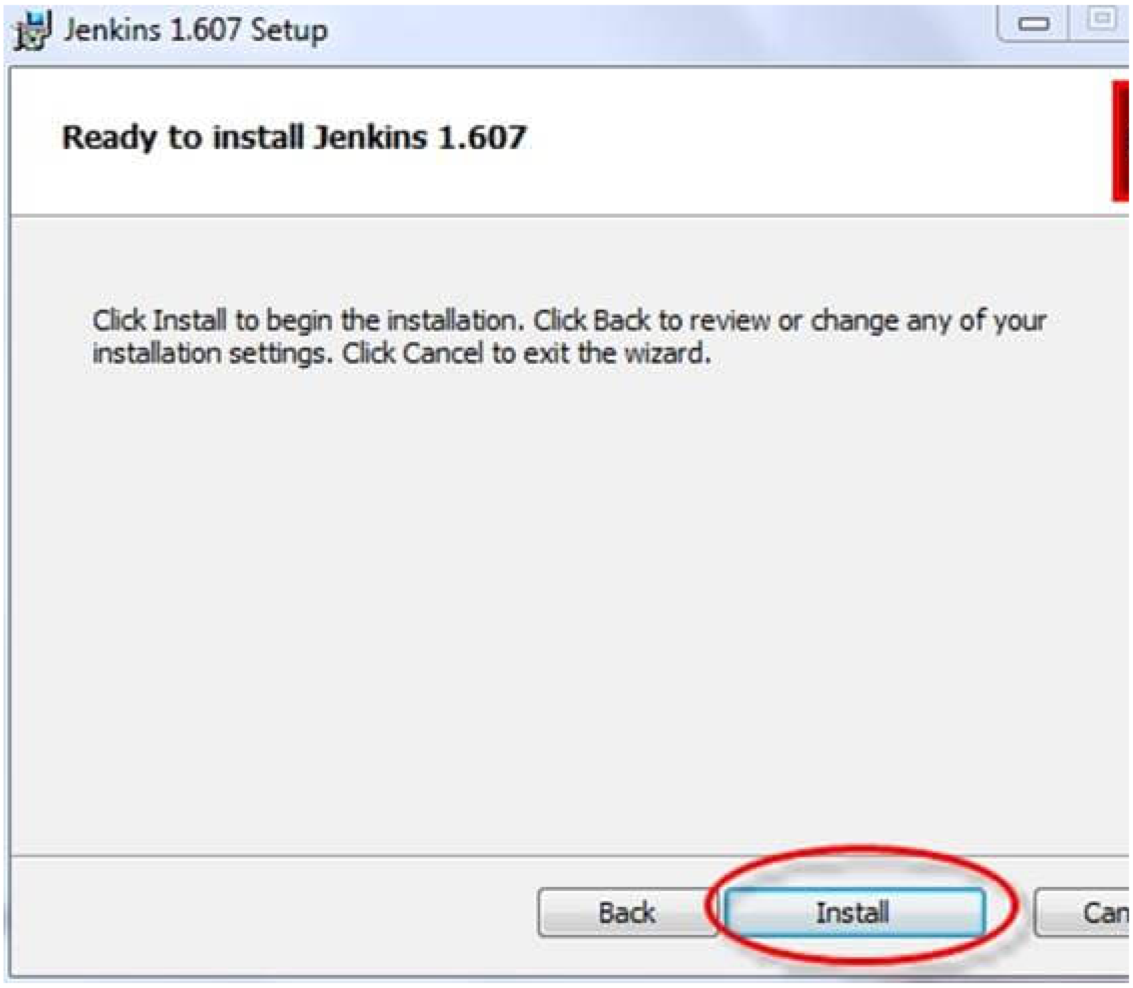
Step 3) In **Jenkins 1.607 Setup** window click on **Next** button.



Welcome to the Jenkins 1.607 Setup Wizard

The Setup Wizard will install Jenkins 1.607 on your comp
Click Next to continue or Cancel to exit the Setup Wizar

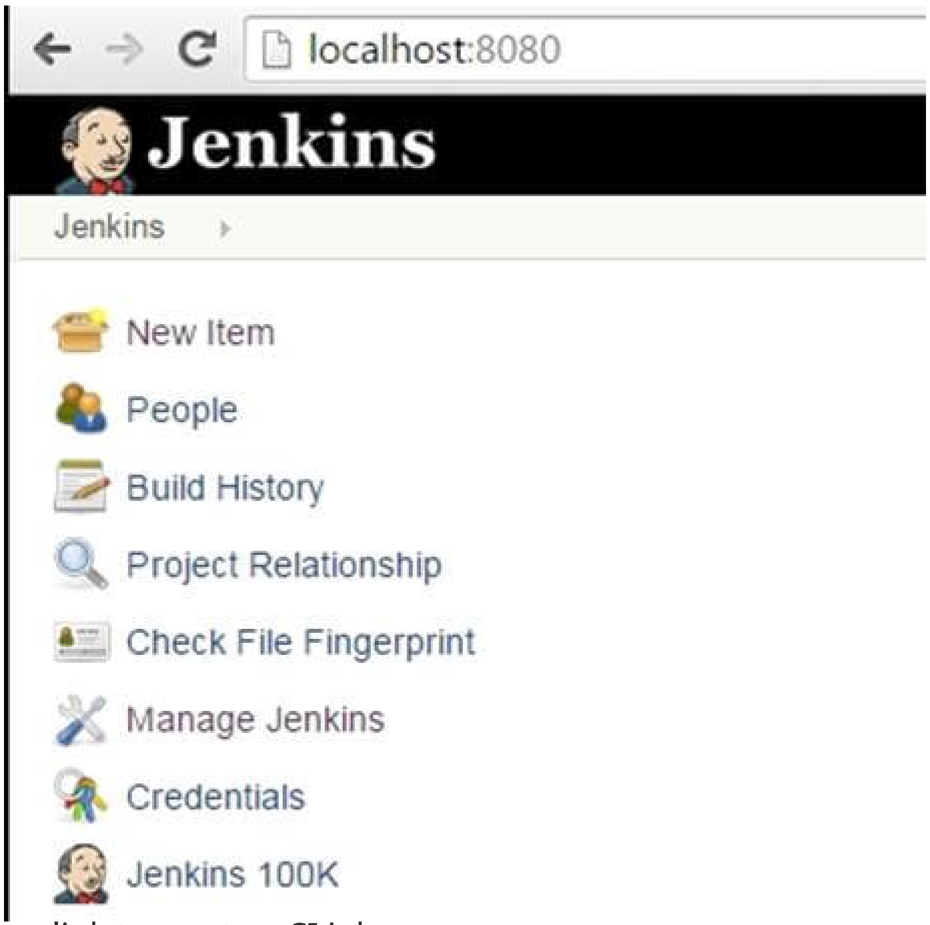
Step 4) Click on **Install** button in the end.



