**13) Explain what does @Test(invocationCount=?) and (threadPoolSize=?) indicates?**

* **@Test (threadPoolSize=?):** The threadPoolSize attributes tell TestNG to form a thread pool to run the test method through multiple threads. With threadpool, the running time of the test method reduces greatly.
* **@Test(invocationCount=?):** The invocationcount tells how many times TestNG should run this test method

***TestNG*** is a testing framework inspired from ***JUnit*** and ***NUnit*** but introducing some new functionality that make it more powerful and easier to use.

It is an open source automated testing framework; where **NG** of Test**NG** means **N**ext **G**eneration.

**Benefits of TestNG**

There are number of benefits but from Selenium perspective, major advantages of TestNG are :

1. It gives the ability to produce ***HTML Reports*** of execution
2. ***Annotations***made testers life easy
3. Test cases can be ***Grouped & Prioritized*** more easily
4. ***Parallel***testing is possible
5. Generates ***Logs***
6. Data ***Parameterization***is possible

## Annotations in TestNG

**@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.

**@BeforeTest**: The annotated method will be run before any test method belonging to the classes inside the tag is run.

**@AfterTest**: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.

**@BeforeGroups**: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

**@AfterGroups**: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

**@BeforeClass**: The annotated method will be run before the first test method in the current class is invoked.

**@AfterClass**: The annotated method will be run after all the test methods in the current class have been run.

**@BeforeMethod**: The annotated method will be run before each test method.for ex: if I want to launch my application every time when test starts ,

**@AfterMethod**: The annotated method will be run after each test method.

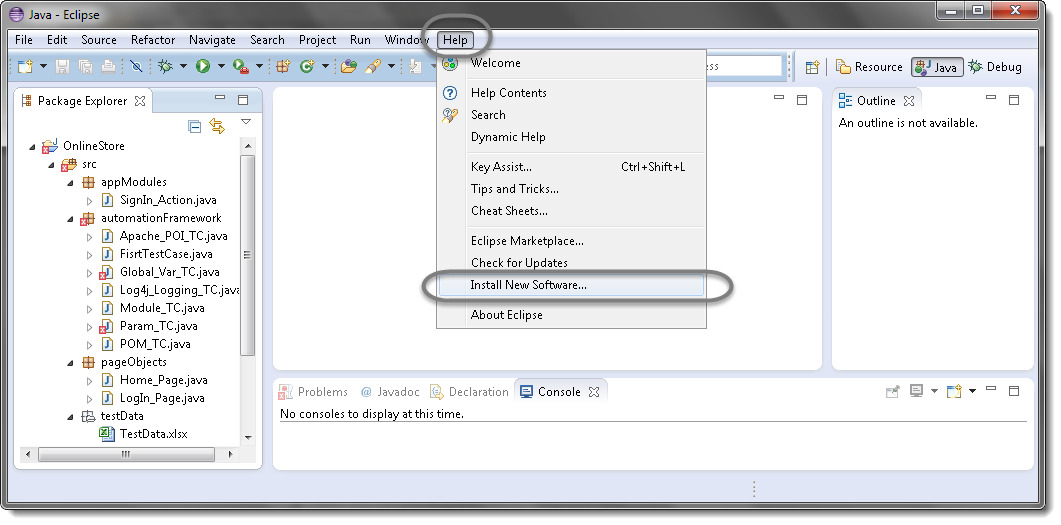
**@Test**: The annotated method is a part of a test case.

# Install TestNG in Eclipse

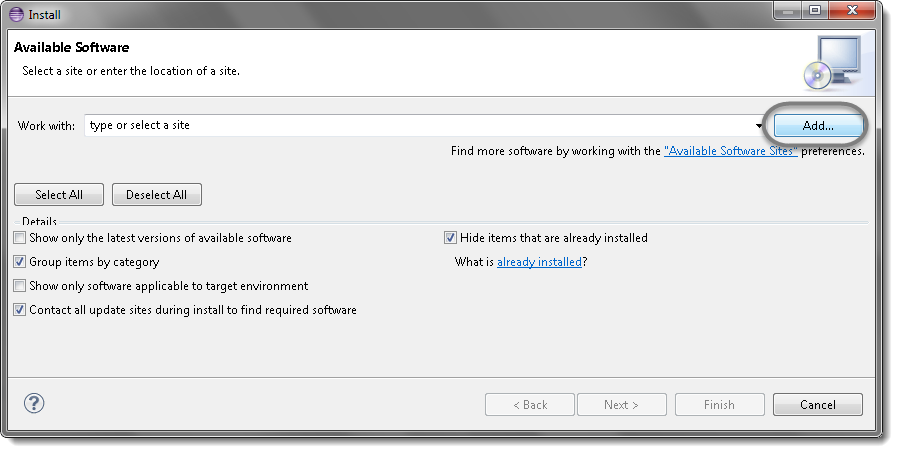
It is easy to install TestNG, as it comes as a plugin for Eclipse IDE. Prerequisite for installing TestNG is your Internet connection should be up & running during installation of this plugin and Eclipse IDE should be installed in your computer. Please see[***Download and Install Eclipse***](http://toolsqa.wpengine.com/selenium-webdriver/download-and-start-eclipse/) to setup Eclipse to you system.

## Steps to follow:

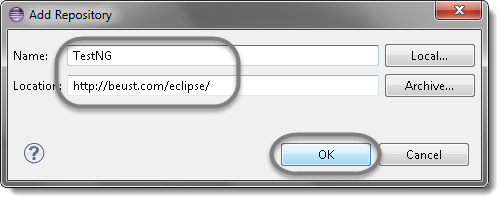
1) Launch the Eclipse IDE and from Help menu, click “**Install New Software**”.



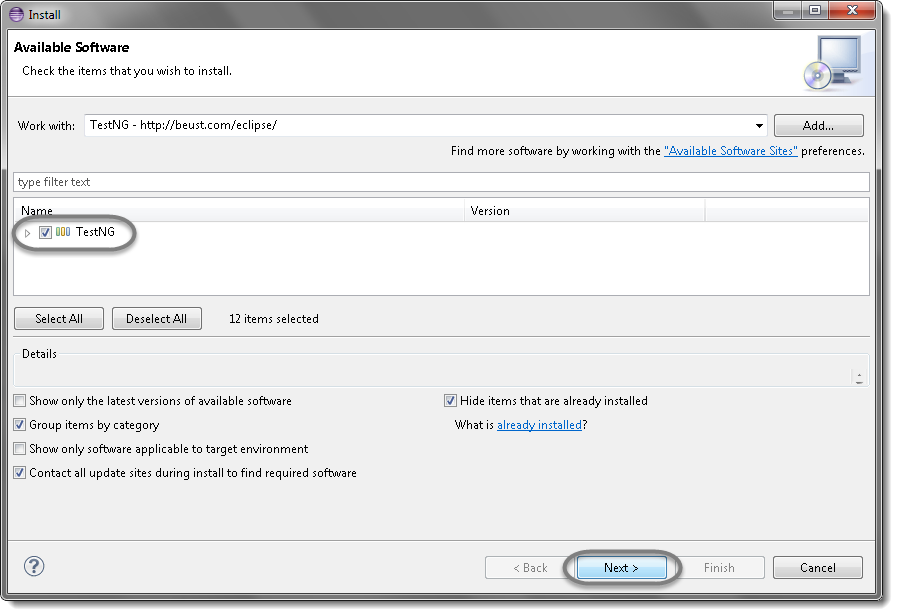
2) You will see a dialog window, click “**Add**” button.



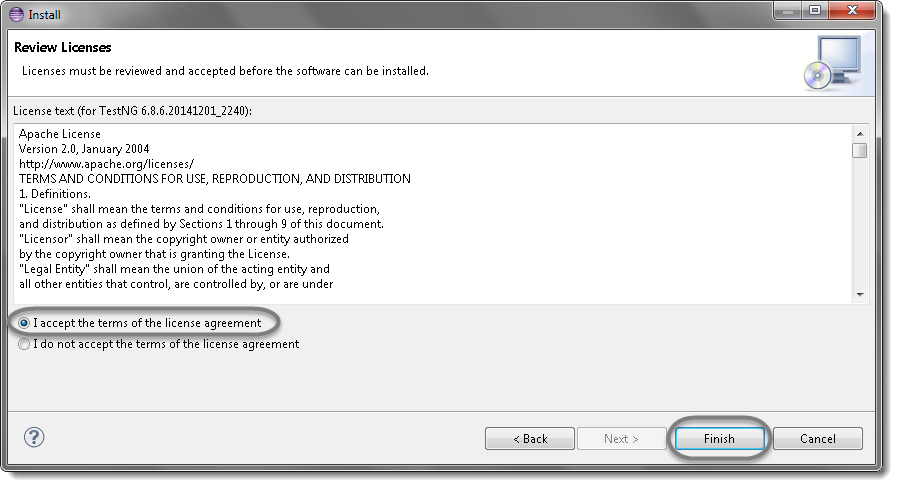
3) Type name as you wish, lets take “**TestNG**” and type “**http://beust.com/eclipse/**” as location. Click OK.



