



Selenium Training – Day3

05 June 2014



Points to be Covered:

1. What is TestNG
2. Setting up TestNG with Eclipse
3. TestNG (Next Generation Testing Framework) – Understanding Annotations
4. TestNG – Test Automation with Selenium

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:

- Annotations.
- Flexible test configuration.
- Support for data-driven testing (with @DataProvider).
- Support for parameters.
- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc...

Requirement:

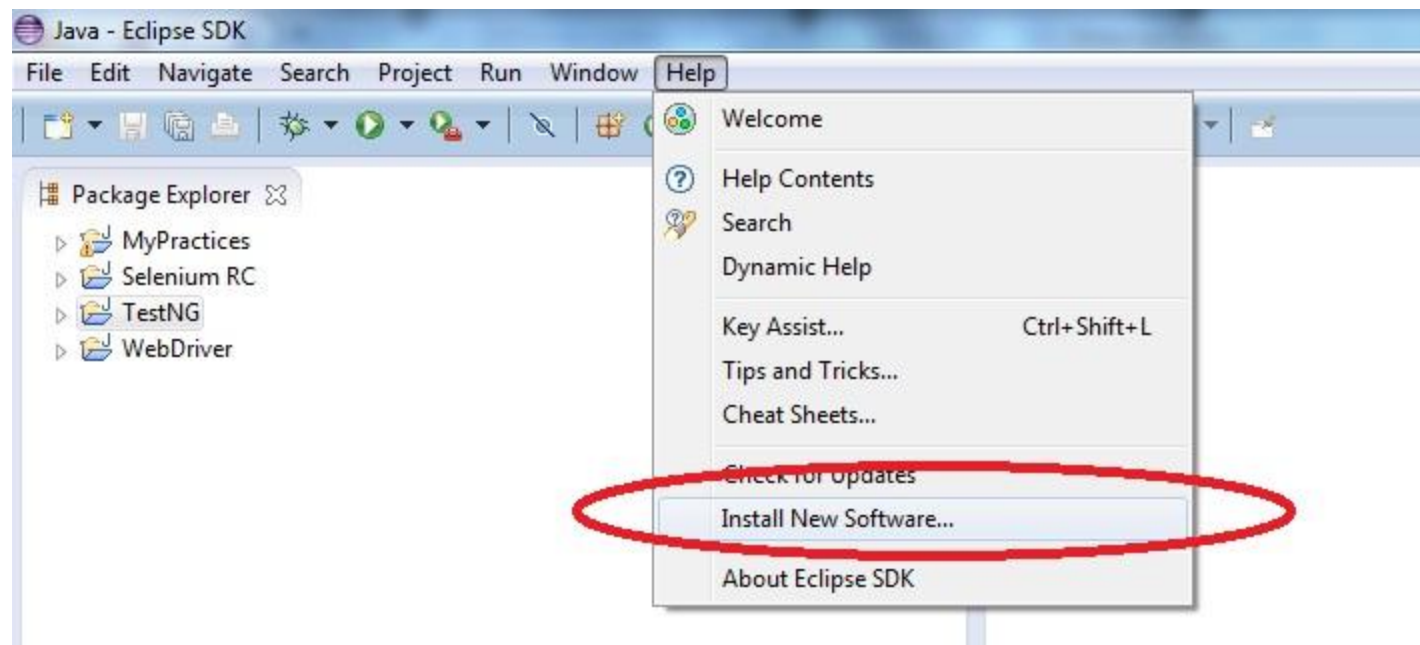
TestNG requires JDK 5 or higher.

