TestNG

TestNG stands for "**Test Next Generation**".

TestNG is a unit testing framework [contains readymade libraries] used to perform the unit testing.

What is unit testing?

- UNIT testing is defined as a type of software testing where individual units/ components of a software are tested.
- Unit testing of software applications is done during the development (coding) of an application. The objective of Unit Testing is to isolate a section of code and verify its correctness. In procedural programming, a unit may be an individual function or procedure.
- Unit testing is usually performed by the developer.

Note:

- Basically "TestNG" is used to perform unit testing, that is for every customer requirement develop develops business class, before giving the build to testing team, developers need to ensure that the source code is working properly. For this he will do unit testing, but testing every business class manually is time consuming because of this reason, developer creates "test class" which will test the respective business class. While developing the test class developer has to perform the following steps.
    1. Call the methods of business class.
    2. Compare expected and actual results.
    3. Generate the report.
    4. Executes all the test classes.
    5. Re-execute failed test classes after debugging etc.
- In order to do above steps developer uses "**TestNG**" because TestNG has readymade features to perform the above mentioned steps and the same steps need to be done in selenium also, hence we use "TestNG" in automation.

Introduction of TestNG

TestNG, where **NG stands for "next generation"** is a test automation framework inspired by JUnit (in Java) and NUnit (in C#).

It can be used for unit, functional, integration, and end-to-end testing

A few of the features that TestNG:

- Extra before and after annotations such as Before/After Suite and Before/After Group.
- Dependency test.

- Grouping of test methods.
- Multithreaded execution.
- In-built reporting framework.

Advantages of TestNG

Advantages offered by TestNG.

1. Multiple Before and After annotation options
2. XML-based test configuration and test suite definition
3. Dependent methods
4. Groups/group of groups
5. Dependent groups
6. Parameterization of test methods
7. Data-driven testing
8. Multithreaded execution
9. Better reporting

Installing TestNG onto Eclipse

Before we can download and start using TestNG, make sure you have Java JDK8 or above is installed on your system. Also make sure that JDK is set in the system path.

Now, let's start with the installation process of TestNG onto Eclipse.

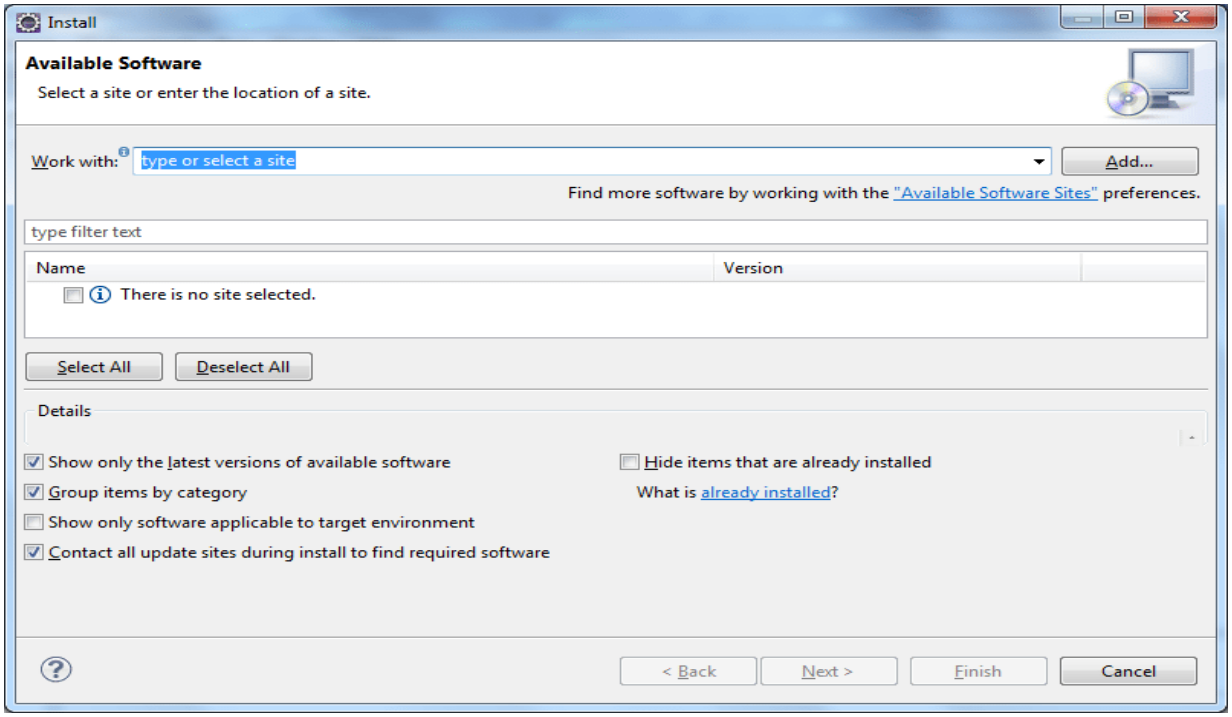
There are two ways,

1. Installing from "Install New Software"
2. Installing from "Eclipse Marketplace"

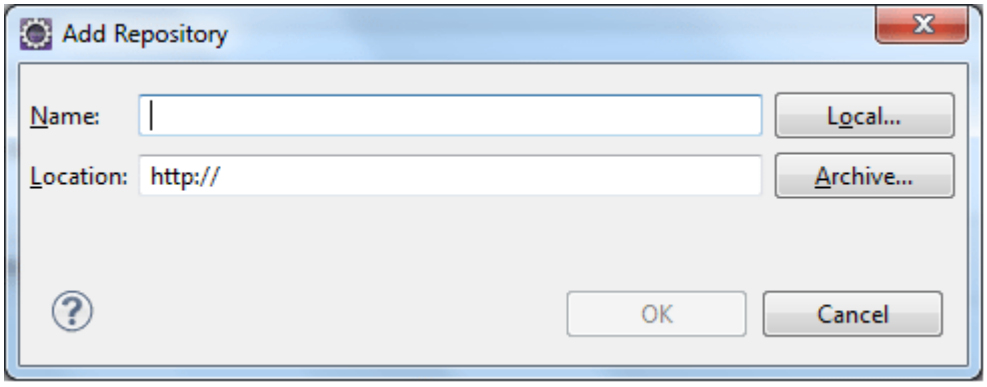
Installing from Eclipse Market Place.

Step 1: Open your Eclipse application.

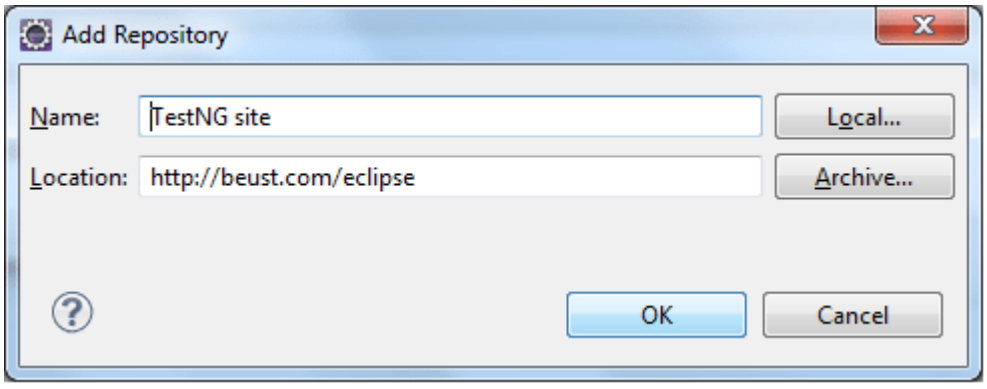
Step 2: Go to **Help | Install New Software**.



Step 3: **Click on the Add…** button next to the Work with text box.