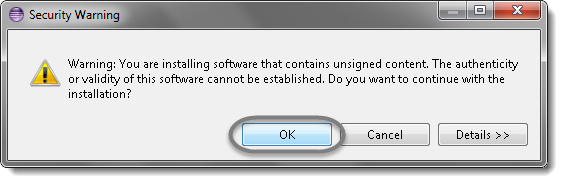
4) You come back to the previous window but this time you must see TestNG option in the available software list. Just Click TestNG and press “**Next**” button.



5) Click “**I accept the terms of the license agreement**” then click **Finish**.

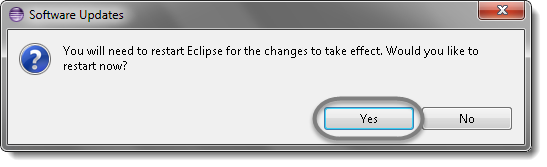


6) You may or may not encounter a Security warning, if in case you do just click **OK**.



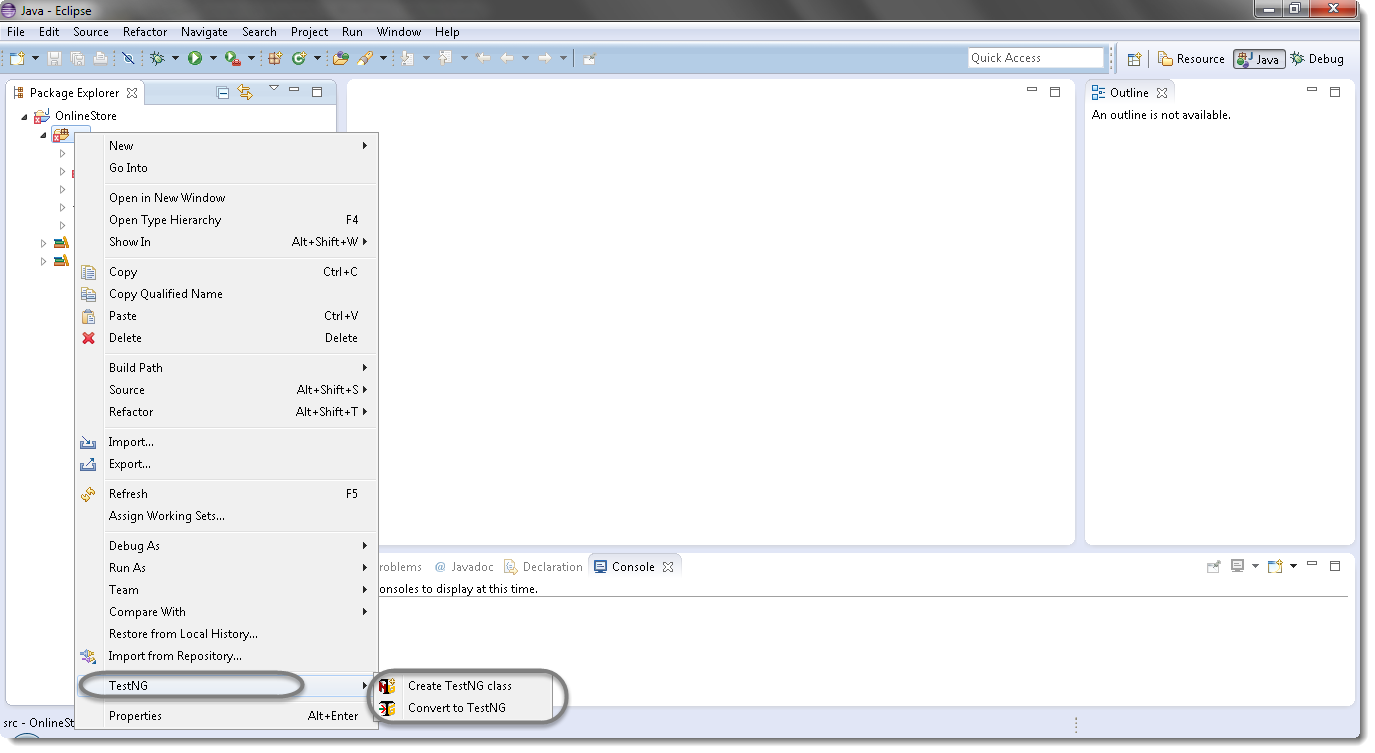
7) Click **Next** again on the succeeding dialog box until it prompts you to Restart the Eclipse.

8) You are all done now, just Click **Yes**.



9) Proceed with your workplace.

10) After restart, verify if TestNG was indeed successfully installed. Right click on you project and see if **TestNG** is displayed in the opened menu.



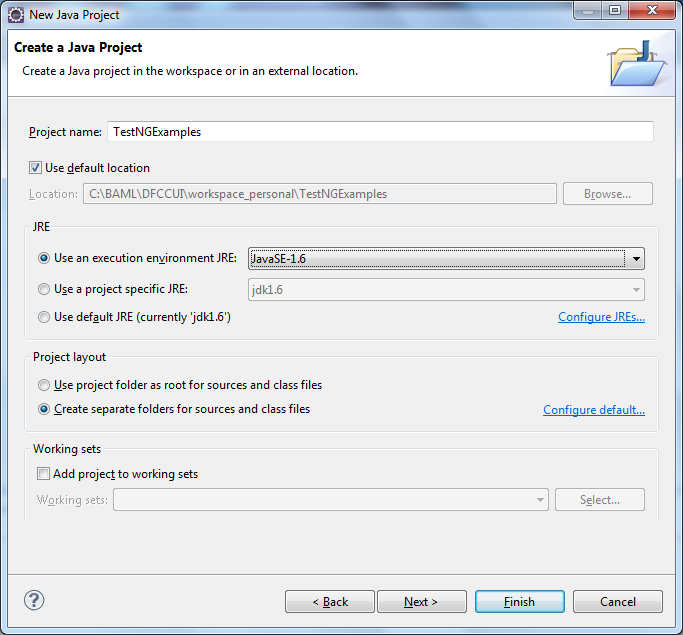
## Creating Java Project with TestNG Dependencies

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

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

2) **Select Java Project** as shown in the following screenshot and click on Next.

3) On the next screen, **enter a Project name** for a Java project, let’s say TestNGExamples, as shown in the following screenshot, and click on Finish:

s

Bottom of Form

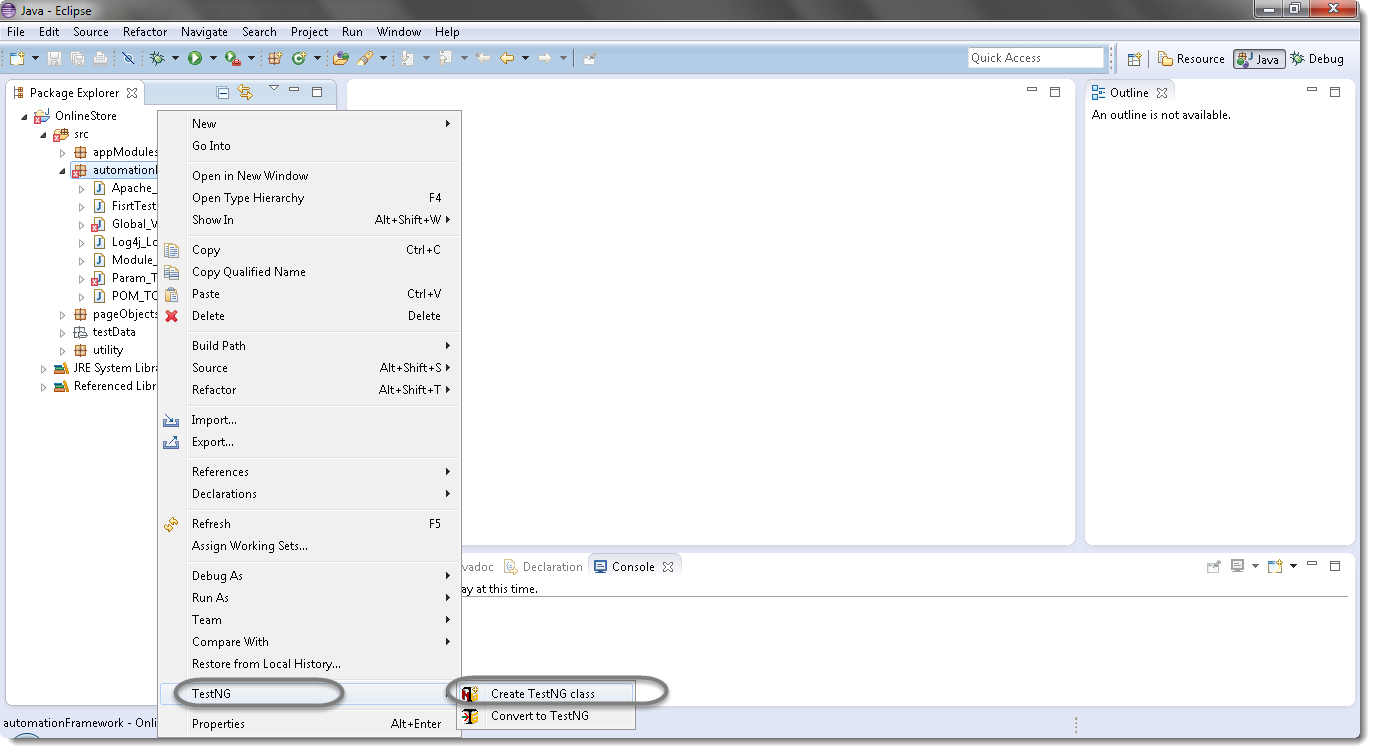
# [First Test case with TestNG](http://toolsqa.wpengine.com/selenium-training/)

## [Steps to follow:](http://toolsqa.wpengine.com/selenium-training/)

[1) Press](http://toolsqa.wpengine.com/selenium-training/) **[Ctrl+N](http://toolsqa.wpengine.com/selenium-training/)** [, select “](http://toolsqa.wpengine.com/selenium-training/)**[TestNG Class](http://toolsqa.wpengine.com/selenium-training/)**[” under](http://toolsqa.wpengine.com/selenium-training/) **[TestNG](http://toolsqa.wpengine.com/selenium-training/)** [category and click](http://toolsqa.wpengine.com/selenium-training/) **[Next](http://toolsqa.wpengine.com/selenium-training/)**[.](http://toolsqa.wpengine.com/selenium-training/)

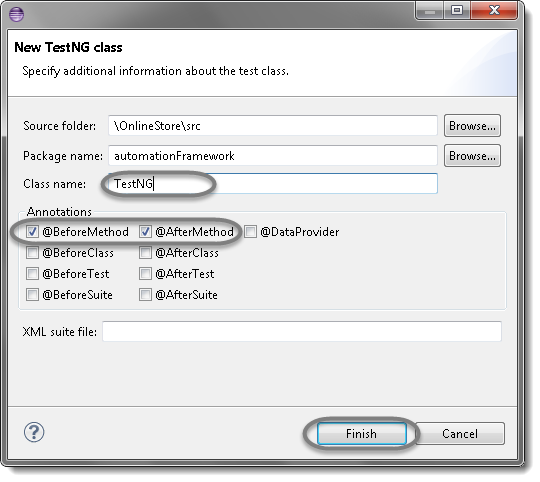
**[Or](http://toolsqa.wpengine.com/selenium-training/)**

[Right click on Test Case folder, go to](http://toolsqa.wpengine.com/selenium-training/) **[TestNG](http://toolsqa.wpengine.com/selenium-training/)** [and select “](http://toolsqa.wpengine.com/selenium-training/)**[TestNG Class](http://toolsqa.wpengine.com/selenium-training/)**[“.](http://toolsqa.wpengine.com/selenium-training/)

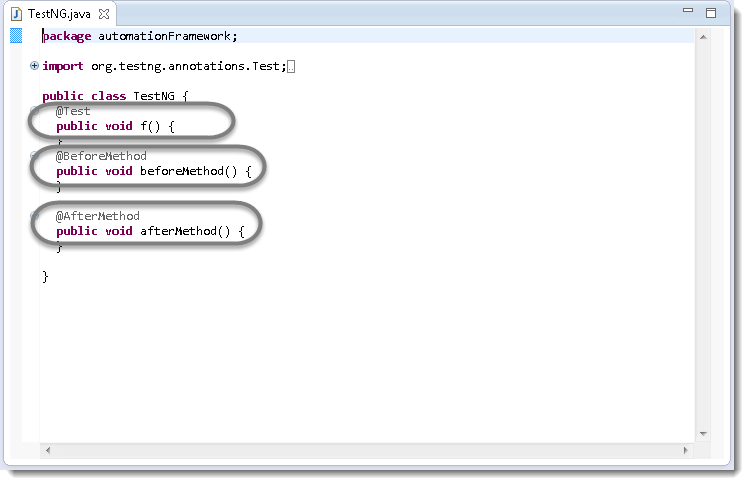
[](http://toolsqa.wpengine.com/selenium-training/)

[2) If your project is set up and you have selected the Test Case folder before creating TestNG class then the source folder and the package name will be prepopullated on the form. Set class name as ‘](http://toolsqa.wpengine.com/selenium-training/)**[TestNG](http://toolsqa.wpengine.com/selenium-training/)**[‘.](http://toolsqa.wpengine.com/selenium-training/)

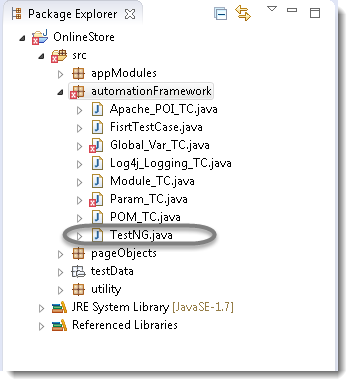
[Under Annotations, check “](http://toolsqa.wpengine.com/selenium-training/)**[@BeforeMethod](http://toolsqa.wpengine.com/selenium-training/)**[”, “](http://toolsqa.wpengine.com/selenium-training/)**[@AfterMethod](http://toolsqa.wpengine.com/selenium-training/)**[” and click](http://toolsqa.wpengine.com/selenium-training/) **[Finish](http://toolsqa.wpengine.com/selenium-training/)**[. That’s it.](http://toolsqa.wpengine.com/selenium-training/)

[](http://toolsqa.wpengine.com/selenium-training/)

[3) Now it will display the newly created TestNg class under the Test Case package(folder). TestNG class will look like the image below with displaying three empty methods. One method f() by default and before & after method, as selected during the creation of the class.](http://toolsqa.wpengine.com/selenium-training/)

[](http://toolsqa.wpengine.com/selenium-training/)

[4) Project explorer will look like this with TestNG class.](http://toolsqa.wpengine.com/selenium-training/)

[](http://toolsqa.wpengine.com/selenium-training/)

[Now it is the time to write the first TestNG test case.](http://toolsqa.wpengine.com/selenium-training/)

[5) Let’s take an example of](http://toolsqa.wpengine.com/selenium-training/) [**[First Test Case](http://toolsqa.wpengine.com/selenium-training/)**](http://toolsqa.wpengine.com/selenium-webdriver/first-test-case/) [and divide the test case in to three parts .](http://toolsqa.wpengine.com/selenium-training/)

**[@BeforeMethod](http://toolsqa.wpengine.com/selenium-training/)** [: Launch Firefox and direct it to the Base URL](http://toolsqa.wpengine.com/selenium-training/)

**[@Test](http://toolsqa.wpengine.com/selenium-training/)** [: Enter Username & Password to Login, Print console message and Log out](http://toolsqa.wpengine.com/selenium-training/)

**[@AfterMethod](http://toolsqa.wpengine.com/selenium-training/)** [: Close Firefox browser package automationFramework;](http://toolsqa.wpengine.com/selenium-training/)

[import java.util.concurrent.TimeUnit;](http://toolsqa.wpengine.com/selenium-training/)

[import org.openqa.selenium.By;](http://toolsqa.wpengine.com/selenium-training/)

[import org.openqa.selenium.WebDriver;](http://toolsqa.wpengine.com/selenium-training/)