Setting up TestNG with Eclipse

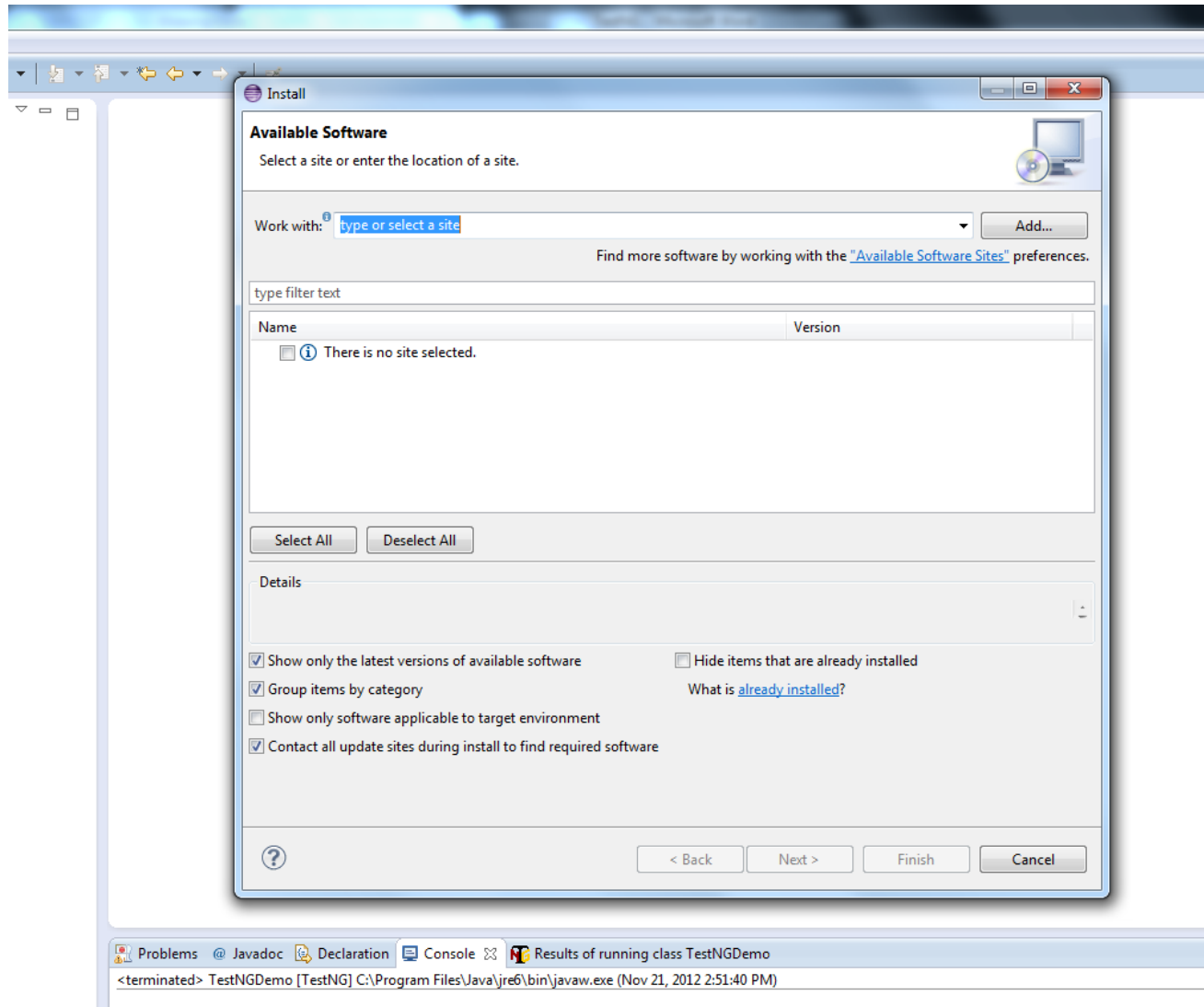
There is no need to download any Jar file or exe file for installation. We need to just utilize the “Install New Software” option available in the Eclipse.

Steps for installation:

1. Click Help → Install New Software

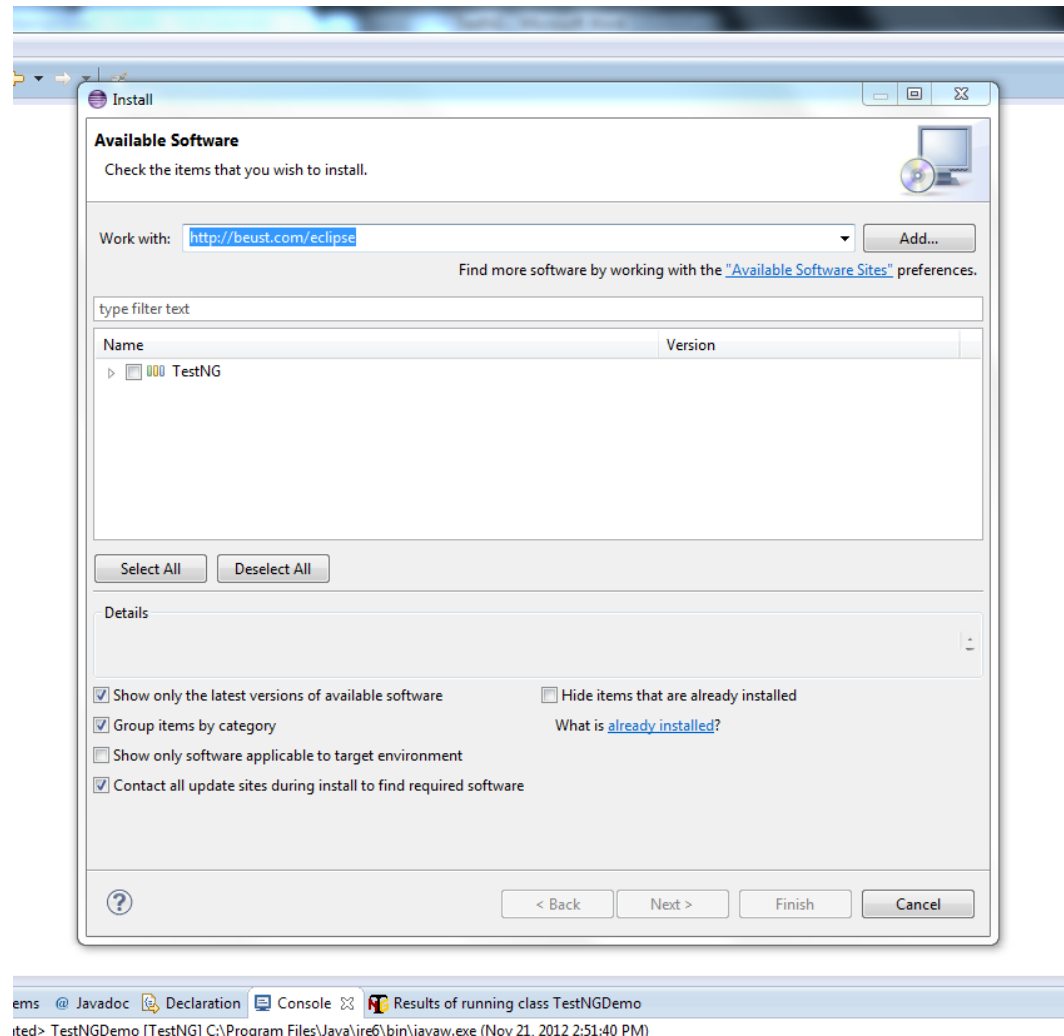


Setting up TestNG with Eclipse



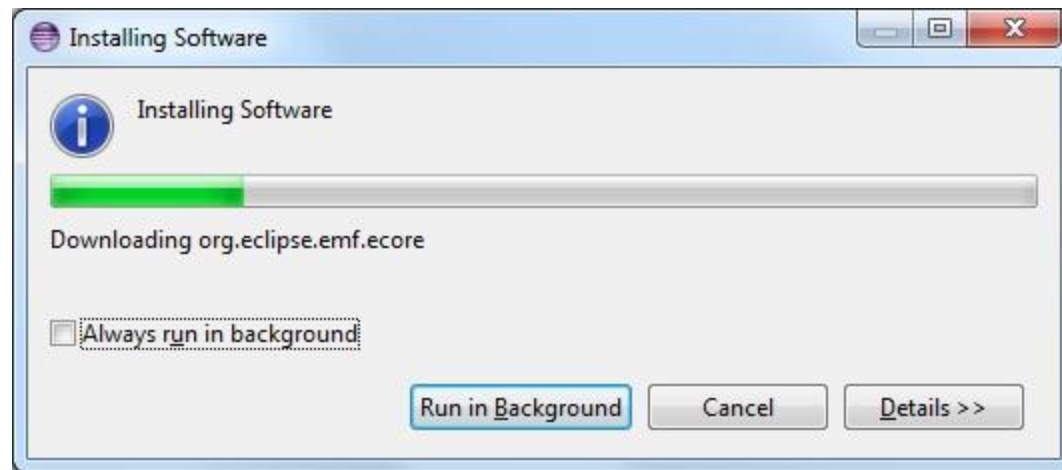
Setting up TestNG with Eclipse

2. Type "<http://beust.com/eclipse>" in the "Work with" edit box and click 'Add' button
3. In the 'Name' column we can see "TestNG" -> Select this and click 'Next' button



Setting up TestNG with Eclipse

4. Click Next and click on the radio button "I accept the terms of the license agreement"
5. Click 'Next' button
6. Click 'Finish'