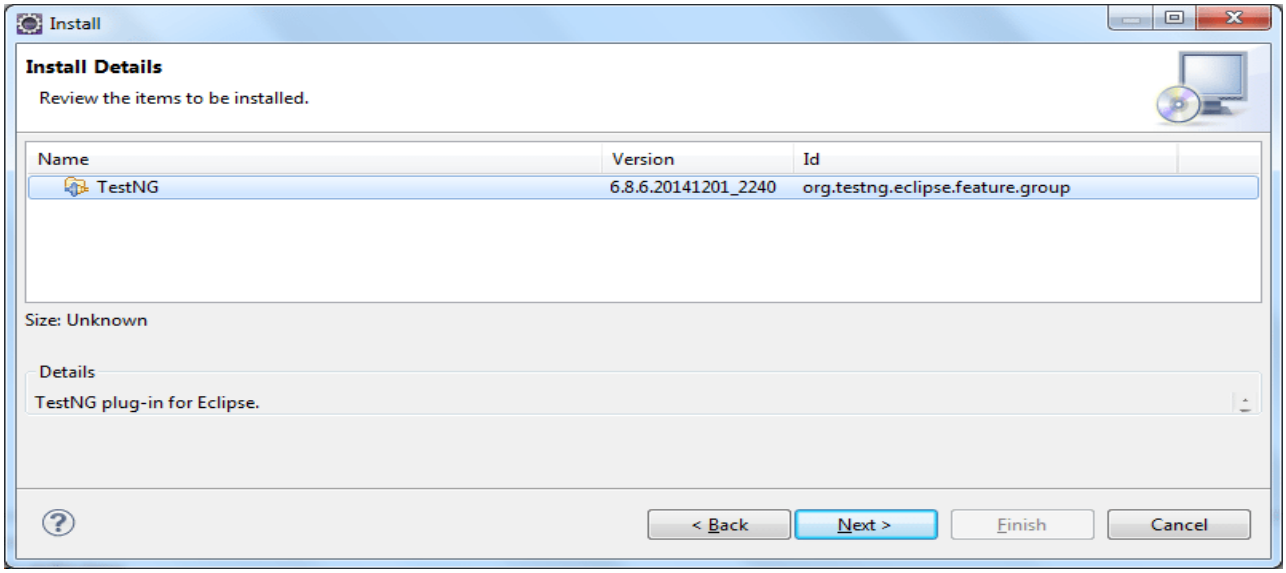


Step 4: **Enter TestNG site into the Name box and enter URL http://beust.com/eclipse into the Location box**. Once done, click on the OK button.



Step 5: On clicking OK, TestNG update site will get added to Eclipse. The available software window will show the tools available to download under the TestNG site.

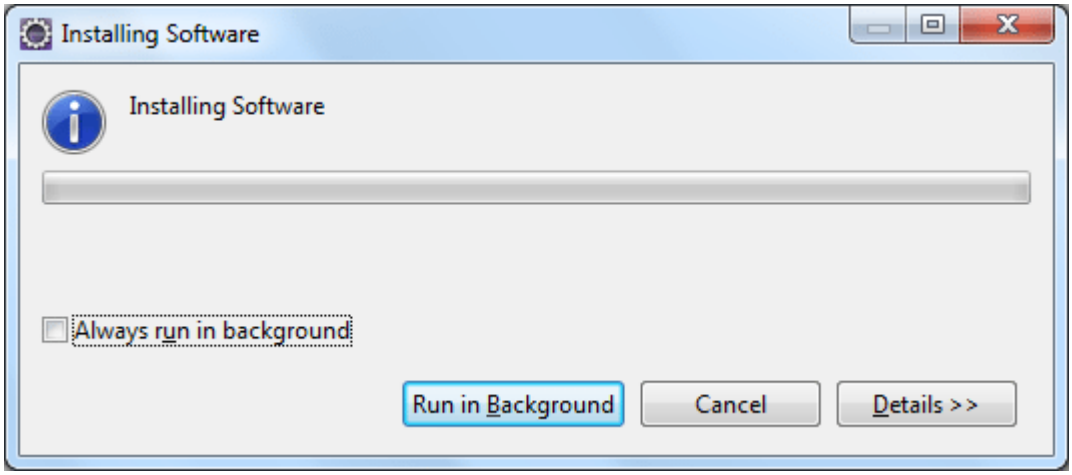
Pramod K S



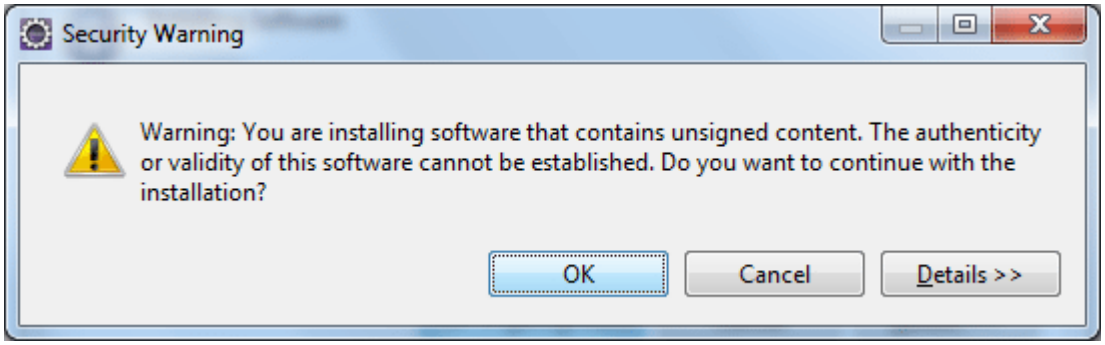
Step 6: **Select TestNG and click on Next**.

Step 7: Eclipse will calculate the software requirements to download the selected TestNG plugin and will show the Install Details screen. **Click on Next on the details screen**.

Step 8: **Accept the License Information and click on Finish**. This will start the download and installation of the TestNG plugin onto Eclipse.

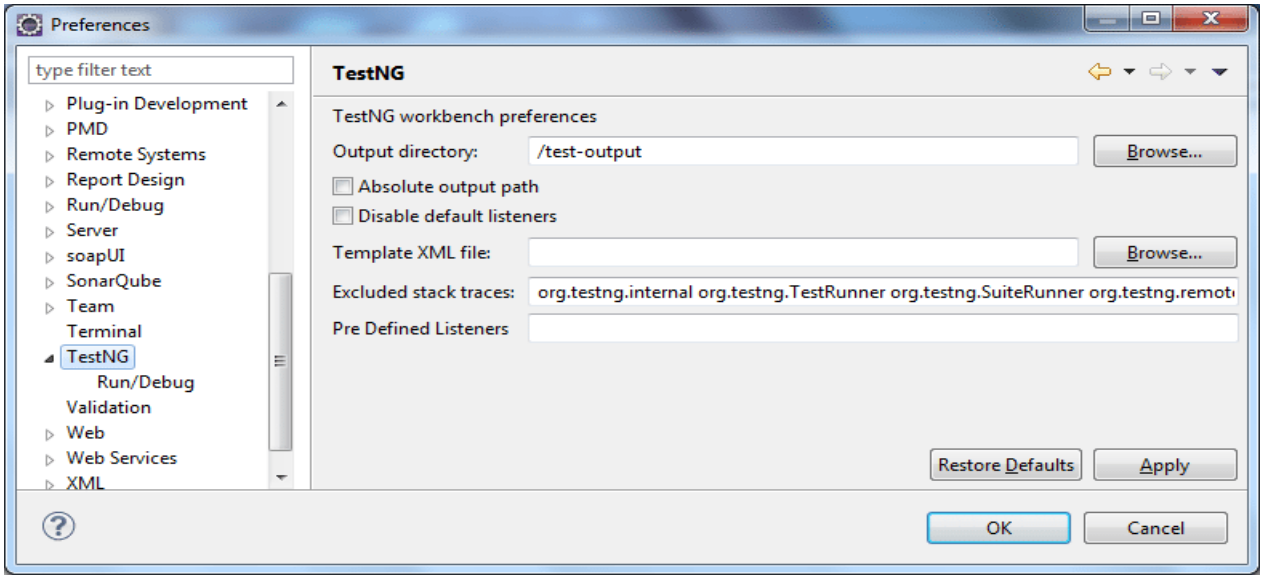


Step 9: In case you get the following **warning window, click on the OK button**.