[import org.openqa.selenium.firefox.FirefoxDriver;](http://toolsqa.wpengine.com/selenium-training/)

[import org.testng.annotations.Test;](http://toolsqa.wpengine.com/selenium-training/)

[import org.testng.annotations.BeforeMethod;](http://toolsqa.wpengine.com/selenium-training/)

[import org.testng.annotations.AfterMethod;](http://toolsqa.wpengine.com/selenium-training/)

[public class TestNG {](http://toolsqa.wpengine.com/selenium-training/)

[public WebDriver driver;](http://toolsqa.wpengine.com/selenium-training/)

[@Test](http://toolsqa.wpengine.com/selenium-training/)

[public void main() {](http://toolsqa.wpengine.com/selenium-training/)

[// Find the element that's ID attribute is 'account'(My Account)](http://toolsqa.wpengine.com/selenium-training/)

[driver.findElement(By.id("account")).click();](http://toolsqa.wpengine.com/selenium-training/)

[// Find the element that's ID attribute is 'log' (Username)](http://toolsqa.wpengine.com/selenium-training/)

[// Enter Username on the element found by above desc.](http://toolsqa.wpengine.com/selenium-training/)

[driver.findElement(By.id("log")).sendKeys("testuser\_1");](http://toolsqa.wpengine.com/selenium-training/)

[// Find the element that's ID attribute is 'pwd' (Password)](http://toolsqa.wpengine.com/selenium-training/)

[// Enter Password on the element found by the above desc.](http://toolsqa.wpengine.com/selenium-training/)

[driver.findElement(By.id("pwd")).sendKeys("Test@123");](http://toolsqa.wpengine.com/selenium-training/)

[// Now submit the form. WebDriver will find the form for us from the element](http://toolsqa.wpengine.com/selenium-training/)

[driver.findElement(By.id("login")).click();](http://toolsqa.wpengine.com/selenium-training/)

[// Print a Log In message to the screen](http://toolsqa.wpengine.com/selenium-training/)

[System.out.println(" Login Successfully, now it is the time to Log Off buddy.");](http://toolsqa.wpengine.com/selenium-training/)

[// Find the element that's ID attribute is 'account\_logout' (Log Out)](http://toolsqa.wpengine.com/selenium-training/)

[driver.findElement(By.id("account\_logout"));](http://toolsqa.wpengine.com/selenium-training/)

[}](http://toolsqa.wpengine.com/selenium-training/)

[@BeforeMethod](http://toolsqa.wpengine.com/selenium-training/)

[public void beforeMethod() {](http://toolsqa.wpengine.com/selenium-training/)

[// Create a new instance of the Firefox driver](http://toolsqa.wpengine.com/selenium-training/)

[driver = new FirefoxDriver();](http://toolsqa.wpengine.com/selenium-training/)

[//Put a Implicit wait, this means that any search for elements on the page could take the time the implicit wait is set for before throwing exception](http://toolsqa.wpengine.com/selenium-training/)

[driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);](http://toolsqa.wpengine.com/selenium-training/)

[//Launch the Online Store Website](http://toolsqa.wpengine.com/selenium-training/)

[driver.get("http://www.onlinestore.toolsqa.wpengine.com");](http://toolsqa.wpengine.com/selenium-training/)

[}](http://toolsqa.wpengine.com/selenium-training/)

[@AfterMethod](http://toolsqa.wpengine.com/selenium-training/)

[public void afterMethod() {](http://toolsqa.wpengine.com/selenium-training/)

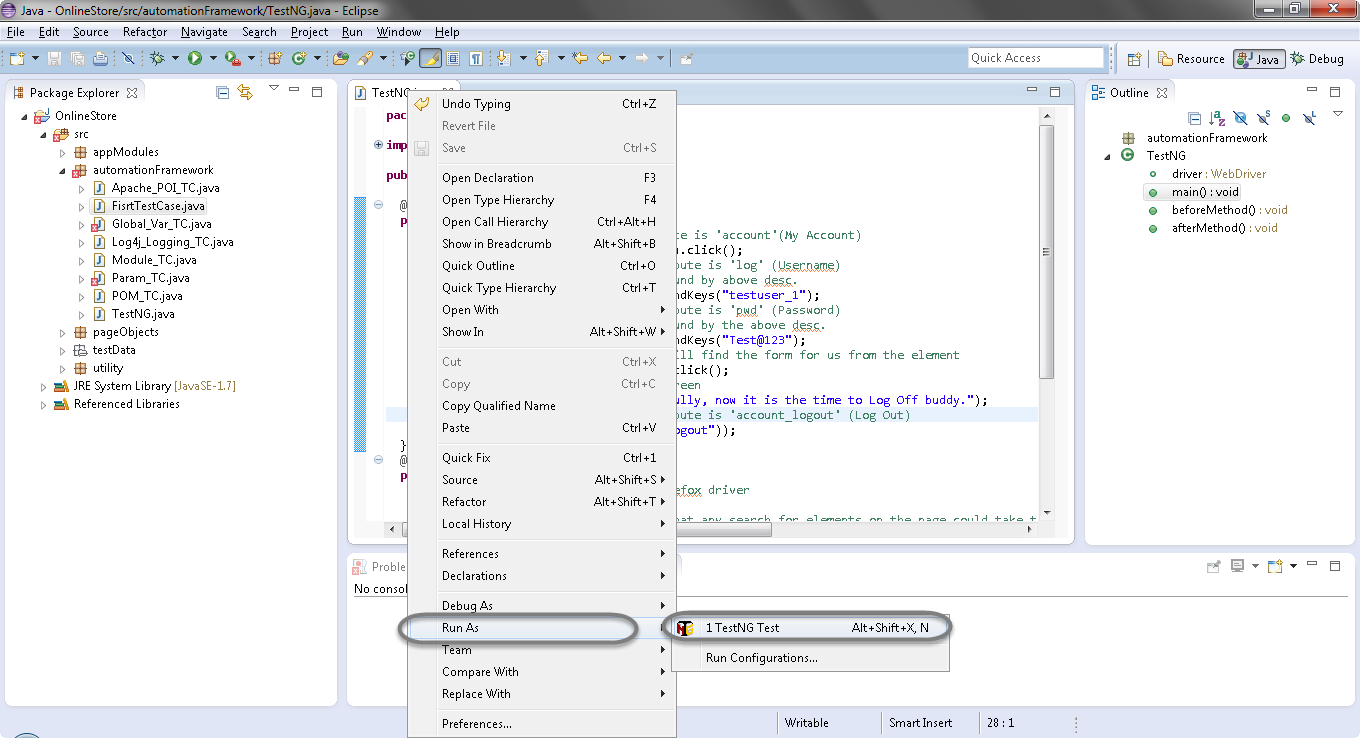
[// Close the driver](http://toolsqa.wpengine.com/selenium-training/)

[driver.quit();](http://toolsqa.wpengine.com/selenium-training/)

[}](http://toolsqa.wpengine.com/selenium-training/)

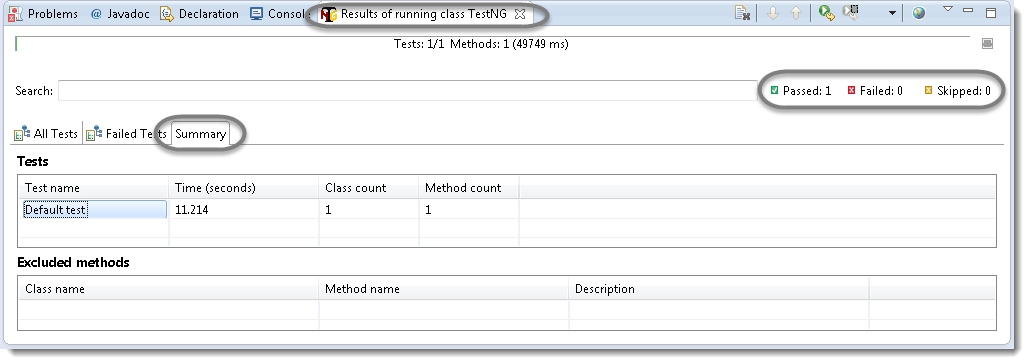
[}](http://toolsqa.wpengine.com/selenium-training/)

[6) Run the test by right click on the test case script and select](http://toolsqa.wpengine.com/selenium-training/) **[Run As](http://toolsqa.wpengine.com/selenium-training/)** [>](http://toolsqa.wpengine.com/selenium-training/) **[TestNG Test](http://toolsqa.wpengine.com/selenium-training/)**[.](http://toolsqa.wpengine.com/selenium-training/)