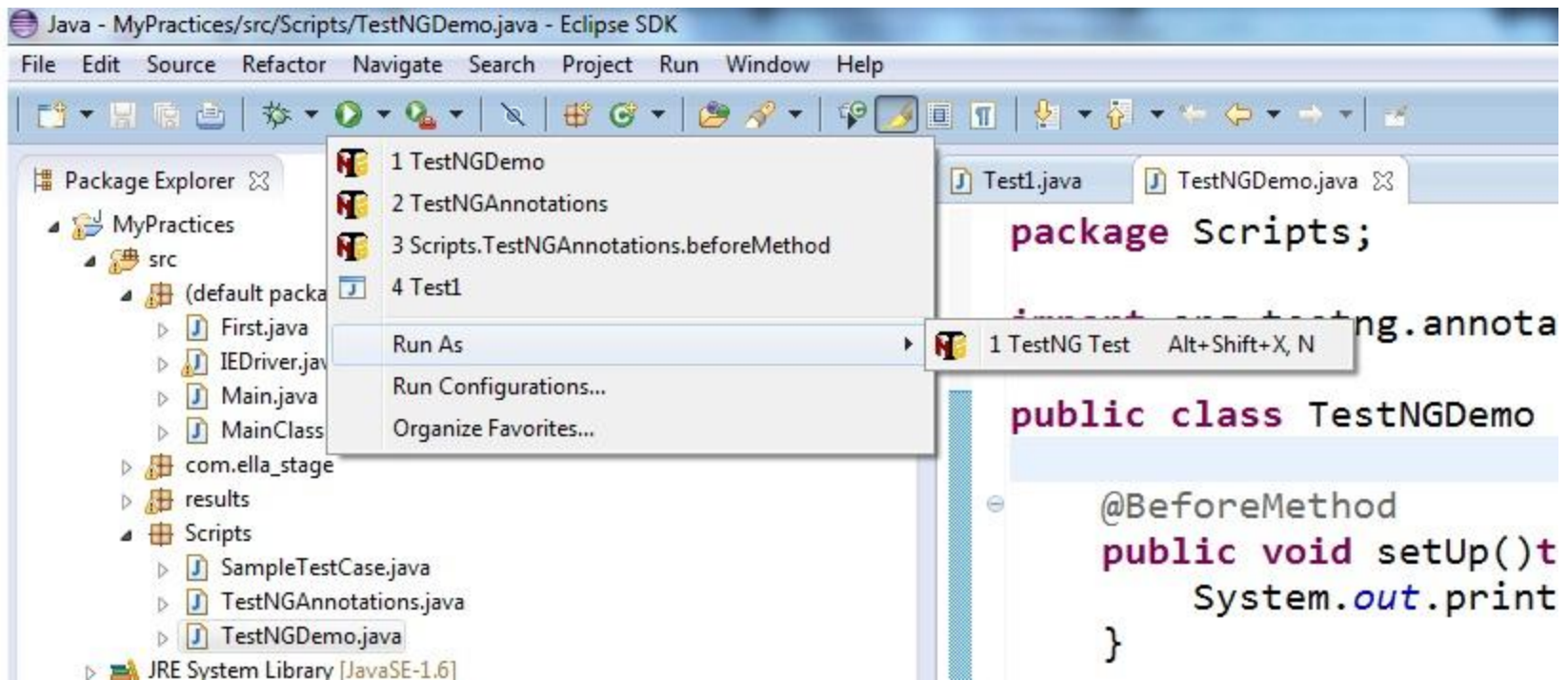


Setting up TestNG with Eclipse

This will install the TestNG plug-in for Eclipse

After the installation, it will ask for restart of Eclipse. Then restart the Eclipse.

Once the Eclipse is restarted, we can see the TestNG icons & menu items as in the below figures.



Setting up TestNG with Eclipse

```
Scripts;
```

```
org.testng.annotations.*;
```

```
class Test
```

```
beforeMethod
```

```
public void
```

```
System.out
```

```
test
```

```
public void
```

```
System.out
```

```
test
```

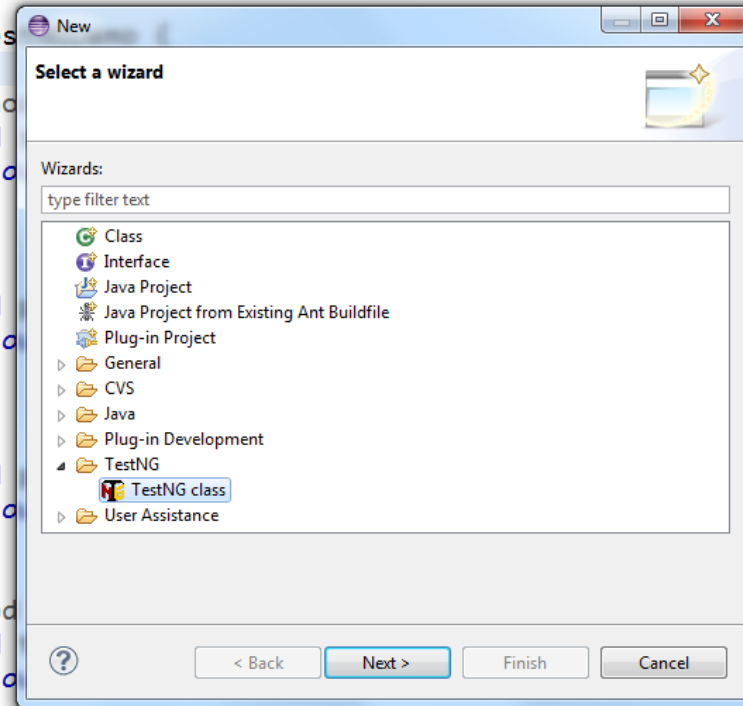
```
public void
```

```
System.out
```

```
afterMethod
```

```
public void
```

```
System.out
```



Annotation:

Annotation defines a type and it can be applied to several Java elements such as Java methods, classes, etc. Annotation adds meta-data facility to Java elements. Some examples of Java built-in annotations are:

@Override

@Deprecated

@SuppressWarnings

Each annotation will instruct the compiler to do something. For example, @Override tells the compiler to check whether the parent class contains the same method (also checks whether the overriding method follows all the rules correctly).

TestNG – Understanding Annotations

Let us have a look at TestNG annotations.