Step 10: Once the installation is complete, Eclipse will prompt you to **restart it**. Click on Yes on the window prompt.

Step 11: Once Eclipse is restarted, **verify the TestNG plugin installation** by going to Window | Preferences. You will see a TestNG section under the preferences window.

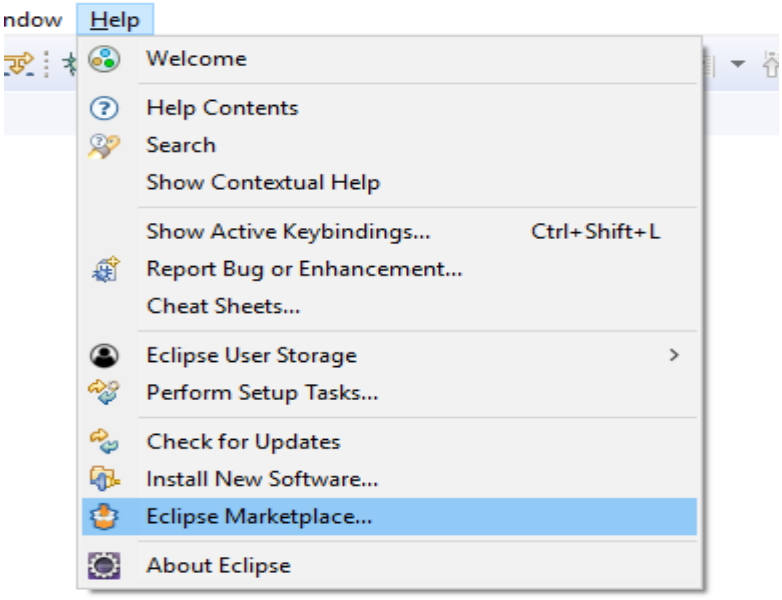


We have successfully installed the TestNG plugin into our Eclipse installation. This will help us in executing our TestNG tests or suite using Eclipse.

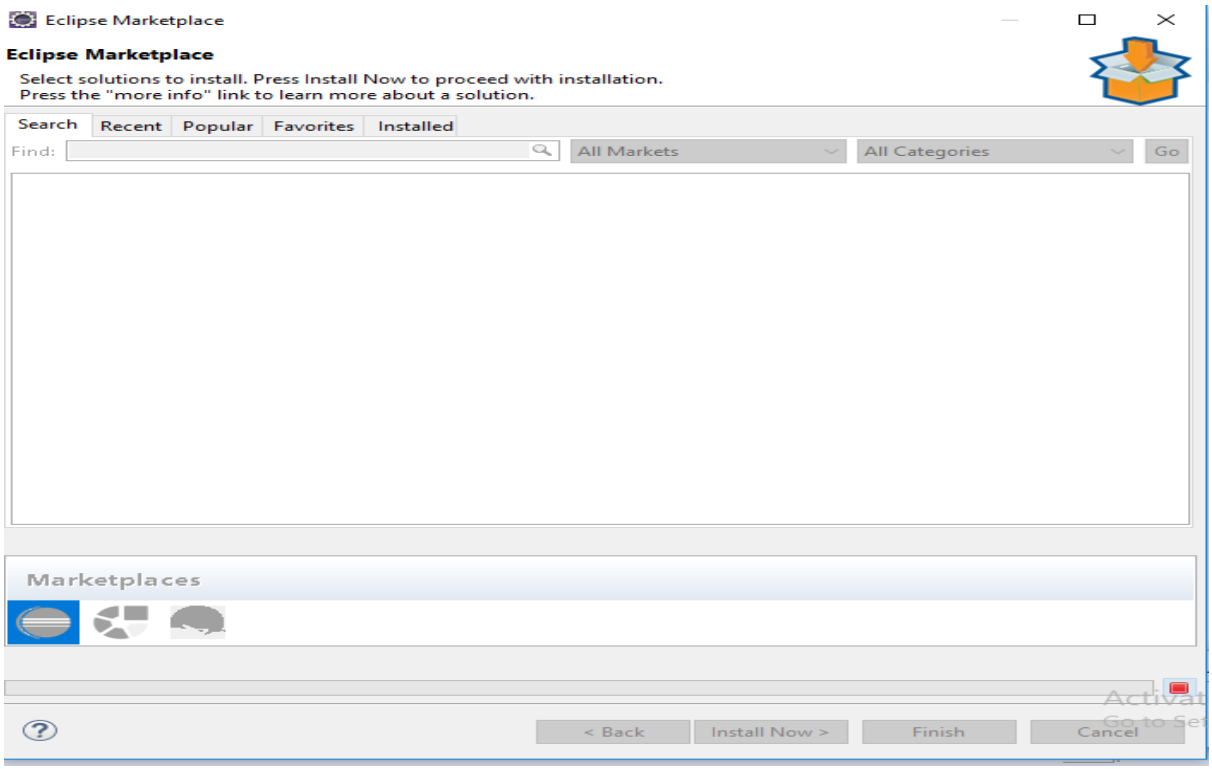
Installing TestNG from eclipse market place.

Step 1: Open the Eclipse Application

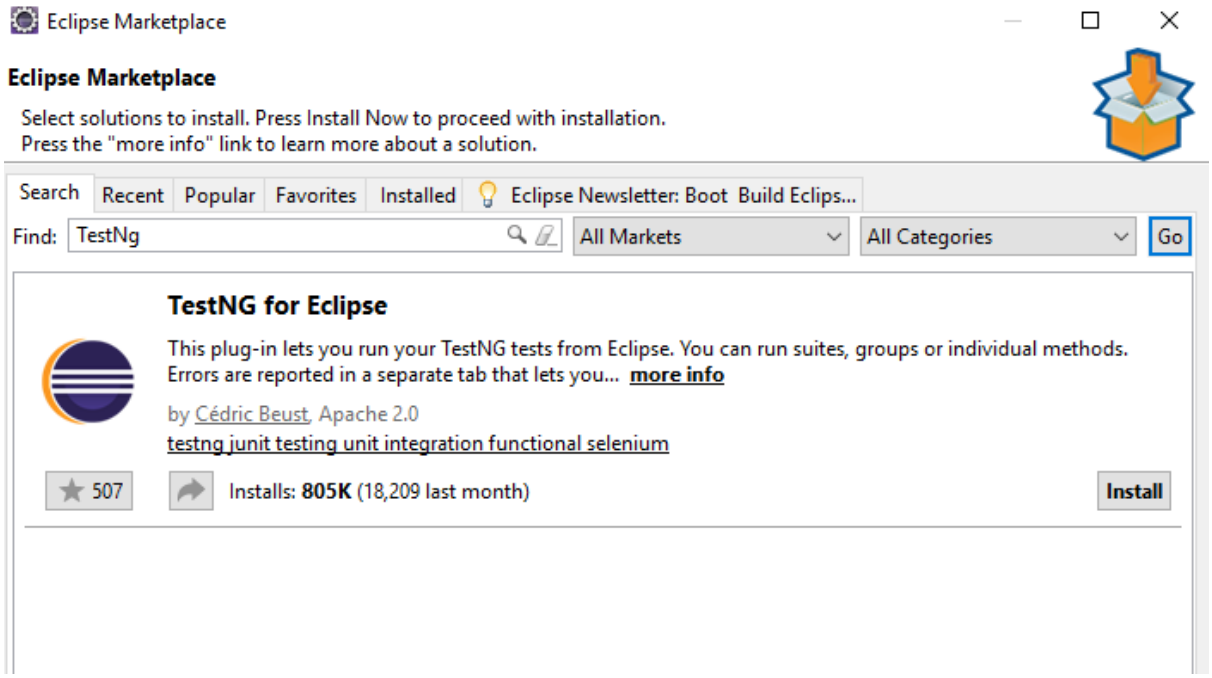
Step 2: Go to **Help| Eclipse Market place**