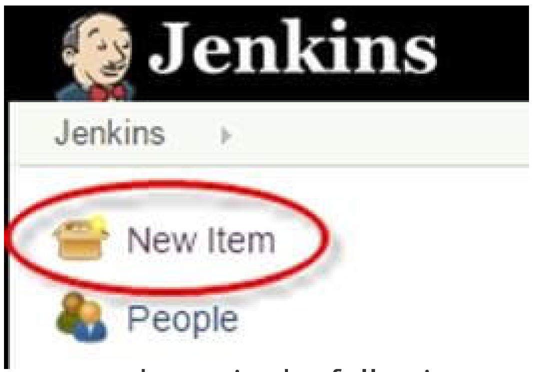
Ready to install Jenkins 1.607

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

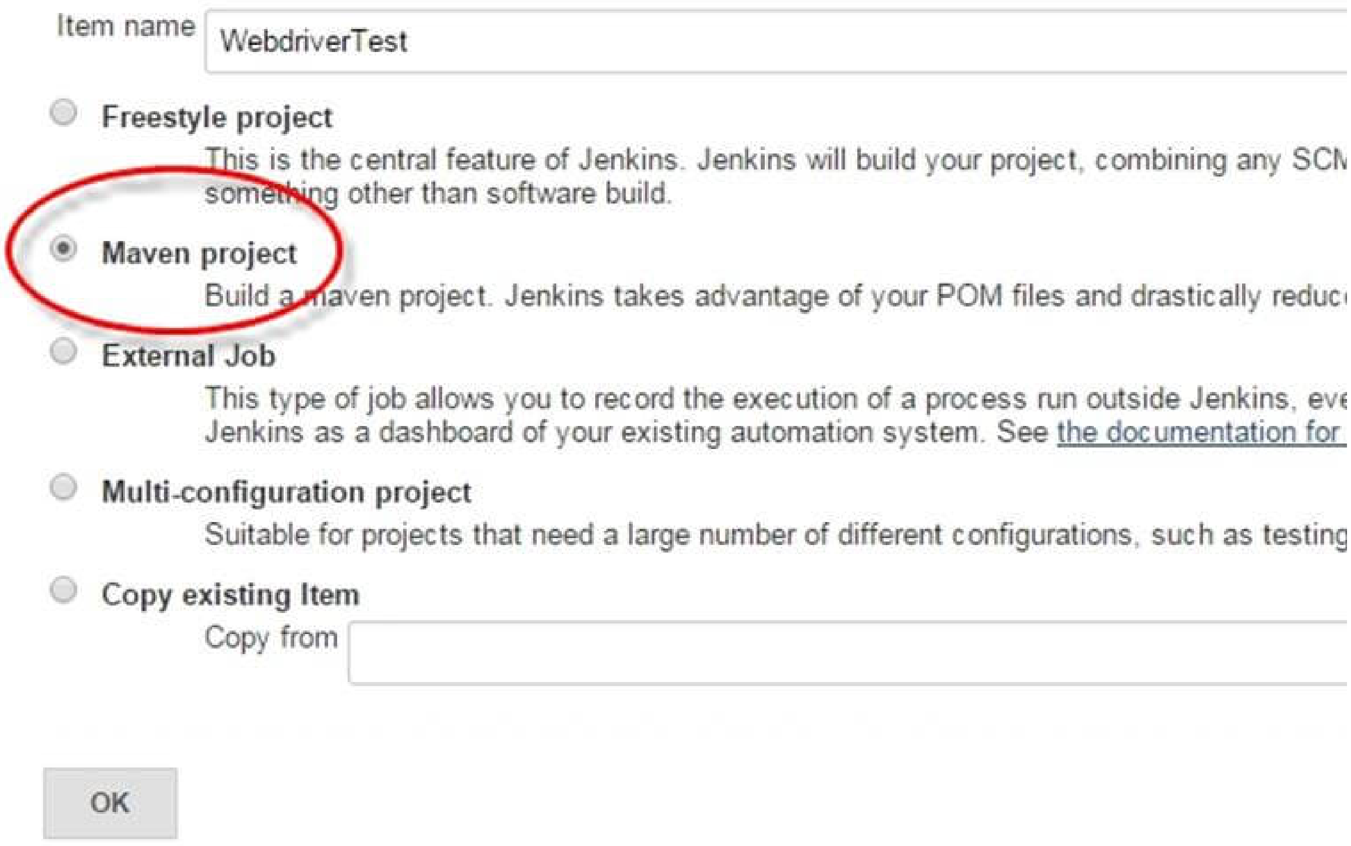
Step 5) Once installation is done, navigate to the Jenkins Dashboard (http://localhost:8080
browser window.



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



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



Item name WebdriverTest

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM
something other than software build.

[✓ Maven project]
Build a maven project. Jenkins takes advantage of your POM files and drastically reduc

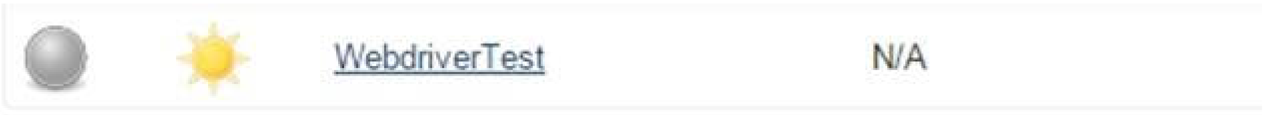
External Job
This type of job allows you to record the execution of a process run outside Jenkins, eve
Jenkins as a dashboard of your existing automation system. See the documentation for