1. @BeforeSuite
2. @AfterSuite
3. @BeforeTest
4. @AfterTest
5. @BeforeGroups
6. @AfterGroups
7. @BeforeClass
8. @AfterClass
9. @BeforeMethod
10. @AfterMethod
11. @DataProvider
12. @Factory
13. @Listeners
14. @Parameters
15. @Test

Out of these annotations, we will try to understand only “@Test, @BeforeSuite, @AfterSuite, @BeforeClass, @AfterClass, @BeforeMethod, @AfterMethod, @DataProvider, @BeforeTest and @AfterTest”.

Every Java application should contain “main” method and the execution of the entire project starts from “main” method. With using TestNG as a testing framework, there is no need to use “main” method. The execution will be handled by the TestNG (testing framework).

@Test

If a Java method marked with @Test, then the compiler understands that the method is a “Test Case”.

@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.

@BeforeClass

A Java method marked with @BeforeClass annotation will be executed before executing all the methods in the class. This annotation can be used to make sure that the configuration or some precondition for entire set of “test methods” inside the class will be executed before all the methods.

@AfterClass

A Java method marked with @AfterClass annotation will be executed after executing all the methods in the class. Hence, this annotation helps in executing a particular set of actions that needs to be done after executing all the “Test Cases” in the class.

@BeforeMethod

Execute the method just before executing any Test Case in the class.

@AfterMethod

Execute the method after executing every Test Case in the class.

@DataProvider

Marks a method as supplying data for a test method. The annotated method must return an `Object[][]` where each `Object[]` can be assigned the parameter list of the test method. The `@Test` method that wants to receive data from this `DataProvider` needs to use a `dataProvider` name equals to the name of this annotation.

@BeforeTest

The annotated method will be run before any test method belonging to the classes inside the `<test>` tag is run.

@AfterTest

The annotated method will be run after all the test methods belonging to the classes inside the `<test>` tag have run.

TestNG – Understanding Annotations

Example:

Now, open Eclipse IDE and create a .Java file and use the following code.

```
package test;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class TestNGAnnotations {

    @BeforeClass
    public void beforeClass(){
        System.out.println("In BeforeClass");
    }

    @BeforeMethod
    public void beforeMethod(){
        System.out.println("In BeforeMethod");
    }
}
```


TestNG – Understanding Annotations

@Test

```
public void testOne(){  
    System.out.println("In TestOne");  
}
```

@Test

```
public void testTwo(){  
    System.out.println("In TestTwo");  
}
```

@AfterMethod

```
public void afterMethod(){  
    System.out.println("In AfterMethod");  
}
```

@AfterClass

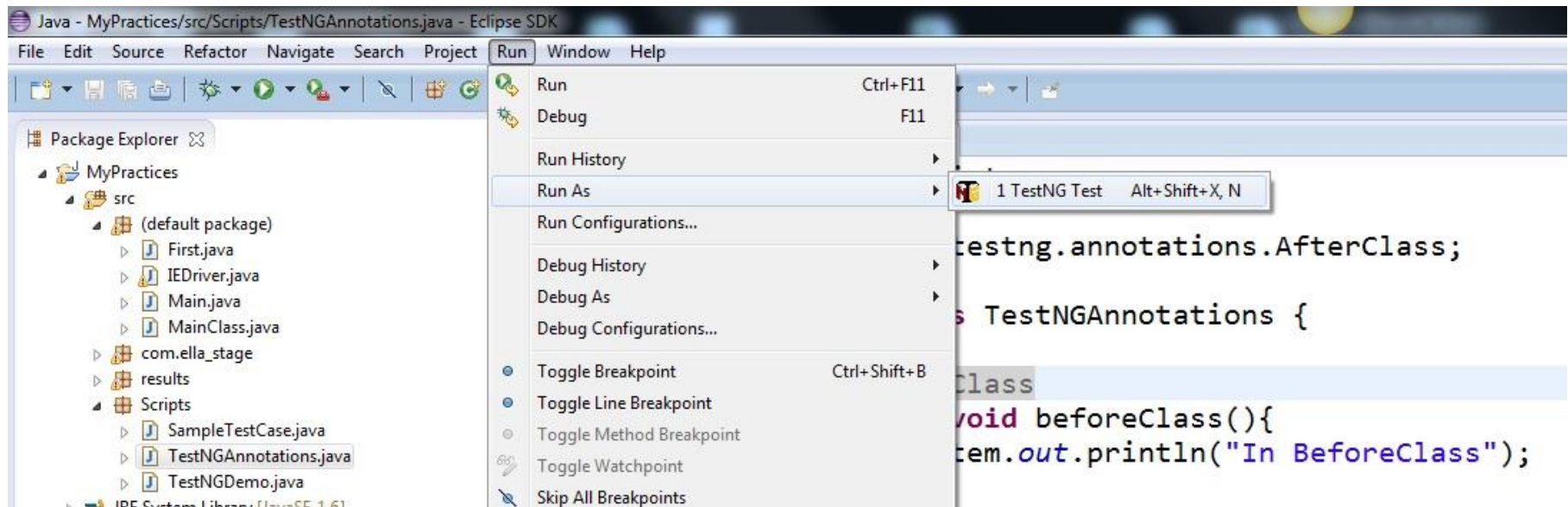
```
public void afterClass(){  
    System.out.println("In AfterClass");  
}
```

```
}
```