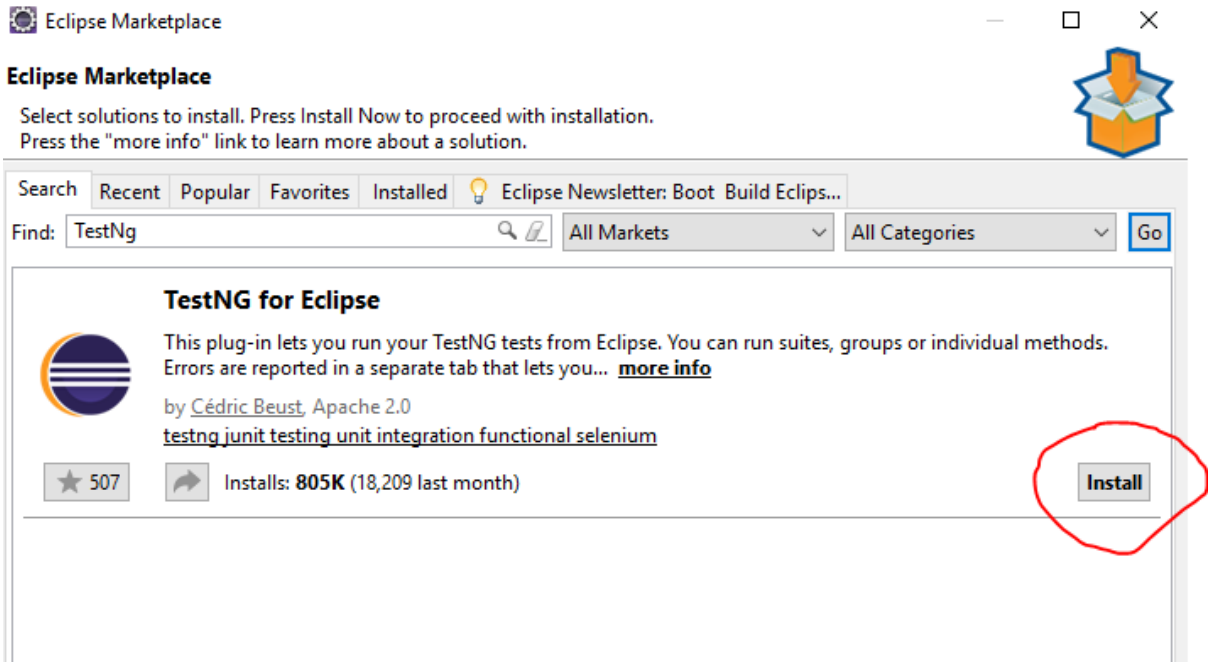
Step 3: Click on Eclipse Market place.



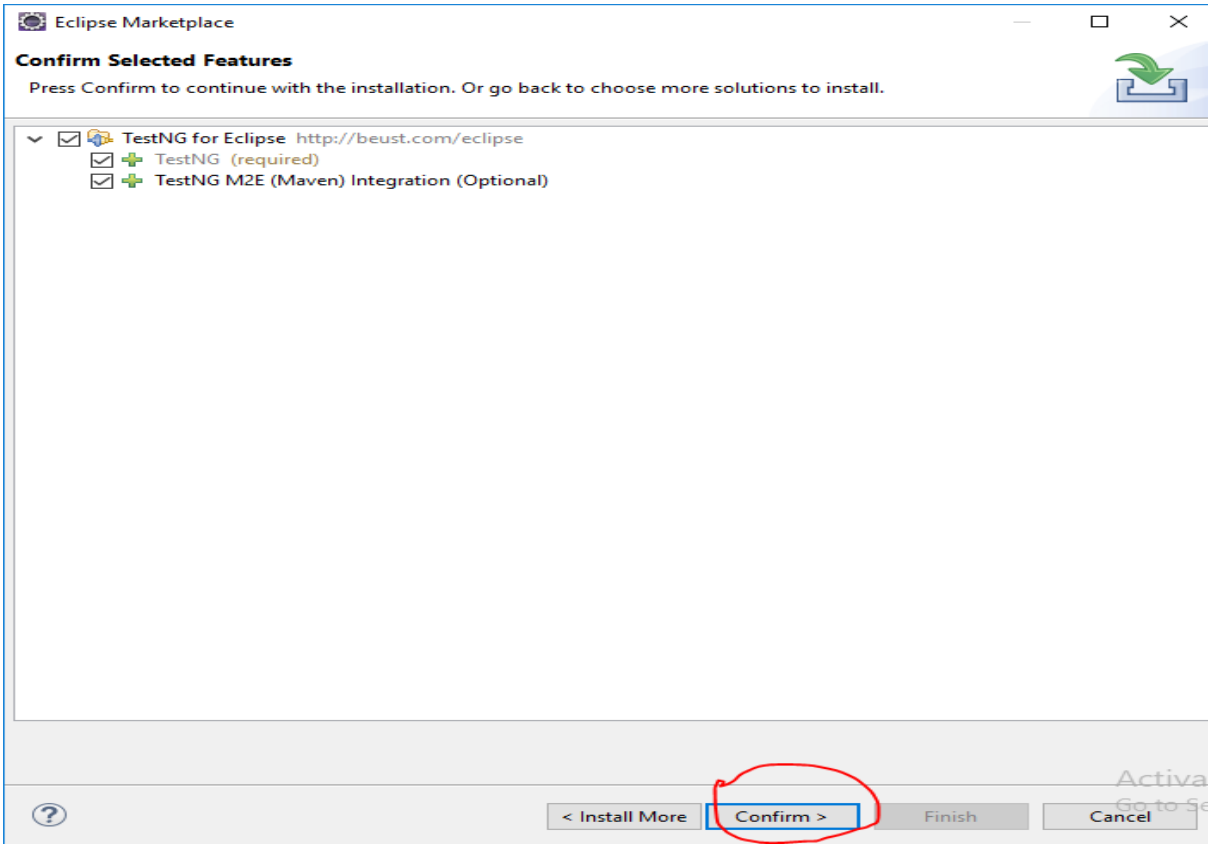
Step 4: Enter “TestNG” in “Find” edit box and click on “Go” button.