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing

Copy existing Item
Copy from

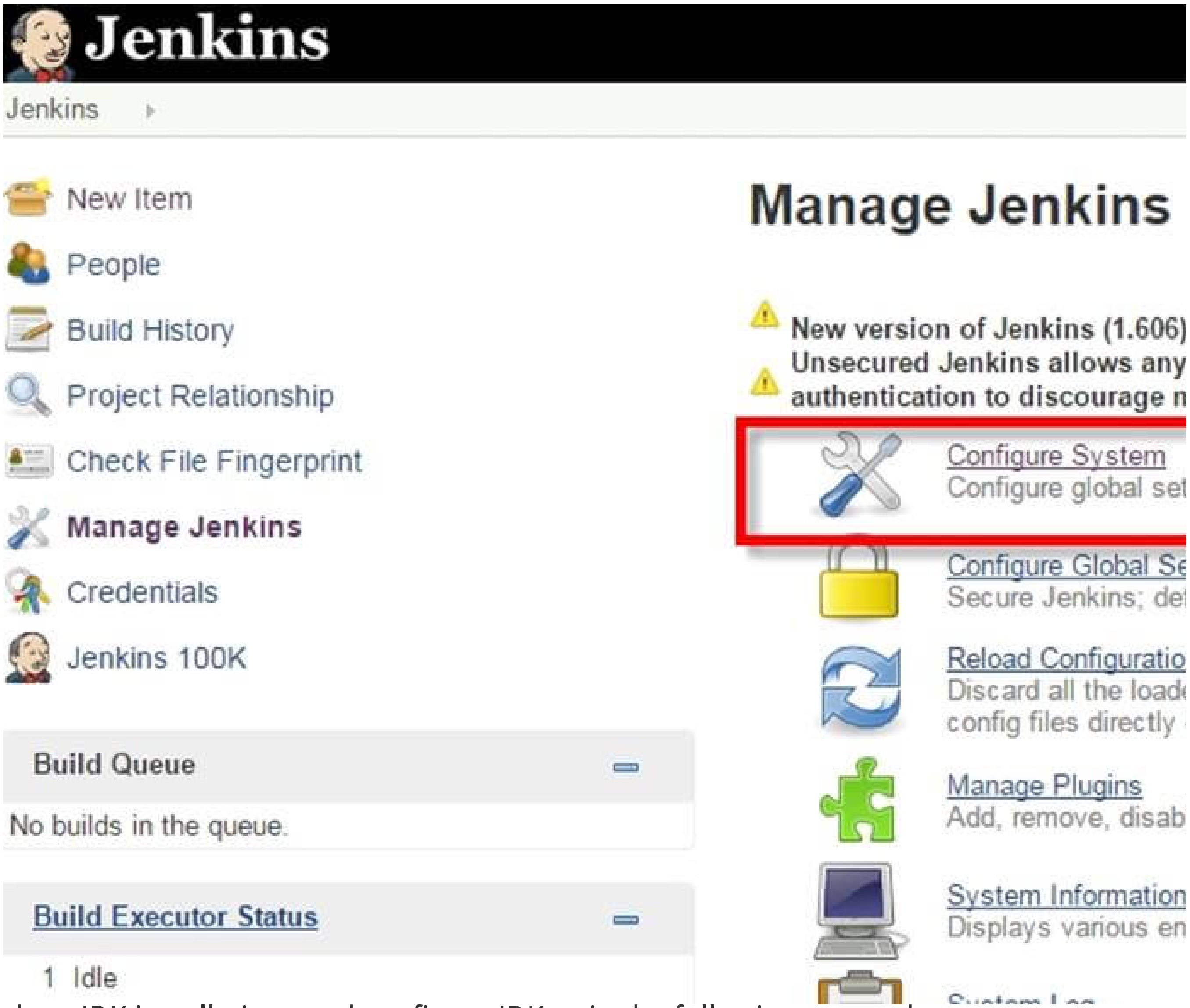
Using the Build a **Maven Project** option, Jenkins supports building and testing Maven proje

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

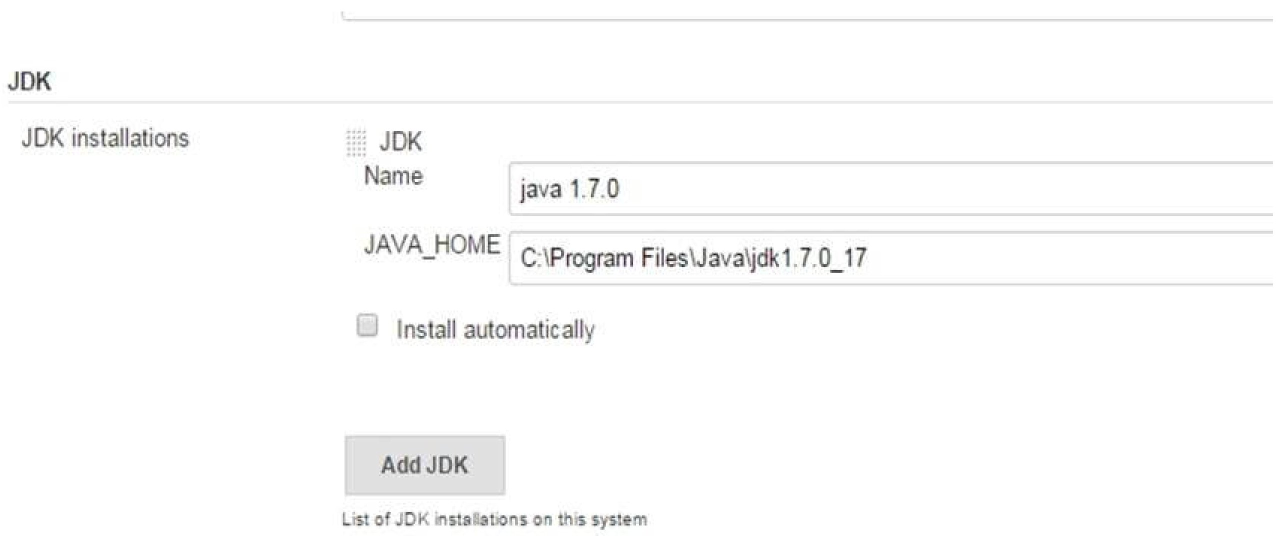


WebdriverTest N/A

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



Click on JDK installations and configure JDK as in the following screenshot:


JDK

JDK installations

JDK

Name: java 1.7.0

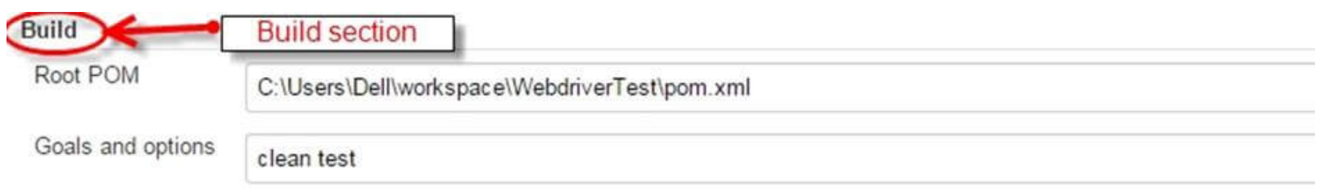
JAVA_HOME: C:\Program Files\Java\jdk1.7.0_17