[](http://toolsqa.wpengine.com/selenium-training/)

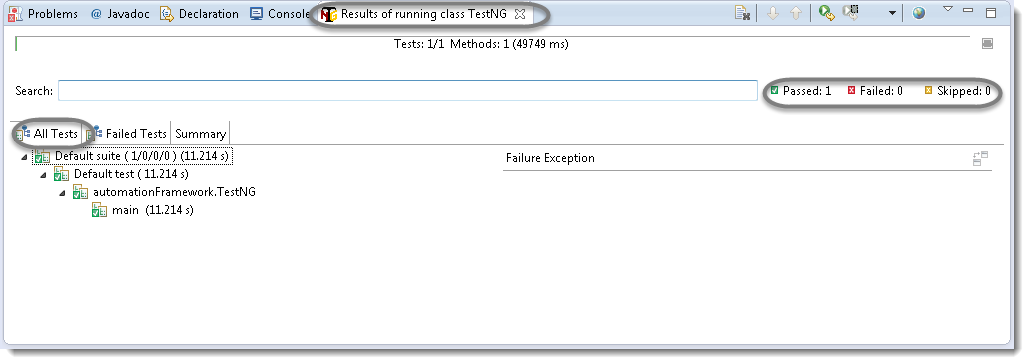
## ****[Results of running the Testng Test Case](http://toolsqa.wpengine.com/selenium-training/)****

[7) Give it few minutes to complete the execution, once it is finished the results will look like this in the](http://toolsqa.wpengine.com/selenium-training/) **[TestNg Result](http://toolsqa.wpengine.com/selenium-training/)** [window.](http://toolsqa.wpengine.com/selenium-training/)

[](http://toolsqa.wpengine.com/selenium-training/)

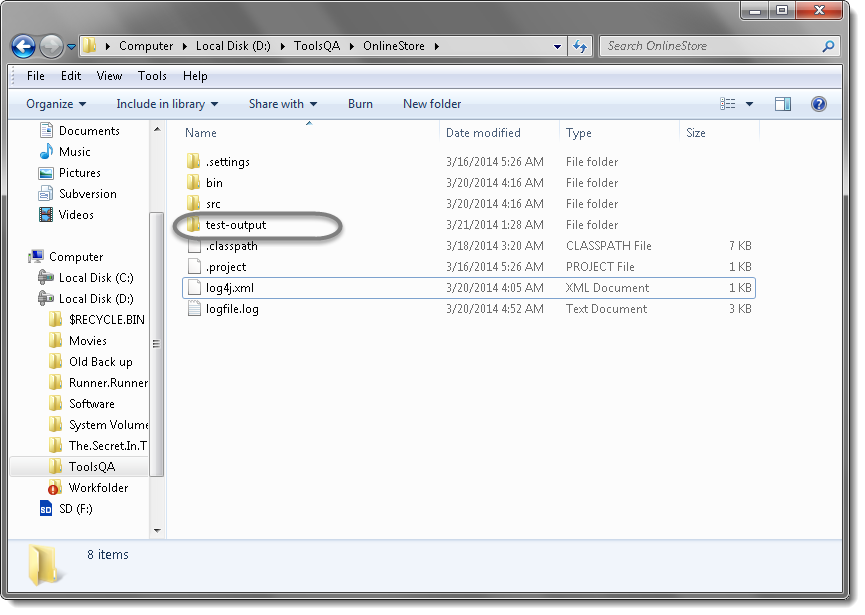
[It displayed ‘passed : 1’. This means test is successful and  Passed.](http://toolsqa.wpengine.com/selenium-training/)

[There are 3 sub tabs. “All Tests”, “Failed Tests” and “Summary”. Just click “All Tests” to see what is there.](http://toolsqa.wpengine.com/selenium-training/)

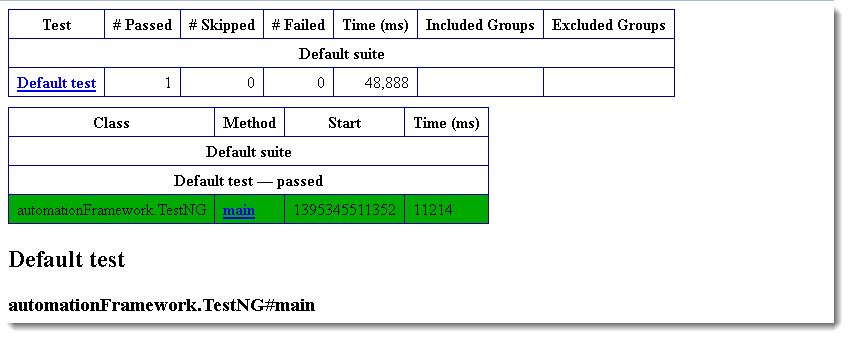
[](http://toolsqa.wpengine.com/selenium-training/)

[As you see, there is information of which test cases are executed and their duration. Take look to other tabs. Better than Junit right?](http://toolsqa.wpengine.com/selenium-training/)

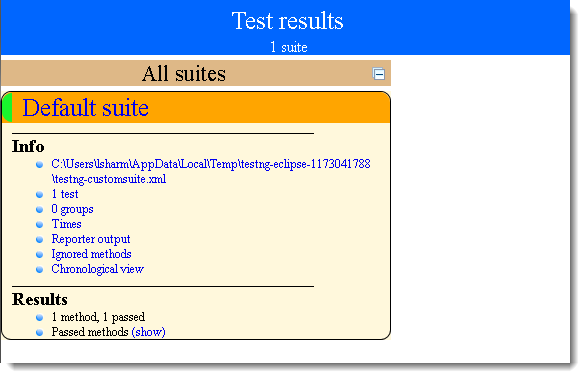
[8) TestNG also produce HTML reports. To access those reports go to the](http://toolsqa.wpengine.com/selenium-training/) **[Project](http://toolsqa.wpengine.com/selenium-training/)** [directory and open](http://toolsqa.wpengine.com/selenium-training/) **[test-output](http://toolsqa.wpengine.com/selenium-training/)** [folder.](http://toolsqa.wpengine.com/selenium-training/)

[](http://toolsqa.wpengine.com/selenium-training/)

) Open ‘**emailable-report.html**‘, as this is a html report open it with browser.



10) TestNG also produce ‘**index.html**‘ report and it resides in the same **test-output** folder. This reports gives the link to all the different component of the TestNG reports like **Groups** & **Reporter Output**. On clicking these will display detailed descriptions of execution. In the advance chapter of TestNG we will go though each of the TestNG topics.



# TestNG Test Suite

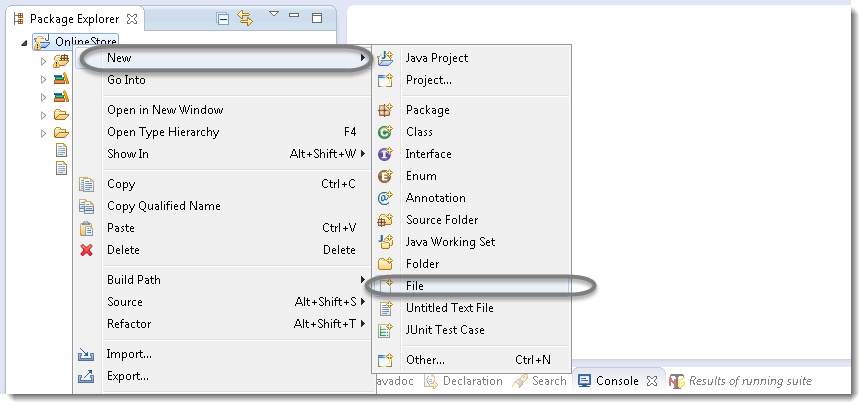
In any project, you will end up to a place where you need to execute so many test cases on a run. Running a set of test cases together is call executing a **Test Suite**.

In TestNG framework, we need to create **testng.xml** file to create and handle multiple test classes. This is the xml file where you will configure your test run, set test dependency, include or exclude any test, method, class or package and set priority etc.