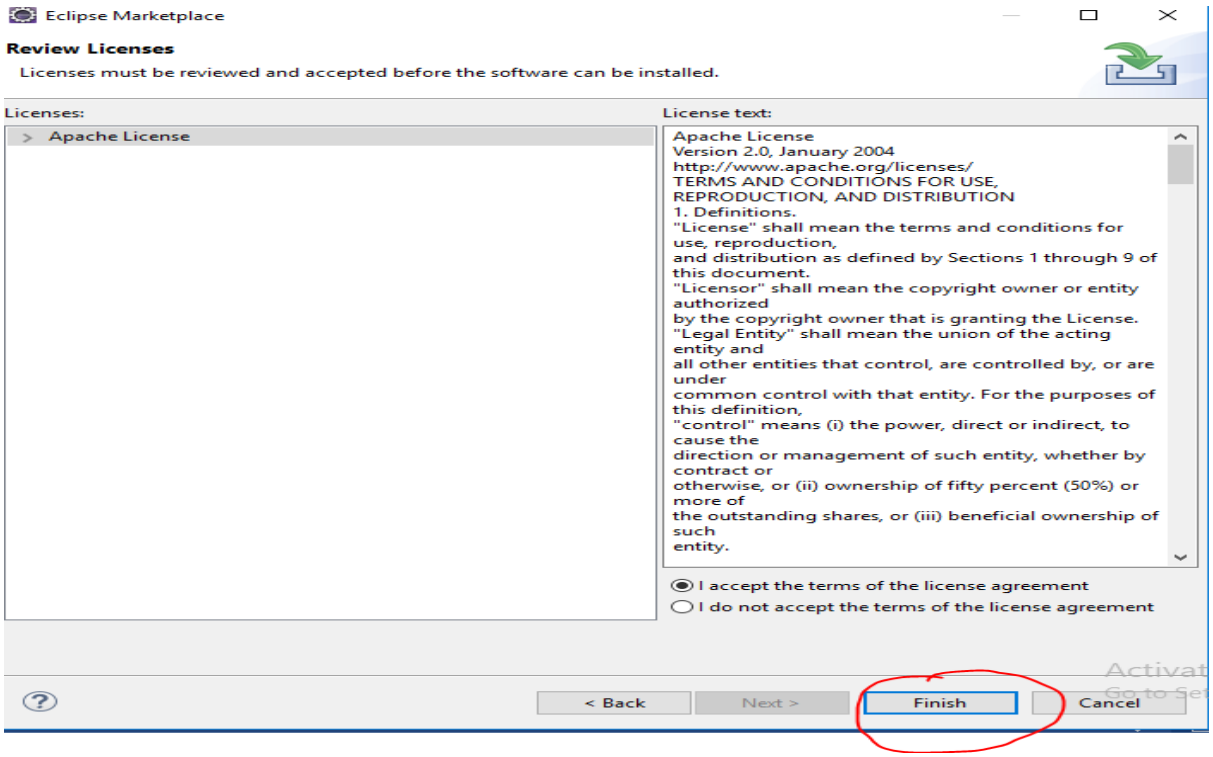
Step 5: Click “Install”



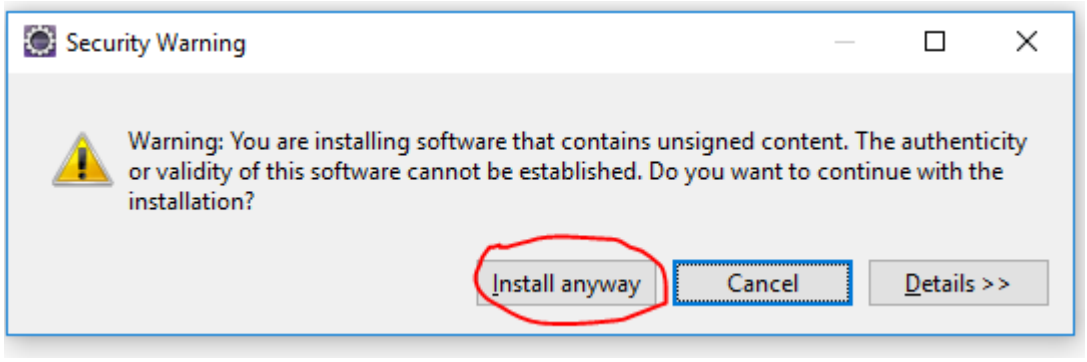
Step 6: Eclipse will calculate the software requirements to download the selected TestNG plugin and will show the Install Details screen. **Click on Confirm on the details screen**. (Make sure all the check boxes have been selected).



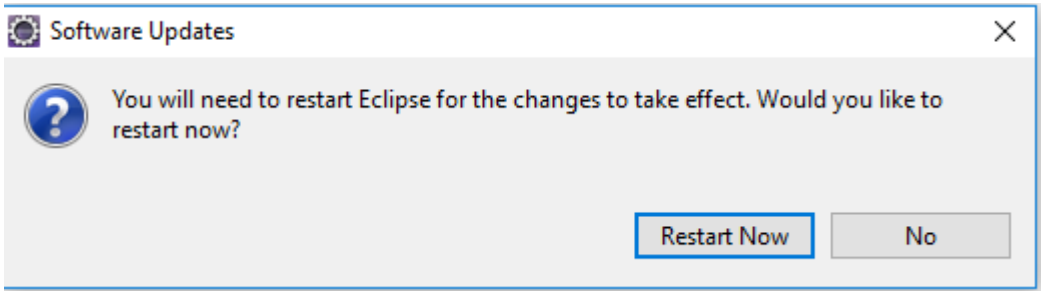
Step 7: **Accept the License Information and click on Finish**. This will start the download and installation of the TestNG plugin onto Eclipse.



Step 8: In case you get the following **warning window, click on the Install Anyway button**.



Step 9: Once the installation is complete, Eclipse will prompt you to **restart it**. Click on Yes on the window prompt.



Note: We have successfully installed the TestNG plugin into our Eclipse installation. This will help us in executing our TestNG tests or suite using Eclipse.