☐ Install automatically

List of JDK installations on this system

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

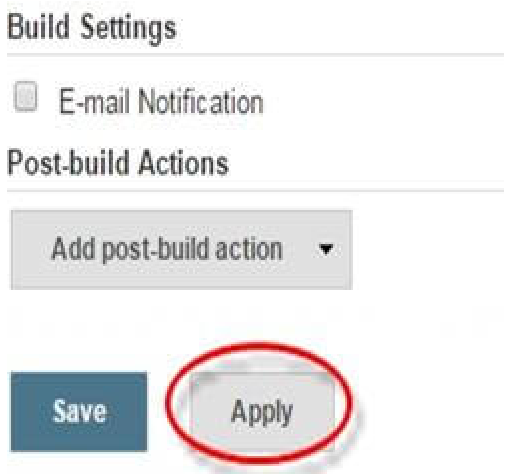
- In the **Root POM** textbox, enter full path to pom.xml
- In Goals and options section, enter "clean test"



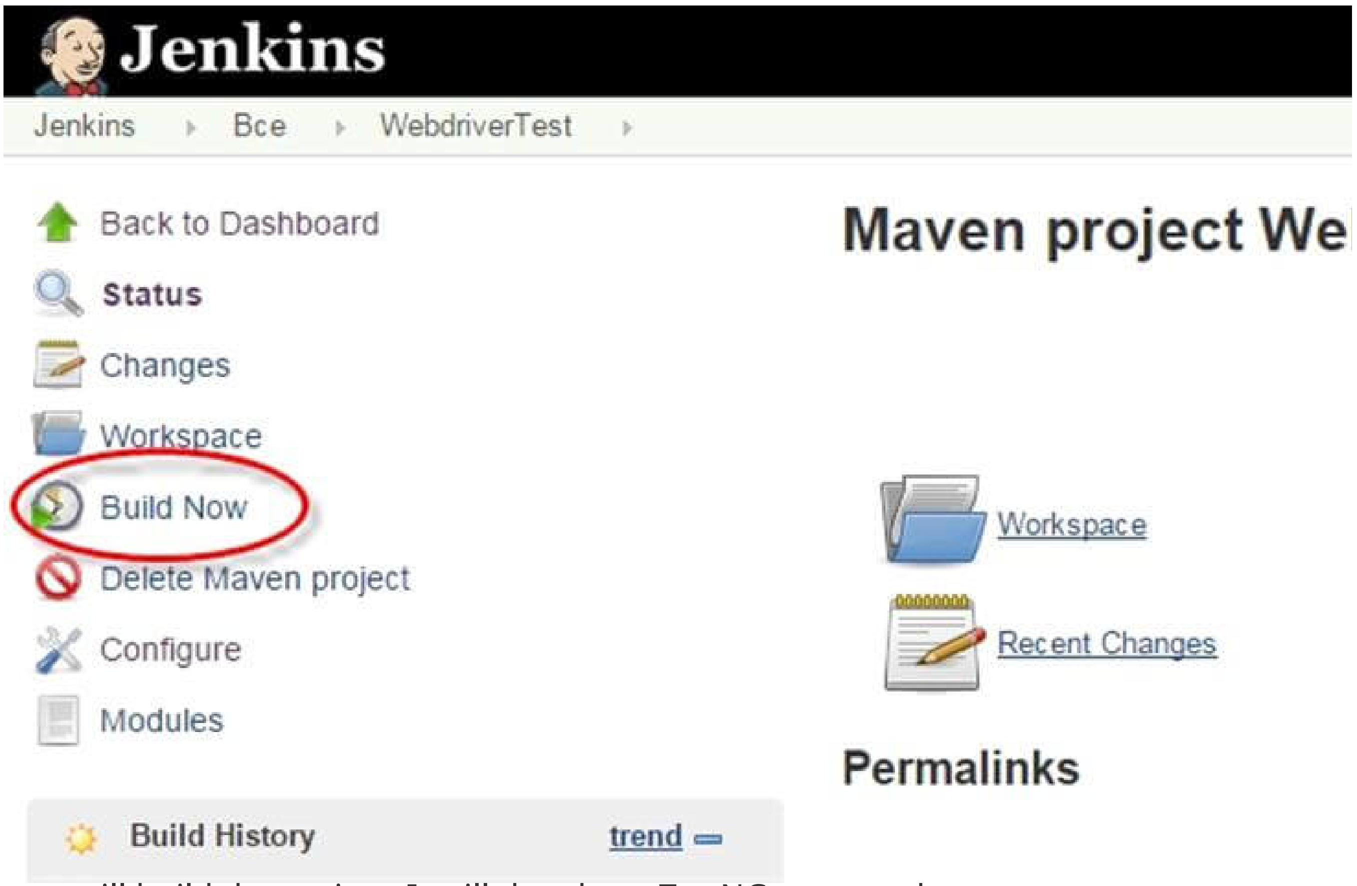
Root POM: C:\Users\Dell\workspace\WebdriverTest\pom.xml

Goals and options: clean test

Step 9) Click on **Apply** button.

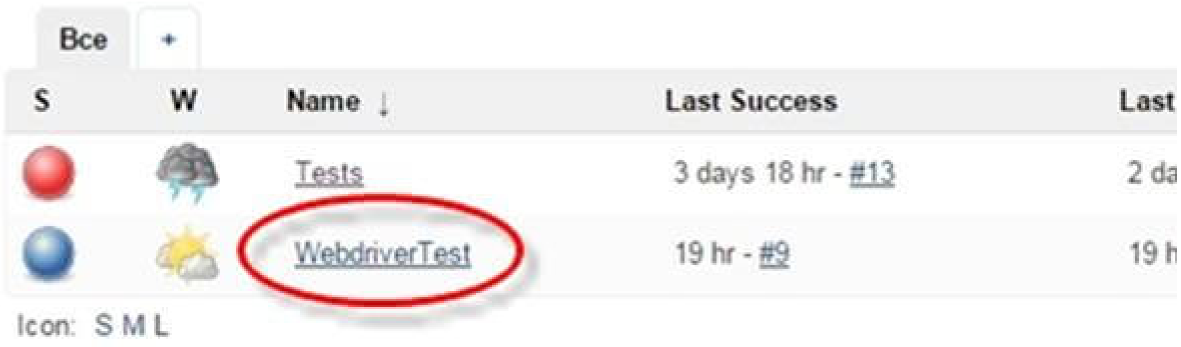


Step 10) On the WebdriverTest project page, click on the **Build Now** link.