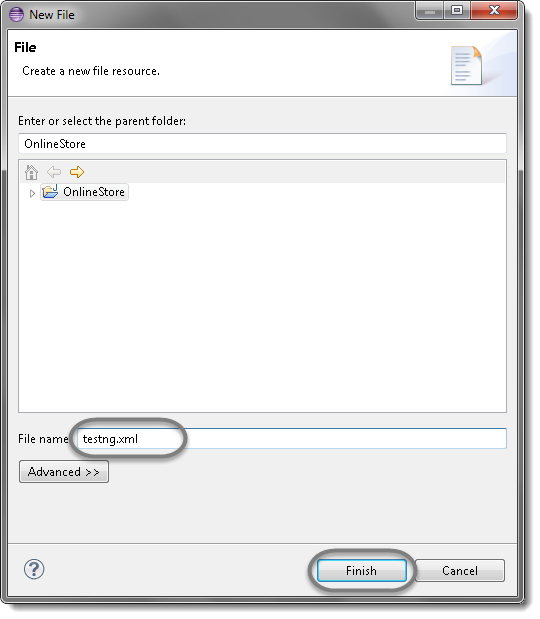
**How to do it…**

**Step 1 : Create a TestNG XML**

1) Right click on Project folder, go to **New** and select ‘**File**‘ as shown in below image.



2) In New file wizard, add file name = ‘**testng.xml**‘ as shown in below given image and click on **Finish** button.



3) It will add **testng.xml** file under your project folder.

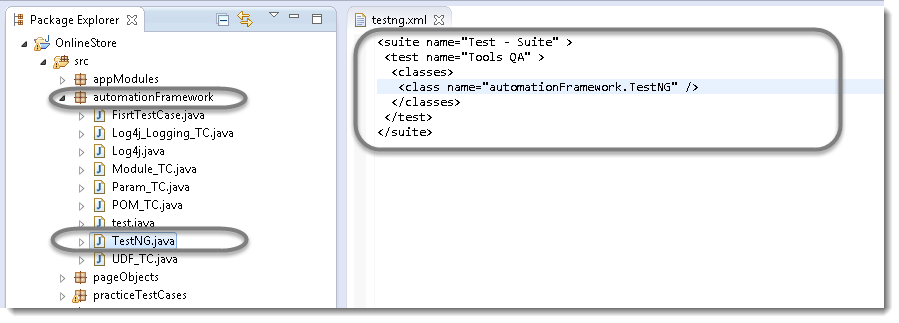
**Step 2 : Write xml code ?**

1) Now add below given code in your testng.xml file.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | &lt;suite name="Any Name" &gt;    &lt;test name="Any Name" &gt;    &lt;classes&gt; &lt;class name="PackageName.TestCaseName" /&gt;    &lt;/classes&gt;    &lt;/test&gt;    &lt;/suite&gt; |

**Note:** You can choose any name for your Test Suite & Test Name as per your need.

2) After giving appropriate names, now your testng.xml file will looks like this:

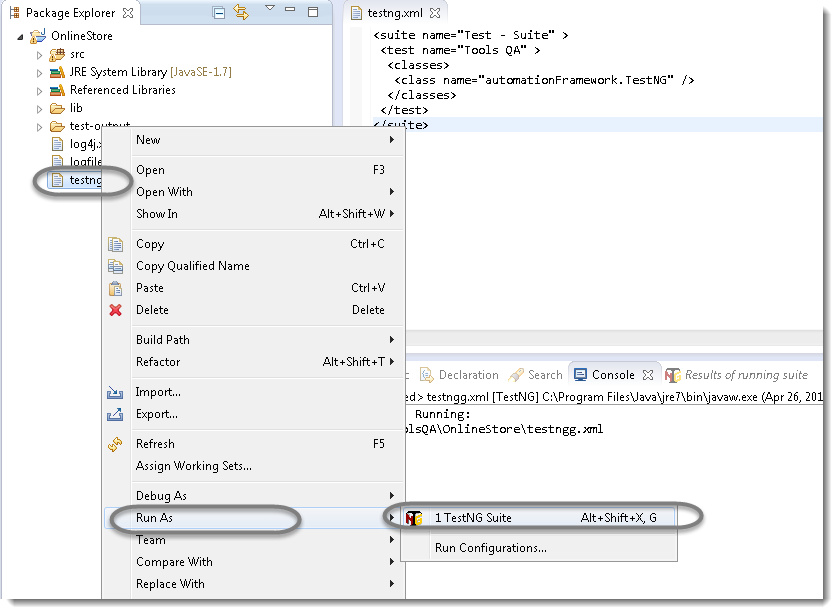


Hope you have understood the xml code, as it is quite simple hierarchy:

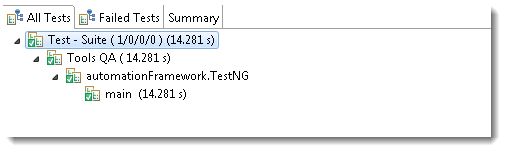
Very first tag is the Suite tag<suite>, under that it is the Test tag<test> and then the Class tag<classes>. You can give any name to the suite and the test but you need to provide the correct name to the <classes> tag which is a combination of your **Package** name and **Test Case** name.

**Step 3 : Execute a testng.xml**

Now it’s time to run the xml. Run the test by right click on the testng.xml file and select **Run As** > **TestNG Suite**.



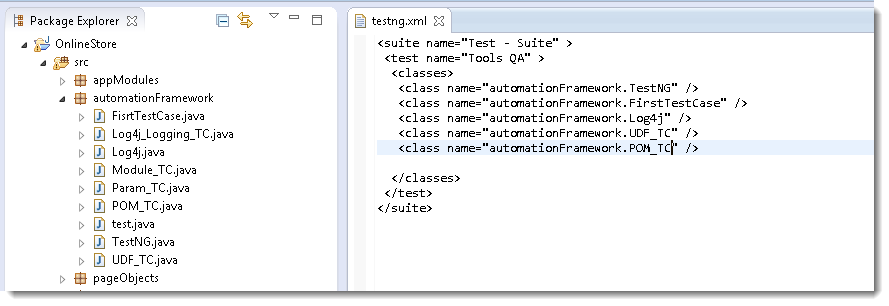
It will take few seconds to start the testng execution engine and soon you will notice that your test will run and complete. Once the execution is complete, you can view test execution result under the TestNg console.



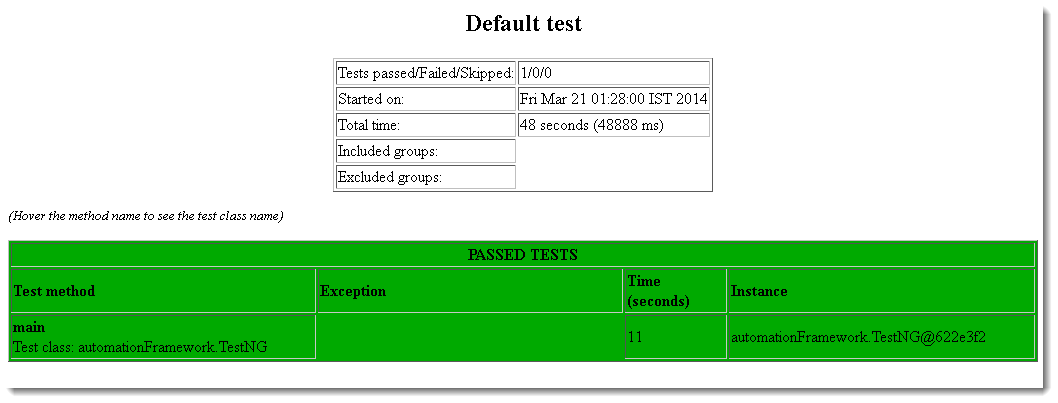
This is the one simple example of creating and running testng.xml file in eclipse.

# Building a Test Suite

Now when you have learned how to build the xml, now it’s time to learn how to build a Test Suite using testng.xml. It is again not a complex task, all you need to do is to add your test cases to your xml file in <classes> tag.



The above test will execute only those tests, which are mentioned in the testng.xml. The rest of the test cases under ‘automationFramework’ package will remain untouched.