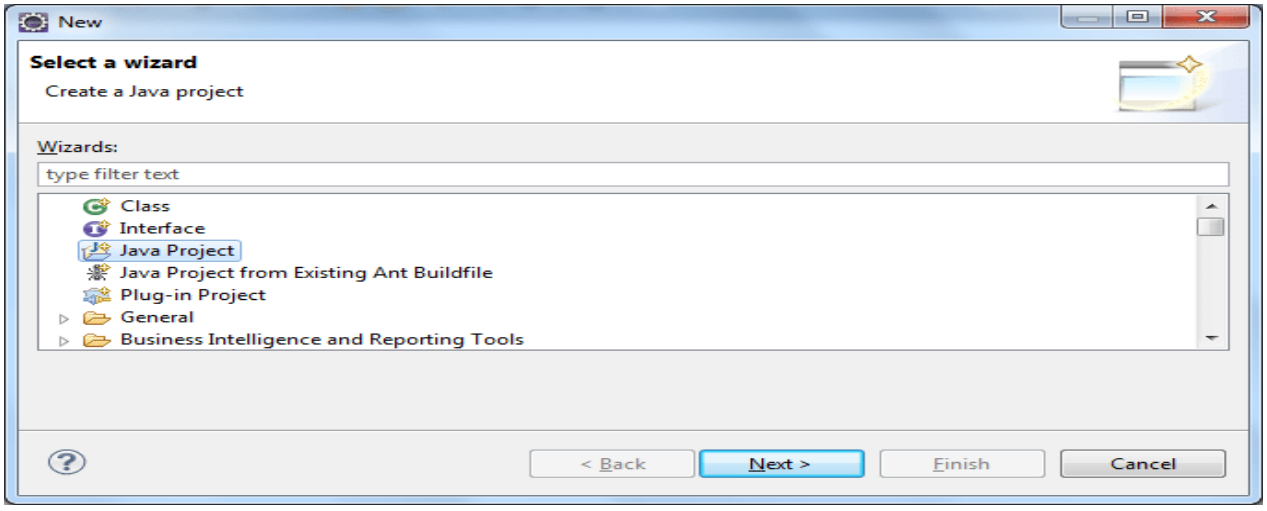
Pramod K S

Creating Java Project with TestNG Dependencies

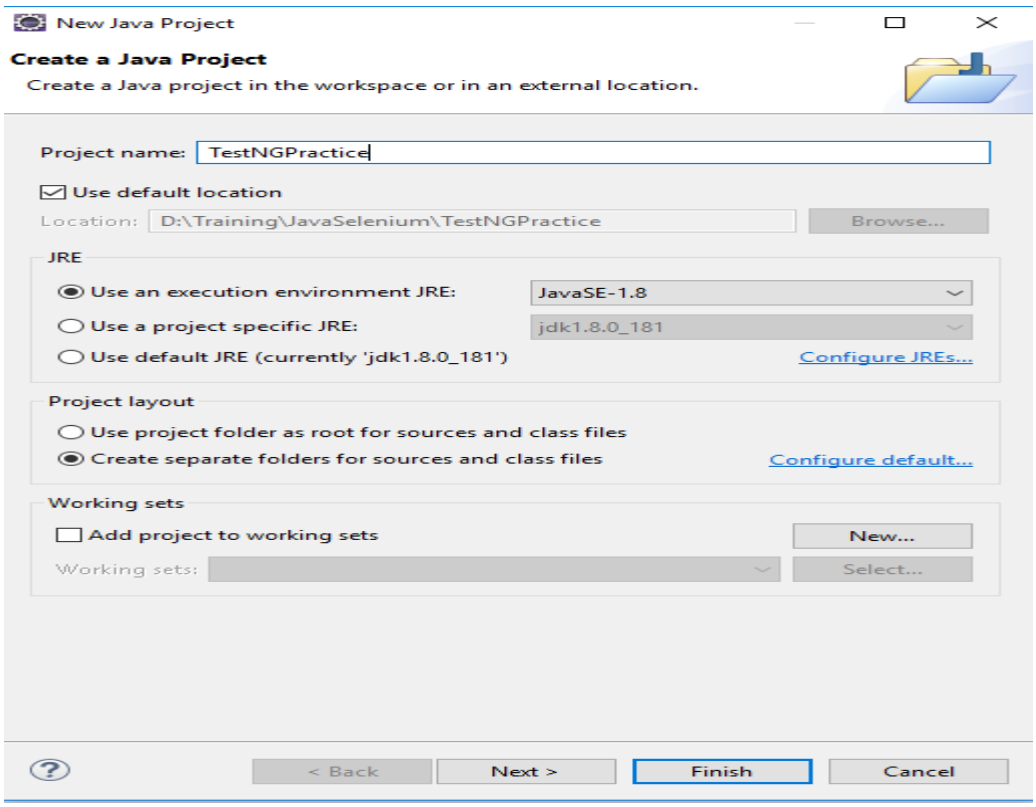
Before we write our first TestNG test, we have to create a Java project in Eclipse and add our TestNG test dependencies.

Step 1: Go to **File | New | Other**. A window with multiple options will be shown.

Step 2: Select Java Project as shown in the following screenshot and click on Next.



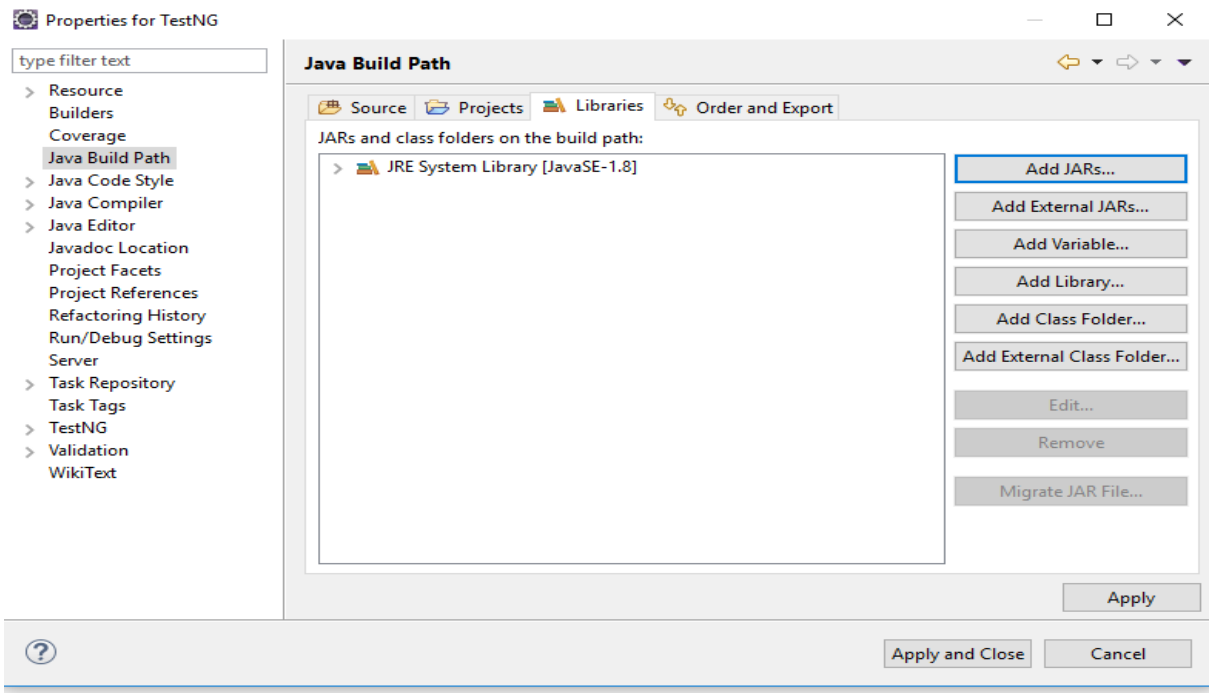
Step 3: On the next screen, **enter a Project name** for a Java project, let's say "**TestNGPractice**", as shown in the following screenshot, and click on Finish:



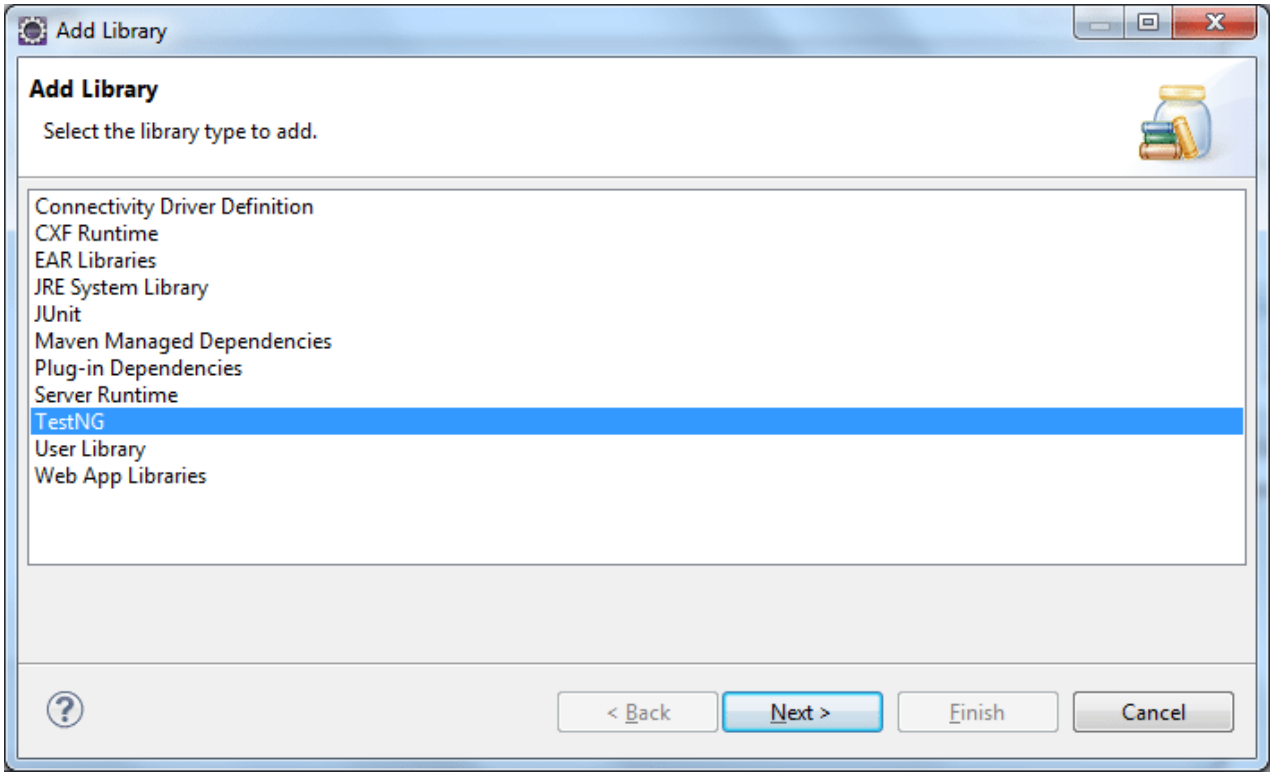
This will create a new Java project in Eclipse.