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

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

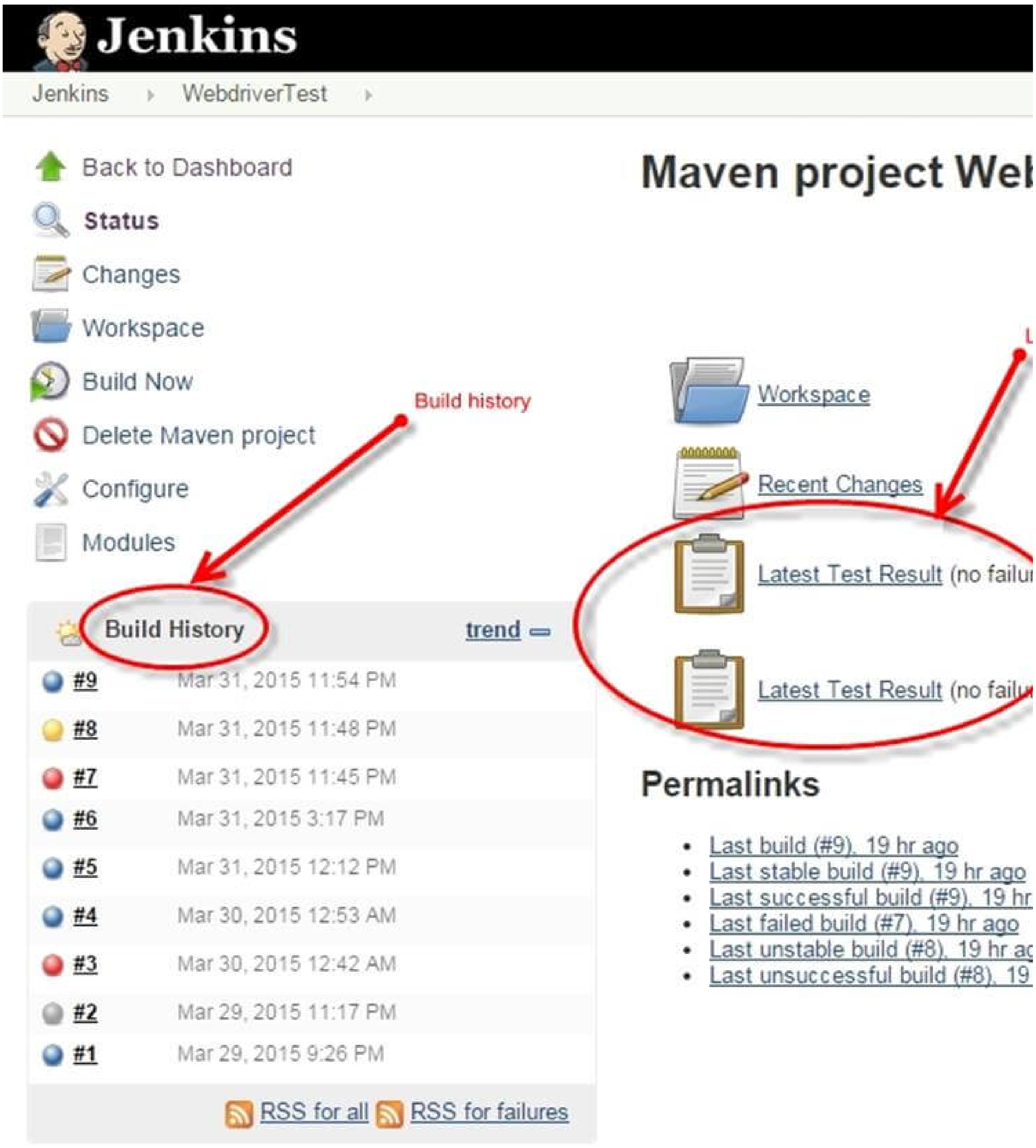


Все +

S	W	Name ↓	Last Success	Last
●		Tests	3 days 18 hr - #13	2 da
●		WebdriverTest	19 hr - #9	19 h

Icon: S M L

Step 12) The WebdriverTest project page displays the build history and links to the results a following screenshot:



Build History trend

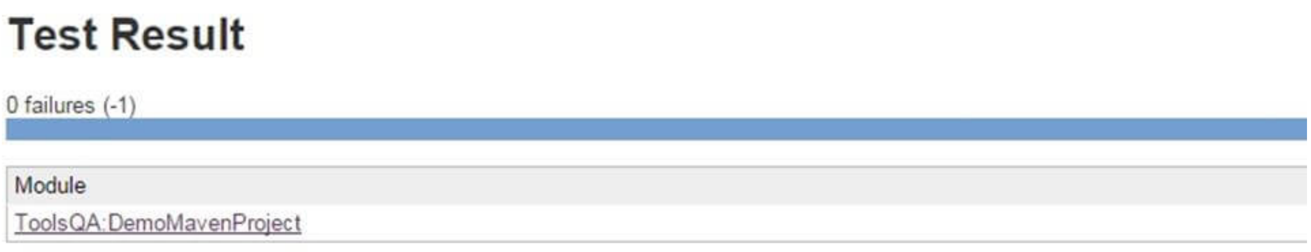
#9	Mar 31, 2015 11:54 PM
#8	Mar 31, 2015 11:48 PM
#7	Mar 31, 2015 11:45 PM
#6	Mar 31, 2015 3:17 PM
#5	Mar 31, 2015 12:12 PM
#4	Mar 30, 2015 12:53 AM
#3	Mar 30, 2015 12:42 AM
#2	Mar 29, 2015 11:17 PM
#1	Mar 29, 2015 9:26 PM

RSS for all RSS for failures

Permalinks

- Last build (#9), 19 hr ago
- Last stable build (#9), 19 hr ago
- Last successful build (#9), 19 hr
- Last failed build (#7), 19 hr ago
- Last unstable build (#8), 19 hr ag
- Last unsuccessful build (#8), 19

Step 13) Click on the "Latest Test Result" link to view the test results as shown in the follow