TestNG – Understanding Annotations

Execute the code as below:

Click Run → Run As → TestNG Test



TestNG – Understanding Annotations

Output of the executing should be as follows:

```
Problems @ Javadoc Declaration Console Results of running class TestNGAnnotations
<terminated> TestNGAnnotations [TestNG] C:\Program Files\Java\jre6\bin\javaw.exe (Nov 21, 2012 4:43:03 PM)
[TestNG] Running:
  C:\Users\veera.kasina\AppData\Local\Temp\testng-eclipse-838263884\testng-customsuite.xml

In BeforeClass
In BeforeMethod
In TestOne
In AfterMethod
In BeforeMethod
In TestTwo
In AfterMethod
In AfterClass
PASSED: testOne
PASSED: testTwo

=====
      Default test
      Tests run: 2, Failures: 0, Skips: 0
=====

Default suite
Total tests run: 2, Failures: 0, Skips: 0
=====

[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@750159: 30 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@f84386: 22 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@1546e25: 58 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@127734f: 216 ms
[TestNG] Time taken by org.testng.reporters.EmailableReporter@efd552: 9 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms
```

TestNG – Understanding Annotations

This execution shows how exactly the annotations instruct the java to execute the test methods.

The below figure shows the order of execution



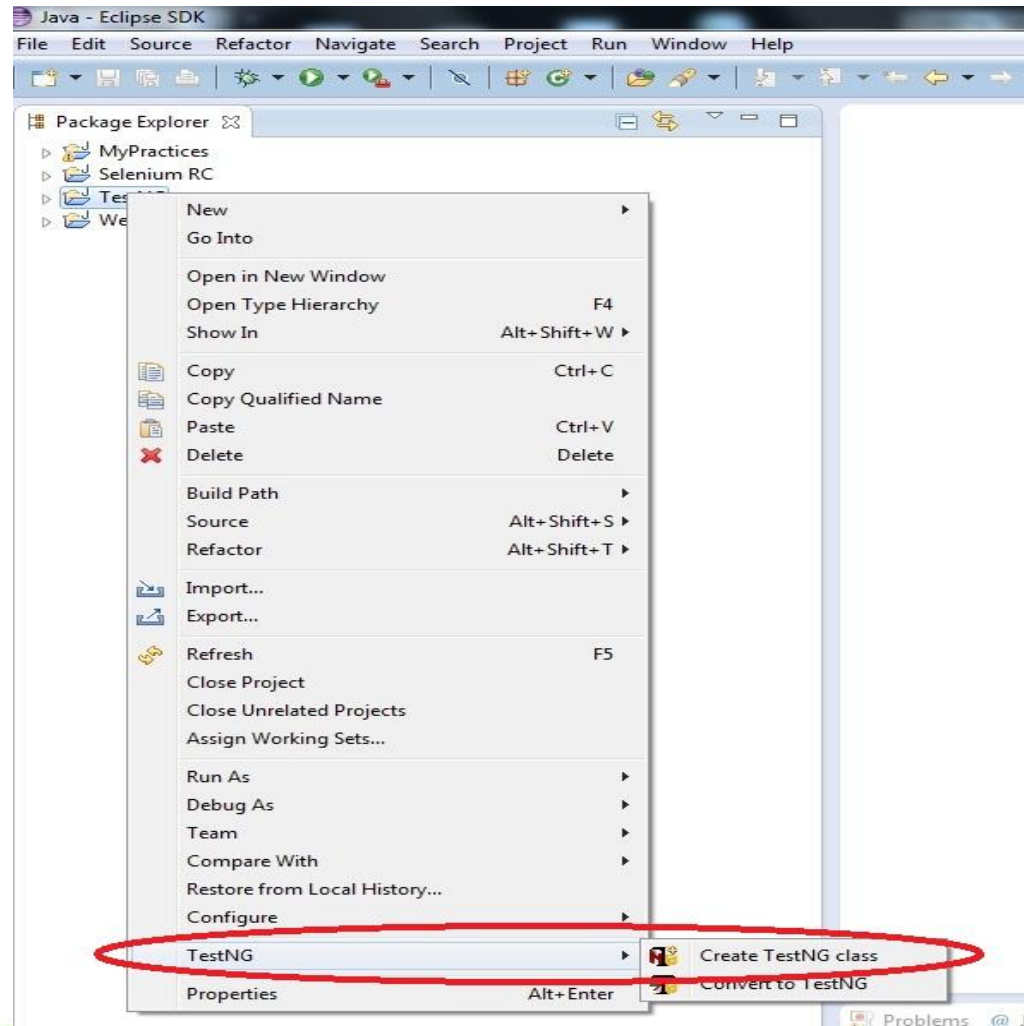
TestNG framework can be used for automation testing with Selenium (web application automation testing tool). Then, definitely a question will popup in our mind “why this framework is needed?” Is it possible to execute Selenium tests without frameworks like JUnit or TestNG. Answer is “Yes”, it is possible to execute Selenium tests without using these frameworks. But, is it possible to get proper reports without framework? How the verification points or checkpoints are going to be handled? How the exception handling is done? If we do not have any frameworks, then it is difficult to get proper reports, handle checkpoints, or exception handling.