Pramod K S

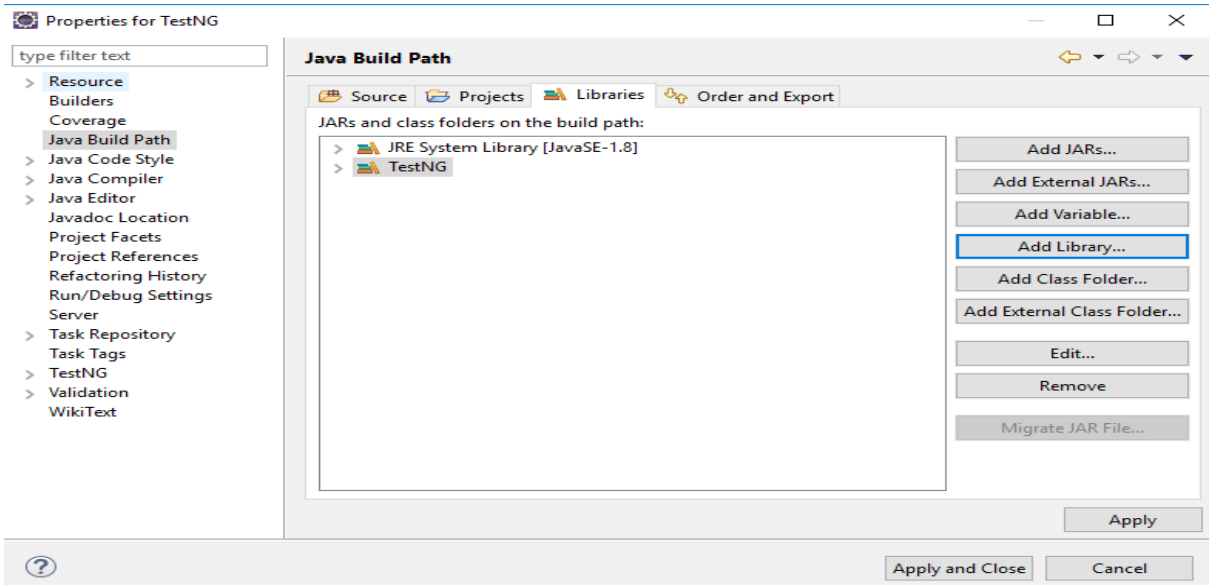
Step 4: Now **go to Project | Properties**. **Select Java Build Path** on the left-hand side on the Properties window as shown in the following screenshot. This will display the build path for the newly created project.



Step 5: Click on the Libraries tab and click on the Add Library… option.
Step 6: Select TestNG on the Add Library window as shown in the following screenshot and click on Next:



Step 7: Click on Finish on your next window. This will add the TestNG library to your Eclipse project.



Great, we have successfully created a new Java project in Eclipse and added a TestNG library to the build path of the project.

Creating your first TestNG class

What is TestNG class?

- It is a java class which has a test method.
- Test method refers to any method which is developed using test annotation.
- While developing TestNG classes, use Test method instead of main method.

Create a Java Class and add the Test Method.

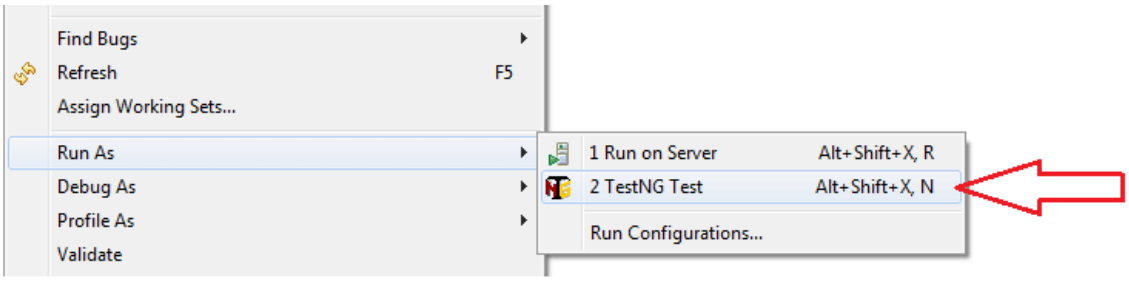
Example TestNG class.

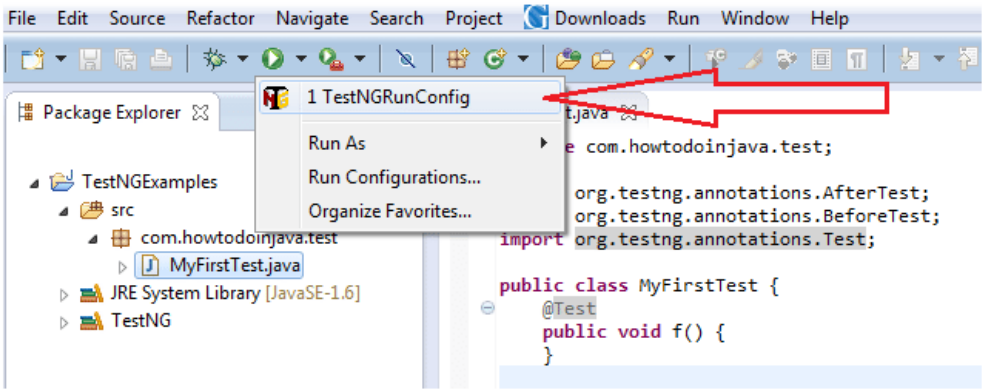
```
import org.testng.annotations.Test;

public class TestNGClass {

    @Test
    public void testMethod() {
        System.out.println("Running test method");
    }
}
```

We have successfully added a new TestNG test class to the newly created Java project in Eclipse. Now let's run the newly created test class through Eclipse.

Running Test Class:

- Run the test class by selecting it and then right-clicking on it, selecting Run as from the menu, and then choosing TestNG Test.