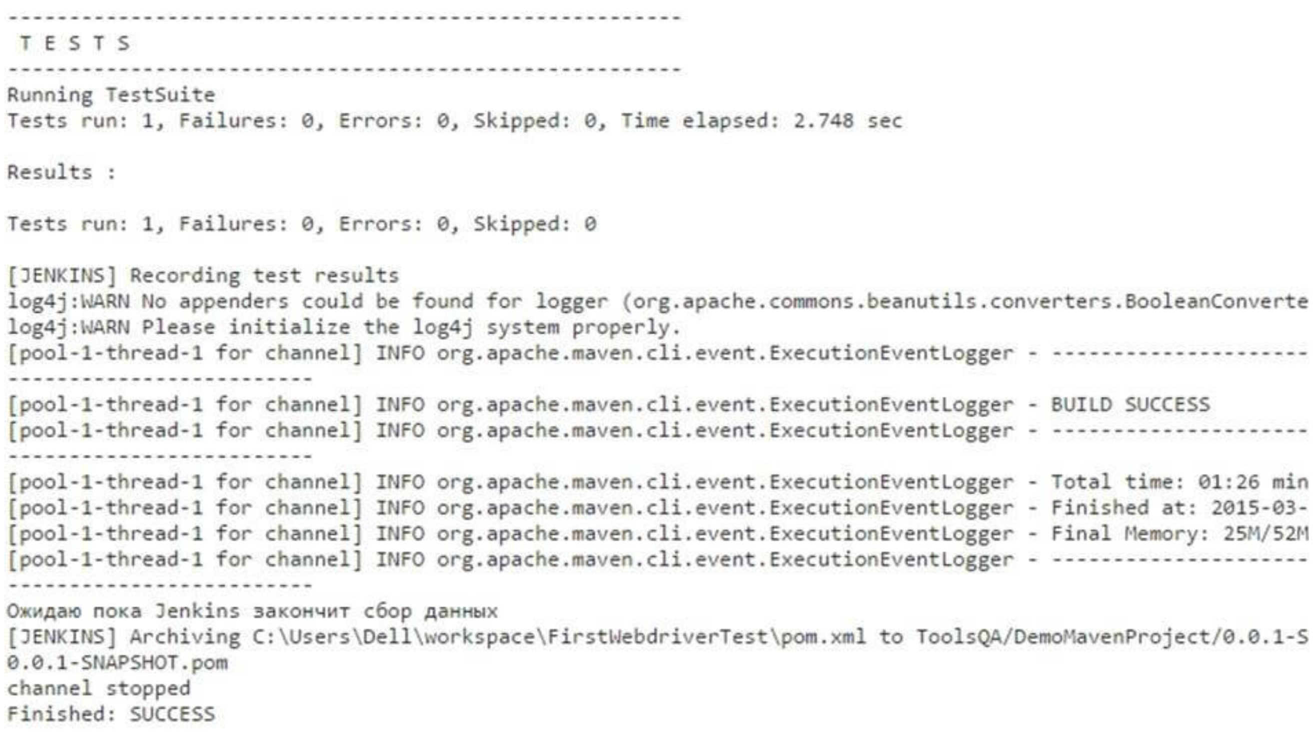


Test Result

0 failures (-1)

Module
ToolsQA:DemoMavenProject

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

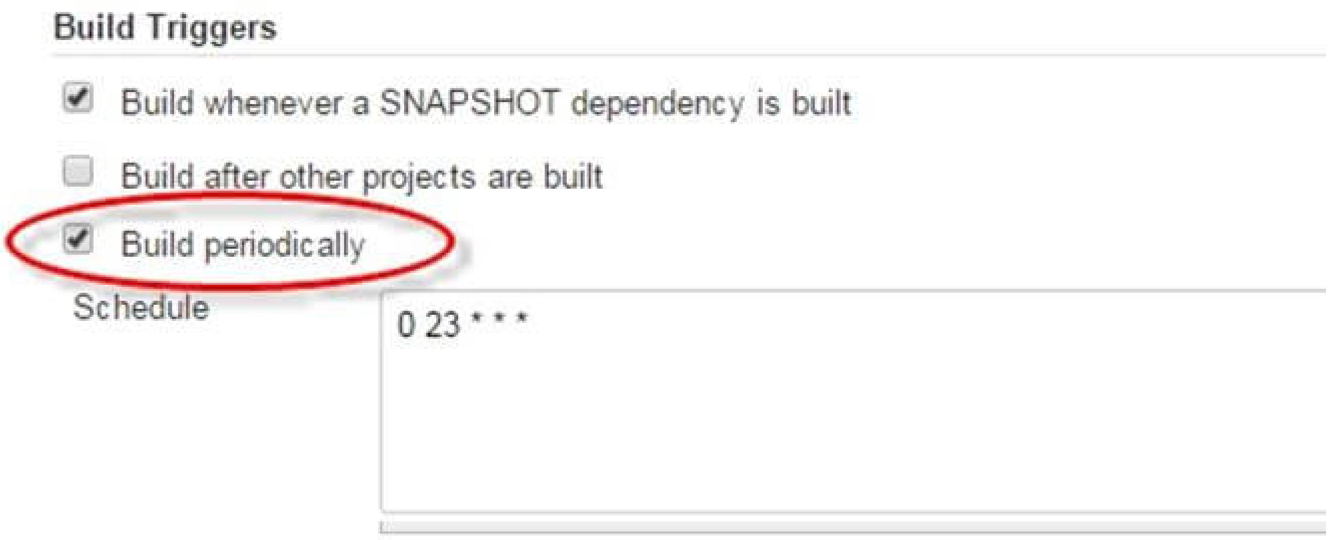


```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running TestSuite
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.748 sec

Results :

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

[JENKINS] Recording test results
log4j:WARN No appenders could be found for logger (org.apache.commons.beanutils.converters.BooleanConverte
log4j:WARN Please initialize the log4j system properly.
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - --------------------
------------------------
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - BUILD SUCCESS
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - --------------------
------------------------
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - Total time: 01:26 min
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - Finished at: 2015-03-
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - Final Memory: 25M/52M
[pool-1-thread-1 for channel] INFO org.apache.maven.cli.event.ExecutionEventLogger - --------------------
------------------------
Ожидаю пока Jenkins закончит сбор данных
[JENKINS] Archiving C:\Users\Dell\workspace\FirstWebdriverTest\pom.xml to ToolsQA/DemoMavenProject/0.0.1-S
0.0.1-SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```

Scheduling Jenkins for automatic execution.

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

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

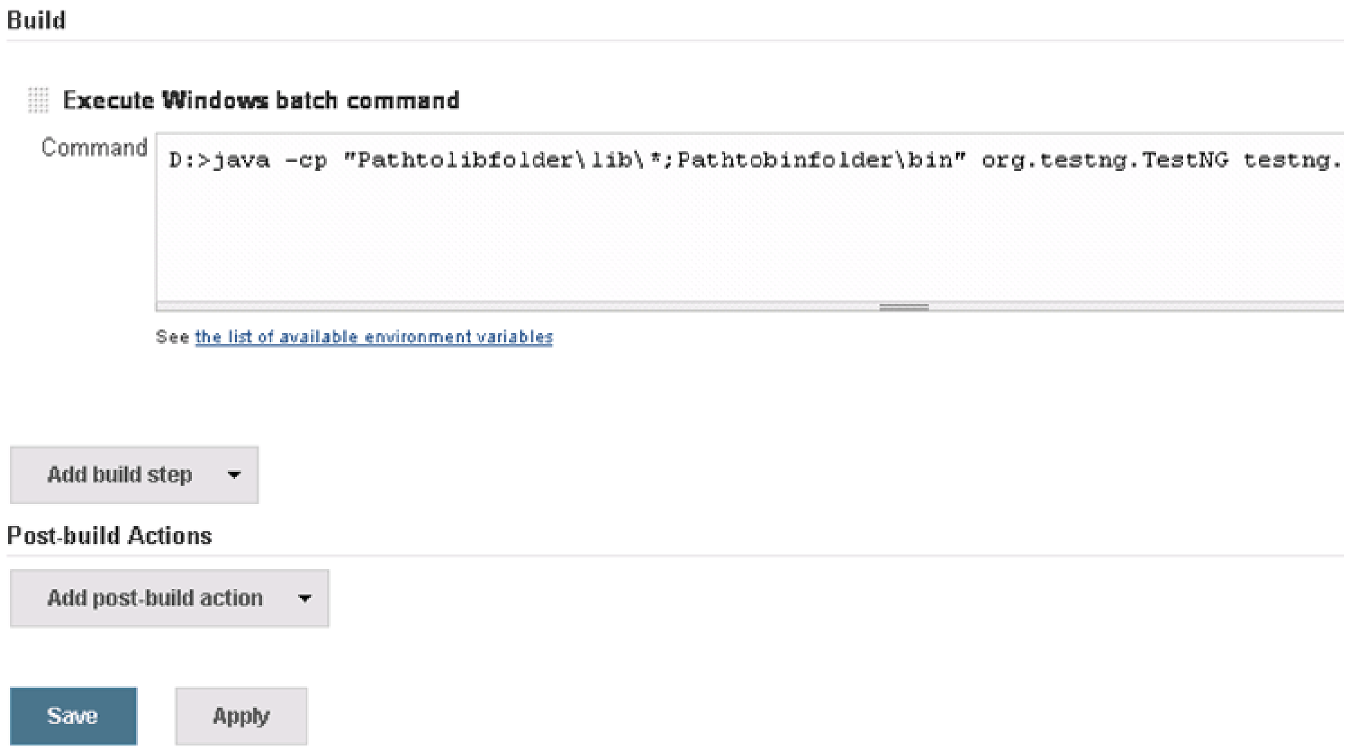

Build Triggers
Build whenever a SNAPSHOT dependency is built
Build after other projects are built
Build periodically
Schedule