# TestNG Annotations, Groups & OnDepends

<suite>

    <test>

 <classes>

 <method>

        <test>

     </method>

 </classes>

    </test>

 </suite>

It says that @Test is the smallest annotation here. @Method will be executed first, before and after the execution of @Test. The same way @Class will be executed first, before and after the execution of @Method and so on.

import org.testng.annotations.AfterMethod;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.BeforeSuite;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Sequencing {

@Test

public void testCase1() {

System.out.println("This is the Test Case 1");

}

@Test

public void testCase2() {

System.out.println("This is the Test Case 2");

}

@BeforeMethod

public void beforeMethod() {

System.out.println("This will execute before every Method");

}

@AfterMethod

public void afterMethod() {

System.out.println("This will execute after every Method");

}

@BeforeClass

public void beforeClass() {

System.out.println("This will execute before the Class");

}

@AfterClass

public void afterClass() {

System.out.println("This will execute after the Class");

}

@BeforeTest

public void beforeTest() {

System.out.println("This will execute before the Test");

}

@AfterTest

public void afterTest() {

System.out.println("This will execute after the Test");

}

@BeforeSuite

public void beforeSuite() {

System.out.println("This will execute before the Test Suite");

}

@AfterSuite

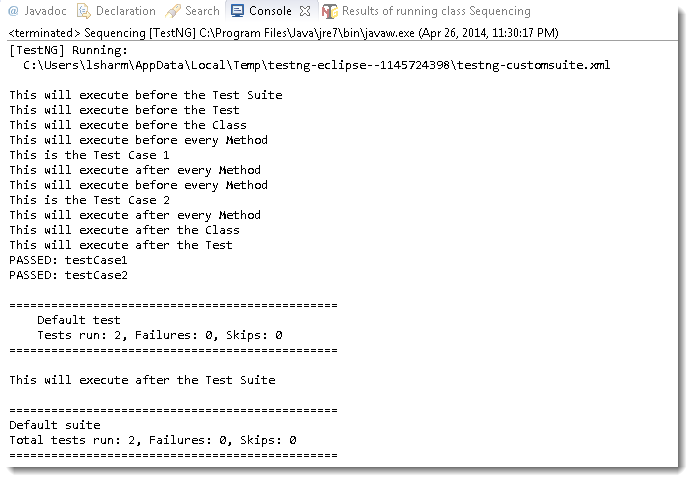
public void afterSuite() {

System.out.println("This will execute after the Test Suite");

}

}

Output of the above code will be like this:



It is clearly visible that the @Suite annotation is the very first and the very lastly executed. Then @Test followed by @Class. Now if you notice, the @Method has executed twice. As @Test is a method in the class, hence @Method will always executed for each @Test method.

## Test Case Grouping

‘**Groups**‘ is one more annotation of TestNG which can be used in the execution of multiple tests. Let’s say you have hundred tests of class vehicle and in it ten method of car, ten method of scooter and so on. You probably like to run all the scooter tests together in a batch. And you want all to be in a single test suite. With the help of grouping you can easily overcome this situation.

**How to do it…**

1) Create two methods for Car, two methods for Scooter and one method in conjunction with Car & Sedan Car.

2) Group them separately with using  (groups = { ” Group Name” })

import org.testng.annotations.Test;

public class Grouping {

  @Test (groups = { "Car" })

  public void Car1() {

  System.out.println("Batch Car - Test car 1");

  }

  @Test (groups = { "Car" })

  public void Car2() {

  System.out.println("Batch Car - Test car 2");

  }

  @Test (groups = { "Scooter" })

  public void Scooter1() {

  System.out.println("Batch Scooter - Test scooter 1");

  }

  @Test (groups = { "Scooter" })

  public void Scooter2() {

  System.out.println("Batch Scooter - Test scooter 2");

  }

  @Test (groups = { "Car", "Sedan Car" })

  public void Sedan1() {

  System.out.println("Batch Sedan Car - Test sedan 1");

  }

3) Create a testng xml like this:

<suite name="Suite">

    <test name="Practice Grouping">

        <groups>

    <run>

<include name="Car" />

    </run>

</groups>

<classes>

    <class name="automationFramework.Grouping" />

</classes>

    </test>

</suite>

**Note:**We have just call the group ‘Car’ from the xml and it also executed the test for Sedan Car, as we have mentioned the ‘Car’ as well while declaring the group of Sedan Car.

Clubbing of groups is also possible, take a look at the below xml:

<test name="Practice Grouping">

      <groups>

         <define name="All">

   <include name="Car"/>

   <include name="Scooter"/>

</define>

<run>

   <include name="All"/>

</run>

   </groups>

<classes>

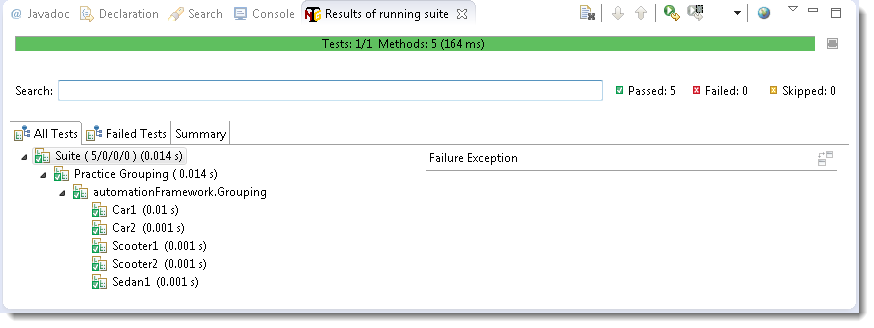
      <class name="automationFramework.Grouping" />

</classes>

   </test>

</suite>

You can see that we have created a new Group with the name ‘All’ and include other groups in it. Then simply called the newly created group for execution. The output will be like this:



## Dependent Test

Sometimes, you may need to invoke methods in a Test case in a particular order or you want to share some data and state between methods. This kind of dependency is supported by TestNG as it supports the declaration of explicit dependencies between test methods.

TestNG allows you to specify dependencies either with:

* Using attributes *dependsOnMethods* in @Test annotations OR
* Using attributes *dependsOnGroups* in @Test annotations.

Take a look over the below example:

import org.testng.annotations.Test;

public class Dependent {

  @Test (dependsOnMethods = { "OpenBrowser" })

  public void SignIn() {

  System.out.println("This will execute second (SignIn)");

  }

  @Test

  public void OpenBrowser() {

  System.out.println("This will execute first (Open Browser)");

  }

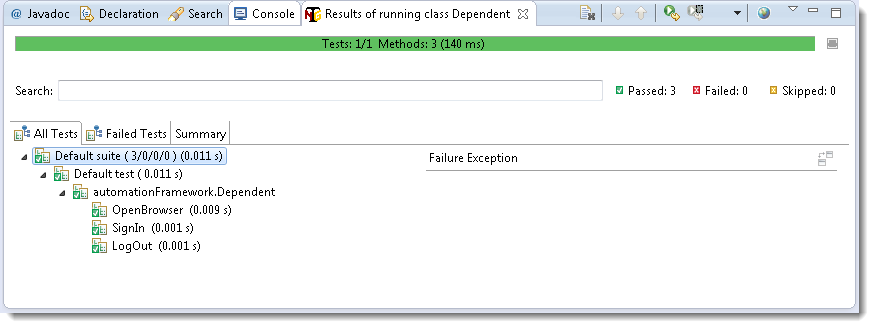
  @Test (dependsOnMethods = { "SignIn" })

  public void LogOut() {

  System.out.println("This will execute third (Log Out)");

  }

The output will be like this:



**Attention:**By default, methods annotated by @Test are executed alphabetically. Take a look over the next topic to see how to prioritize @Test.

## Sequencing & Prioritizing

You need to use the ‘**priority**‘ parameter, if you want the methods to be executed in your order. **Parameters** are keywords that modify the annotation’s function.

Let’s take the same above example and execute all @Test methods in right order. Simply assign priority to all @Test methods starting from 0(Zero).

import org.openqa.selenium.WebDriver;

import org.testng.annotations.Test;

public class MultipleTest {

public WebDriver driver;

  @Test(priority = 0)

  public void One() {

      System.out.println("This is the Test Case number One");

  }

  @Test(priority = 1)

  public void Two() {

  System.out.println("This is the Test Case number Two");

  }

  @Test(priority = 2)

  public void Three() {

  System.out.println("This is the Test Case number Three");

  }

  @Test(priority = 3)

  public void Four() {

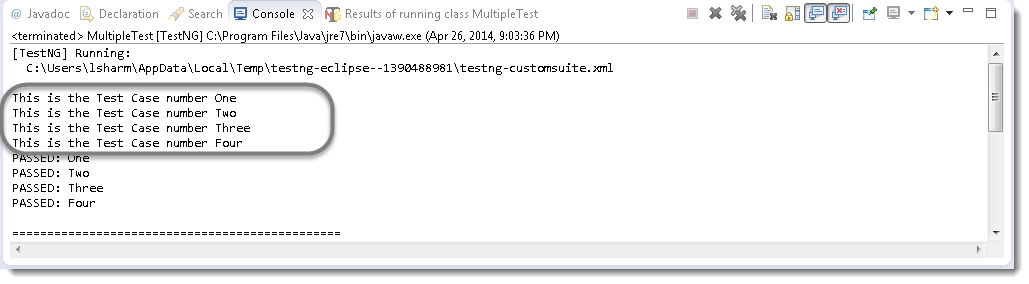
  System.out.println("This is the Test Case number Four");