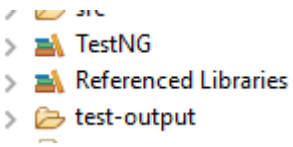




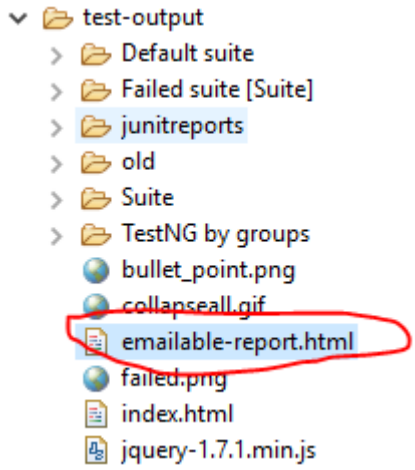
Note: After running, output will be displayed in the console & also it will generate the HTML report.

```
[RemoteTestNG] detected TestNG version 6.14.2
Running test method
PASSED: testMethod

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

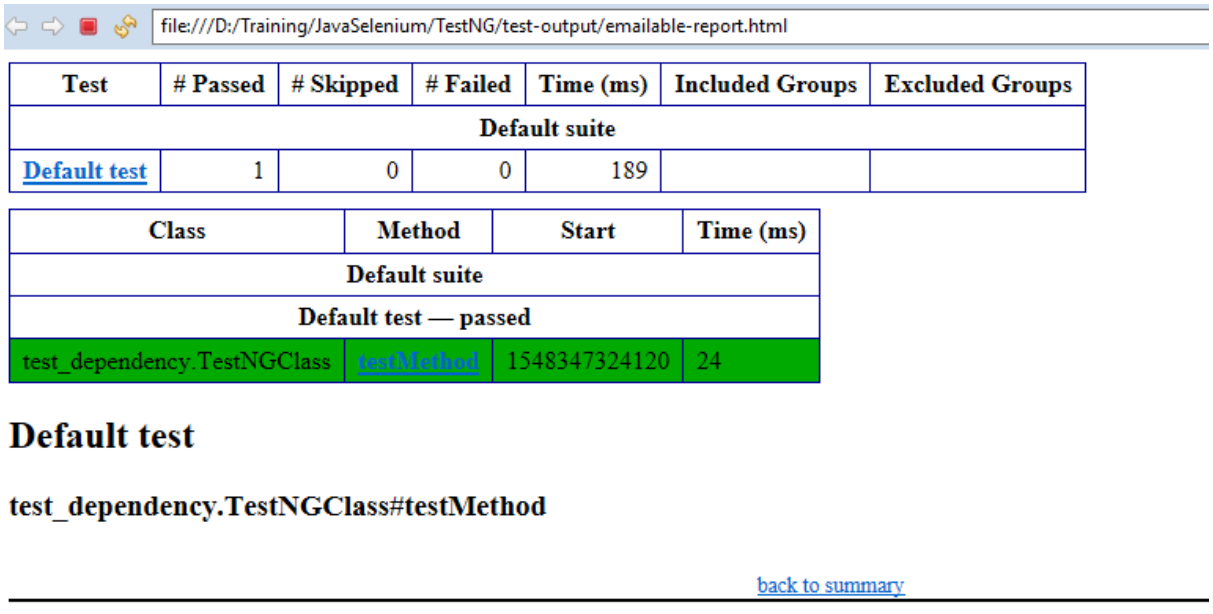
To see the HTML report. Right click on the project and refresh. Test-output folder will be generated.



Expand the “test-output” folder, find the “emailable-report.html”.



Right click on emailable-report.html and Open with| Web Browser.



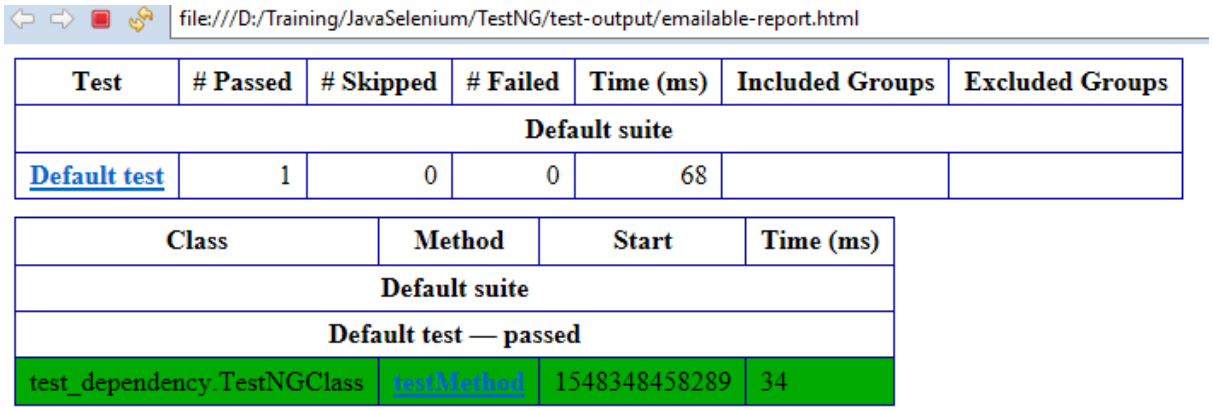
Note: Now, output message is printing on the console but not on the html file.

In-order see the output message on both console and html file. Use the “Reporter.log” method and pass “true” as second argument.

Modify the scripts now, and execute it again.

```java
import org.testng.Reporter;
import org.testng.annotations.Test;

public class TestNGClass {

    @Test
    public void testMethod() {
        Reporter.log("Running test method", true);
    }
}
```

Run -> Refresh the project -> Check both console and html. You will see the output message on both the places.



TestNG Suite

TestNG suite is an XML (Extensible Mark-up Language) which contains list of TestNG classes to be executed.

To create the TestNG XML, right click on the project, go to TestNG, select convert to TestNG & click finish.

It will generate the “testng.xml” file inside the project.

In order to execute it,

Right click on “testng.xml” -> Run as -> select TestNG suite.

Tree Structure of "testng.xml":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite">
    <test name="Test">
        <classes>
            <class name="Qsp.Demo_one" />
            <class name="Qsp.Demo_two" />
            <class name="Qsp.Demo_three" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Note: Instead of specifying the individual class names we can specify only package name, so that it can execute all the test classes present in the specified package.

Example:

"Package Tree Structure"

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite">
    <test name="Test">
        <packages>
            <package name="Qsp"></package>
        </packages>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Note: We can also add multiple packages.

Question: Can we develop multiple test methods in one test class.

Answer: Yes

Question: If test class contains multiple test methods, what is the order of execution?

Answer: It will execute it in alphabetical order.

You can also specify methods to be included and excluded:

Example "testng.xml"

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite">
    <test name="Test">
        <classes>
            <class name="Qsp.Demo_one">
                <methods>
                    <include name="testMethod_One"></include>
                    <exclude name="testMethod_two"></exclude>
                </methods>
            </class>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Note: We need to use the “methods” tag inside the “class” tag,

Use “include” tag to include the test for execution.

Use “exclude” tag to exclude the test for execution.

Methods, which are inside the include tag will be executed.

Methods, which are inside the exclude tag will not be executed.

TestNG Annotations Tutorial

- TestNG uses annotations to help developers to write the tests. The following is a table containing information about all the annotations provided by TestNG and a brief description of them. TestNG Test Configuration Annotations.

ANNOTATION	DESCRIPTION
@BeforeSuite	The annotated method will be executed before any tests declared inside a TestNG suite.

@AfterSuite	The annotated method will be executed after any tests declared inside a TestNG suite.
@BeforeTest	The annotated methods will be executed before each test section declared inside a TestNG suite.
@AfterTest	The annotated methods will be executed after each test section declared inside a TestNG suite.
@BeforeGroups	BeforeGroups annotated method will run before any of the test method of the specified group is executed.
@AfterGroups	AfterGroups annotated method will run after any of the test method of the specified group gets executed.
@BeforeClass	BeforeClass annotated method is executed before any of the test method of a test class.

@AfterClass	AfterClass annotated method is executed after the execution of every test methods of a test class are executed.
@BeforeMethod	These annotated methods are executed before the execution of each test method.
@AfterMethod	These annotated methods are executed after the execution of each test method.
@DataProvider	Marks a method as a data providing method for a test method. The said method has to return an Object double array (Object [ ] [ ]) as data.
@Factory	Marks an annotated method as a factory that returns an array of class objects (Object [ ]). These class objects will then be used as test classes by TestNG. This is used to run a set of test cases with different values.

@Listeners	Applied on a test class. Defines an array of test listener's classes extending "org.testng.ITestNGListener". Helps in tracking the execution status and logging purpose.
@Parameters	This annotation is used to pass parameters to a test method. These parameter values are provided using the testng.xml configuration file at runtime.
@Test	Marks a class or a method as a test method. If used at class level, all the public methods of a class will be considered as a test method.