```
0 23 * * *
```

Using Jenkings without Maven

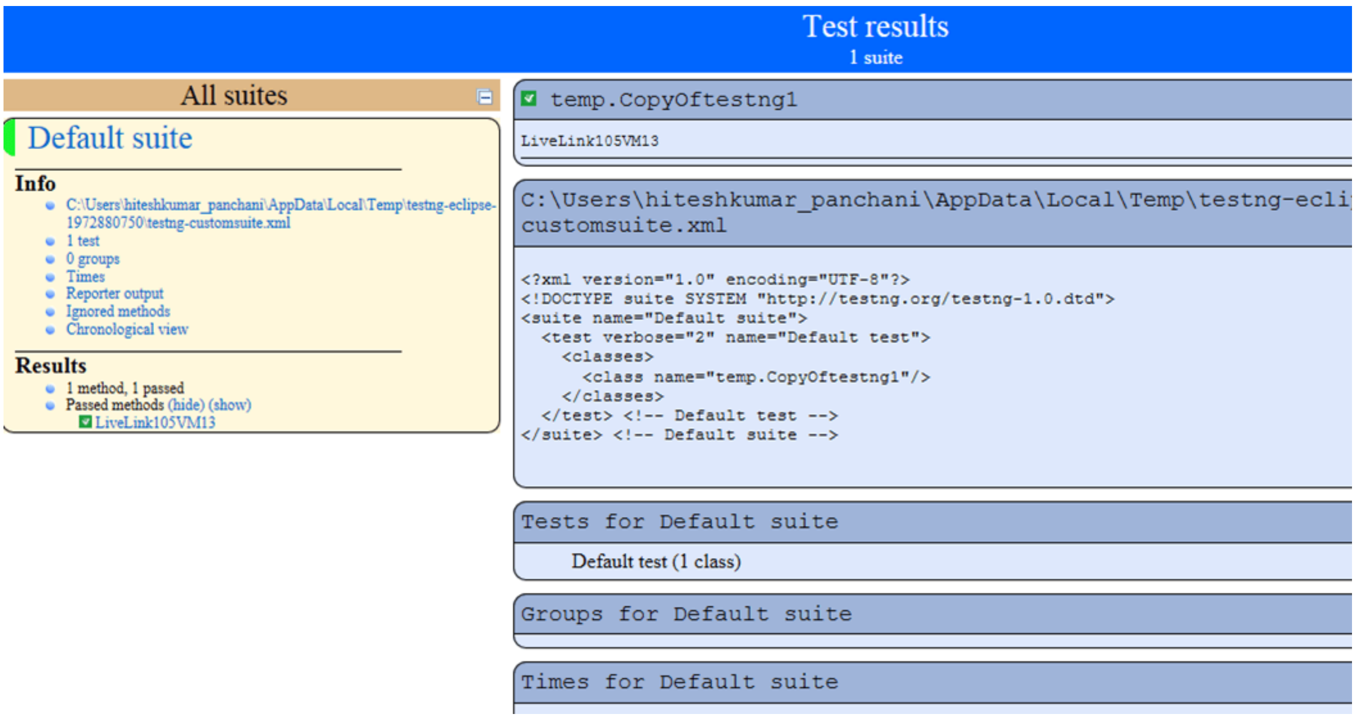
To run pure TestNg script in Jenkins, enter the following in build

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


Build
Execute Windows batch command
Command
See the list of available environment variables
Add build step
Post-build Actions
Add post-build action
Save
Apply

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

- 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 comma TestNG.
- Result will be stored in custom report HTML file that can be sent via email with a Jenkin c
- Output of the code will be



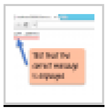
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Default suite">
  <test verbose="2" name="Default test">
    <classes>
      <class name="temp.CopyOftestng1"/>
    </classes>
  </test> <!-- Default test -->
</suite> <!-- Default suite -->
```