Either we should use any of the frameworks available like JUnit, TestNG or we should design our own framework.

TestNG – Test Automation with Selenium

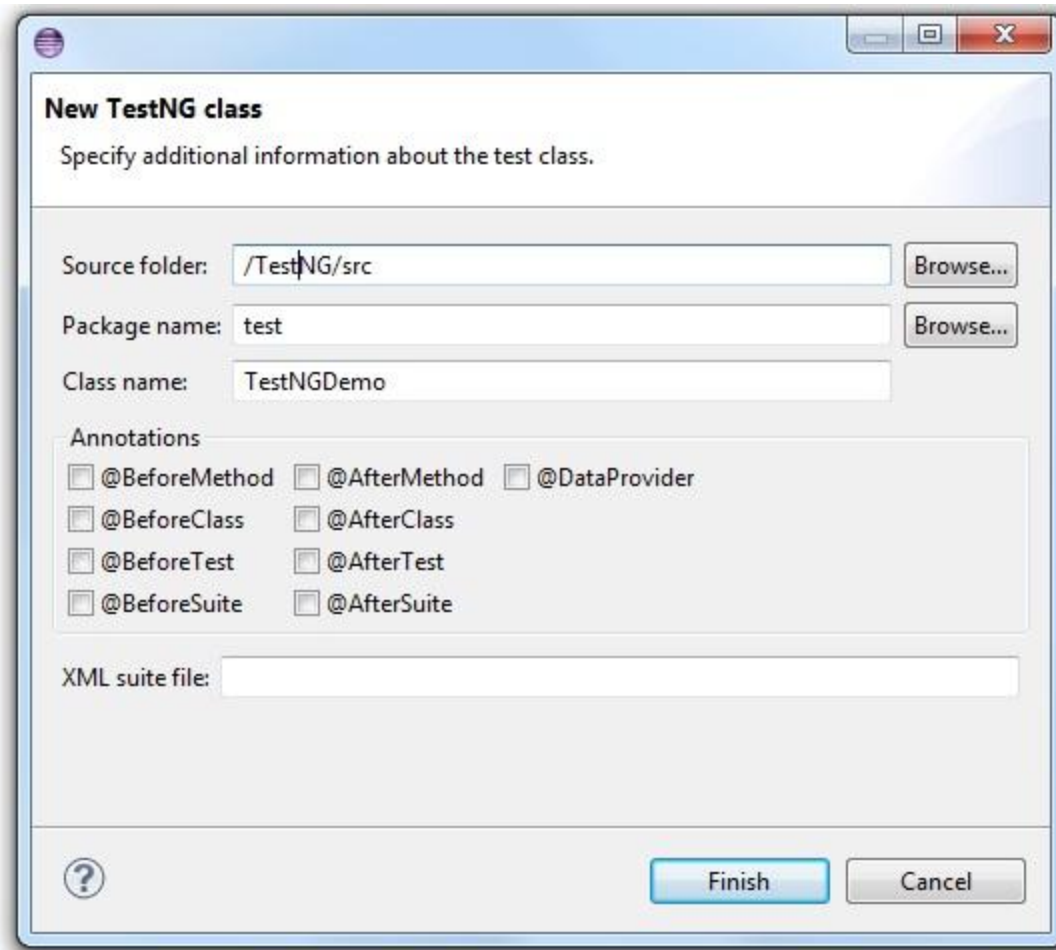
How to Create a TestNG Class:

1. Right click on Project folder -> TestNG -> Create TestNG Class



TestNG – Test Automation with Selenium

2. Provide the Source folder, Package name and Class name
3. Click on Finish



New TestNG class

Specify additional information about the test class.

Source folder:

Package name:

Class name:

Annotations

<input type="checkbox"/> @BeforeMethod	<input type="checkbox"/> @AfterMethod	<input type="checkbox"/> @DataProvider
<input type="checkbox"/> @BeforeClass	<input type="checkbox"/> @AfterClass	
<input type="checkbox"/> @BeforeTest	<input type="checkbox"/> @AfterTest	
<input type="checkbox"/> @BeforeSuite	<input type="checkbox"/> @AfterSuite	

XML suite file:

TestNG – Test Automation with Selenium

Let us have a look at the code snippet below:

```
package test;
import org.testng.annotations.*;
public class TestNGDemo {

    @BeforeMethod
    public void setUp()throws Exception{
        System.out.println("Selenium Will Start");
    }

    @Test
    public void performOne() {
        System.out.println("Perform Operations-1");
    }

    @Test
    public void performTwo() {
        System.out.println("Perform Operations-2");
    }
}
```

```
@AfterMethod
public void tearDown(){
    System.out.println("Stop Selenium");

}

}
```

The Java class “TestNGDemo” is implemented in Eclipse IDE using TestNG framework.