Benefits of using Jenkins

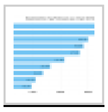
1. Early issue finding – Bug can be detected in early phase of the software development
2. Automatic integration – no separate effort required to integrate all changes
3. Installer – a deployable system available at any point of development
4. Records – part build records maintained
5. Support and Plugins: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

You Might Like

ASP.Net Tutorial


How to Create and Run Asp.Net Unit Testing Project

Install and First C++ Hello World Program


Introduction to C++

Prev

8 Comments Guru99

Recommend 1 Share

Join the discussion

Join the discussion…

Mangesh W • 4 months ago

Hi,
thanks for article, i am new in Automation what is next step after creating build file means hc
build Snapshot file . and where?

• Reply • Share ›

jaysan ias • 9 months ago

Hi -

I followed this tutorial step by step and when I reached the Root POM textbox under configu
an error message. When I entered the full path of my pom.xml, it gives me the following errc

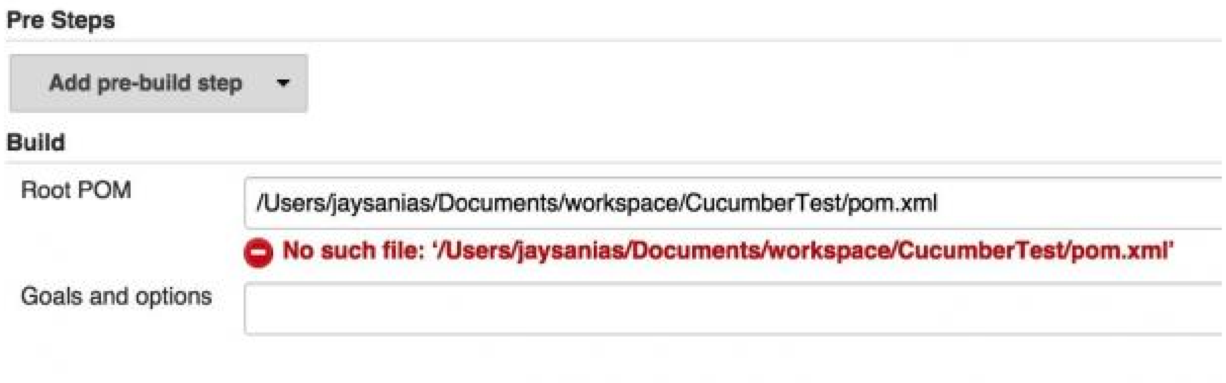
No such file: '/Users/jaysanias/Documents/workspace/CucumberTest/pom.xml' (image attac

I know the above path is correct and there is that pom.xml file in this path but I am sure wha
kind of stuck here for more than a day.

Any help or advise to fix this issue would be greatly appreciated.

P.S. I am running jenkins in our local host and working in Mac

-Jaysan


Pre Steps

Add pre-build step

Build

Root POM: /Users/jaysanias/Documents/workspace/CucumberTest/pom.xml

No such file: '/Users/jaysanias/Documents/workspace/CucumberTest/pom.xml'

Goals and options

• Reply • Share ›

Venkat Rao • 10 months ago

Hi -

I am currently writing an automation framework, where I work has several environments viz.
Environment, Staging and Live. And understandably we have different base urls, user name
environments. I plan to automate the daily tests on different environments using Jenkins. At
hardcoded all the variables in the step definition files, committed in the repo and run it on Je

What I am thinking at this stage is that I should be able to pass the environment I want to tes
Jenkins and based on this a correct configuration with appropriate variables (urls, user name
chosen which is in turn used by the step definition files for execution.

I did some basic look up in google and I am not sure if I am searching for correct things base

I am not looking for shortcuts here, I am wiling to spend the time it takes to learn things by re
examples.

Can someone point me in the right direction as to how to approach this?

Your help is much appreciated, thanks a ton in advance for your time.

• Reply • Share ›

Vall • a year ago

Hi, Could you please tell me how to specify specific tests to run in Pom.xml? I am sure it has attribute, but an example would be nice. Thanks!!

• Reply • Share ›

Ankit Vall • 2 months ago

Hi Val,

You can configure your testng.xml file and include the specific methods inside your te testng.xml inside your pom.xml as <suitexmlfile> tag and that will do the rest of the w

• Reply • Share ›

Vall Ankit • 2 months ago

Thank You. That works!!

• Reply • Share ›

nguoianphu • a year ago

Could we create .jar executable file to run without Maven? I added maven-jar-plugin into por
# java -jar test.jar
Error: Exception in thread "main" java.lang.NoClassDefFoundError: org/testng

• Reply • Share ›

avishek behera • a year ago

A very nice tutorial and explained in a very clear manner...
Found a spelling mistake above in the line
Using Jenkings without Maven- It is supposed to be jenkins instead of Jenkings. May be bei not resist to share it. Hope it makes sense.. :)

• Reply • Share ›

ALSO ON **GURU99**

Installation and Configuration of HIVE and MYSQL

2 comments • 6 months ago

Avat **VIneet kumar singh** — after hive is installed when i m running cmnd create database; it show error communication link failure

Cassandra Architecture

1 comment • 4 months ago

Avat **Amol Shinde** — What a neat D this doc God bless you

Coded UI Test (CUIT)

1 comment • 24 days ago

Avat **Gaurav Sikka** — Any good link to learn coded ui. need to learn from scratch having good knowledge of C#...kindly suggest.

Using Robot API with Seleniu

3 comments • 8 months ago

Avat **Bibhishan** — If some one will b program given above, this prog seems like XPATH has change

Subscribe   Add Disqus to your site Add Disqus Add   Privacy

About

About us
Advertise with Us
Jobs
Privacy Policy

Contact Us

Contact us
FAQ
Write For Us

Android App

GET IT ON Google play

Certificat

ISTQB Cer
MySQL Ce
QTP Certif
Testing Ce
CTAL Exam