In the above code, there are two test methods, which are marked with @Test annotation. There are two other methods, “setUp” and “tearDown” which are marked with @BeforeMethod and @AfterMethod annotations. Hence Before executing each test, setUp method will be executed. After the execution of each test, tearDown gets executed. Hence, Selenium gets instantiated and browser gets opened and closed twice during the execution.

TestNG – Test Automation with Selenium

Output of the execution should be as below:

```
Problems @ Javadoc Declaration Console Results of running class TestNGDemo
<terminated> TestNGDemo [TestNG] C:\Program Files\Java\jre6\bin\javaw.exe (Nov 21, 2012 4:56:51 PM)
[TestNG] Running:
  C:\Users\veera.kasina\AppData\Local\Temp\testng-eclipse-2148161\testng-customsuite.xml

Selenium Will Start
Perform Operations-1
Stop Selenium
Selenium Will Start
Perform Operations-2
Stop Selenium
PASSED: performOne
PASSED: performTwo

=====
    Default test
    Tests run: 2, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 2, Failures: 0, Skips: 0
=====

[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@750159: 13 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@f84386: 20 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@1546e25: 51 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@127734f: 212 ms
[TestNG] Time taken by org.testng.reporters.EmailableReporter@efd552: 8 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms
|
```

TestNG – Test Automation with Selenium

Test execution also creates set of XML & HTML reports. TestNG creates “test-output” folder in the root folder, inside which we can see the reports. Below are the screenshots of some of the reports created by TestNG.

TestNG: Unit Test							
file:///D:/Automation/TestNG/test-output/emailable-report.html							
Test	Methods Passed	Scenarios Passed	# skipped	# failed	Total Time	Included Groups	Excluded Groups
Default test	2	2	0	0	0.0 seconds		

Class	Method	# of Scenarios	Start	Time (ms)
Default test — passed				
test.TestNGDemo	performOne	1	1353497212404	4
	performTwo	1	1353497212409	1

Default test

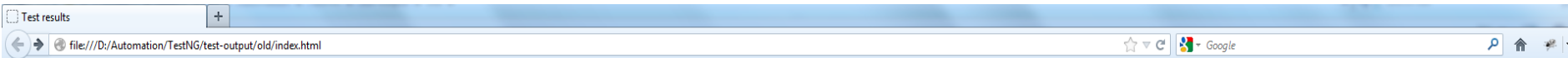
test.TestNGDemo:performTwo

[back to summary](#)

test.TestNGDemo:performOne

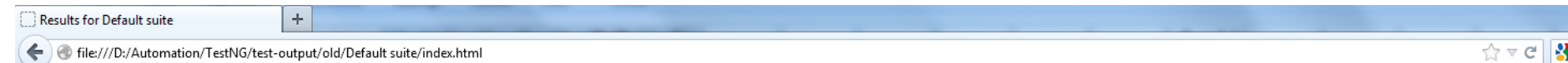
[back to summary](#)

TestNG – Test Automation with Selenium



Test results

Suite	Passed	Failed	Skipped	testng.xml
Total	2	0	0	
Default suite	2	0	0	Link



Results for Default suite

1 test	1 class	2 methods: chronological alphabetical not run (0)
0 group	reporter output	testng.xml

Default test (2/0/0)

[Results](#)

Methods run, sorted chronologically

>> means before, << means after

Default suite

(Hover the method name to see the test class name)

Time	Delta (ms)	Suite configuration	Test configuration	Class configuration	Groups configuration	Method configuration	Test method	Thread	Instances
12/11/21 16:56:52	0						performOne	main@13673945	
12/11/21 16:56:52	5						performTwo	main@13673945	
12/11/21 16:56:52	-4					>>setUp		main@13673945	
12/11/21 16:56:52	5					>>setUp		main@13673945	
12/11/21 16:56:52	5					<<tearDown		main@13673945	
12/11/21 16:56:52	7					<<tearDown		main@13673945	

TestNG – Test Automation with Selenium

TestNG reports +

file:///D:/Automation/TestNG/test-output/index.html

Test results
1 suite

All suites

Default suite

Info

- C:\Users\veera.kasina\AppData\Local\Temp\testng-eclipse-2148161\testng-customsuite.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

- 2 methods, 2 passed
- Passed methods ([show](#))

Be the first to know. Stay the leader.

Thank you!



PROPRIETARY AND CONFIDENTIAL