  }

}

**Note:**TestNG will execute the @Test annotation with the lowest priority value up to the largest.

Output of the above:



## Skipping a Test Case

Think of a situation where you are required to skip one or more @Test from your testng class. In testng, you can easily able to handle this situation by setting the ‘enabled’ parameter to ‘false’ for e.g.:

@Test(enabled = false)

To use two or more parameters in a single annotation, separate them with a comma:

@Test(priority = 3, enabled = false)

Again take the same example and set the value false for the third test.

import org.openqa.selenium.WebDriver;

import org.testng.annotations.Test;

public class MultipleTest {

public WebDriver driver;

  @Test(priority = 0)

  public void One() {

      System.out.println("This is the Test Case number One");

  }

  @Test(priority = 1)

  public void Two() {

  System.out.println("This is the Test Case number Two");

  }

  @Test(priority = 2, enabled = false)

  public void Three() {

  System.out.println("This is the Test Case number Three");

  }

  @Test(priority = 3)

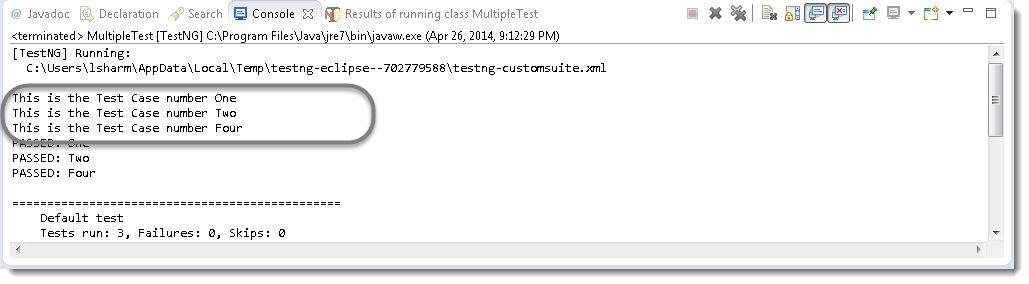
  public void Four() {

  System.out.println("This is the Test Case number Four");

  }

}

Output of the above example:



## TestNG Asserts

TestNG also gives us the power to take decisions in the middle of the test run with the help of **Asserts**. With this we can put various checkpoints in the test. Asserts are the most popular and frequently used methods while creating Selenium Scripts.In selenium there will be many situations in the test where you just like to check the presence of an element. All you need to do is to put an assert statement on to it to verify its existence.

**Different Asserts Statements**

**1) Assert.assertTrue() & Assert.assertFalse()**

 public void f() {

  driver = new FirefoxDriver();

      driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

      driver.get("http://www.store.demoqa.com");

      // Here driver will try to find out My Account link on the application

      WebElement myAccount = driver.findElement(By.xpath(".//\*[@id='account']/a"));

      //Test will only continue, if the below statement is true

      //This is to check whether the link is displayed or not

**Assert.assertTrue(myAccount.isDisplayed());**

      //My Account will be clicked only if the above condition is true

      myAccount.click();

   }

Assert true statement fails the test and stop the execution of the test, if the actual output is false.  Assert.assertFalse() works opposite of Assert.assertTrue(). It means that if you want your test to continue only if when some certain element is not present on the page. You will use Assert false, so it will fail the test in case of the element present on the page.

**2) Assert.assertEquals()**

 public void test() {

  String sValue = "Lakshay Sharma";

  System.out.println(" What is your full name");

  Assert.assertEquals("Lakshay Sharma", sValue);

  System.out.println(sValue);

  }

While running tests there can be cases where certain tests get stuck or may take much more time than expected. In such a case you may need to mark the said test case as fail and then continue. In this tutorial, we will learn to **configure TestNG tests to timeout**after some configured time duration.

**TimeOuts in Testng**

TestNG allows user to **configure a time period to wait for a test to completely execute**. Timeout can be configured in two ways:

* **At suite level:** This will be applicable for all the tests in the said TestNG test suite
* **At each test method level:** This will be applicable for the said test method and will override the time period if configured at the suite level

To specify timeout duration, use “***timeOut***” attribute of @Test annotation.

@Test ( timeOut = 500 )

Let’s create a sample test and learn how timeout works in TestNG.

**Example of Timeout Test at Suite Level**

In below test, we have two test methods i.e. timeTestOne() and timeTestTwo(). timeTestOne() will take 1000ms to execute completely whereas timeTestTwo() will take 400ms to execute completely. We have enforced the execution time using Thread.sleep() method.

|  |
| --- |
| public class TimeoutSuite  {      @Test      public void timeTestOne() throws InterruptedException {          Thread.sleep(1000);          System.out.println("Time test method one");      }        @Test      public void timeTestTwo() throws InterruptedException {          Thread.sleep(400);          System.out.println("Time test method two");      }  } |

Now add a testng.xml file to the project root and put the following code to it. This code defines timeout period to 500ms.

|  |
| --- |
| <suite name="Time test Suite" time-out="500" verbose="1" >    <test name="Timeout Test" >      <classes>        <class name="com.howtodoinjava.test.TimeoutSuite" />      </classes>    </test>  </suite> |

Now run above tests using testng.xml. Output of above test run is given below:

|  |
| --- |
| [TestNG] Running: C:\somepath\TestNGExamples\testng.xml    Time test method two    ===============================================  Time test Suite  Total tests run: 2, Failures: 1, Skips: 0  =============================================== |

As you can see from the test results, only timeTestTwo() for executed because it’s execution time was less than timeout time defined in testng.xml file. timeTestOne() execution got cancelled because it took more time to complete than timeout duration configured.

Let’s now go ahead and learn to set the timeout at a test method level.

**Example of Timeout Test at Method Level**

As mentioned earlier, you can specify the timeout at method level as well. This will give you flexibility to give appropriate time to run specific to each individual test method.

|  |
| --- |
| public class TimeoutMethod  {      @Test(timeOut = 500)      public void timeTestOne() throws InterruptedException {          Thread.sleep(1000);          System.out.println("Time test method one");      }        @Test(timeOut = 500)      public void timeTestTwo() throws InterruptedException {          Thread.sleep(400);          System.out.println("Time test method two");      }  } |
| [[TestNG] Running: C:\Users\somepath\testng-customsuite.xml    Time test method two  PASSED: timeTestTwo  FAILED: timeTestOne    org.testng.internal.thread.ThreadTimeoutException: Method org.testng.internal.TestNGMethod.timeTestOne() didn't finish within the time-out 500    ===============================================      Default test      Tests run: 2, Failures: 1, Skips: 0  =============================================== |

**Question #1) What is the significance of <testng.xml> file?**

**Answer:** In a Selenium TestNG project, we use <testng.xml> file to configure the complete test suite into a single file. This file makes it easy to group all the test suites and their parameters in one file. It also gives the ability to pull out subsets of your tests or split several runtime configurations. Few of the tasks which we can group in the <testng.xml> file are as follows.

**1-** Can configure test suite comprising of multiple test cases to run from a single place.  
**2-** Can include or exclude test methods test execution.  
**3-** Can mark a group to include or exclude.  
**4-** Can pass parameters in test cases.  
**5-** Can add group dependencies.  
**6-** Can configure parallel test execution.  
**7-** Can add listeners.

**Question #2) How to pass parameter through <testng.xml> file to a test case?**

**Answer:**You can set the parameter using the below syntax in the <testng.xml> file.

XHTML



|  |  |
| --- | --- |
| 1 | <parameter name="browser" value="FFX" /> |

Here, name attribute represents the parameter name and value signifies the value of that parameter. Then we can use that parameter in the selenium WebDriver software automation test case using the bellow syntax.

Java



|  |  |
| --- | --- |
| 1 | @Parameters ({"browser"}) |

**Question #3) How to exclude a @Test method from a test case with two @Test methods? Is it possible?**

**Answer:**Yes, you need to add @Test method in the exclude tag of <testng.xml> file as mentioned below.

XHTML



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >  <suite name="Test Exclusion Suite">    <test name="Exclusion Test" >      <classes>        <class name="Your Test Class Name">         <methods>         <exclude name="Your Test Method Name To Exclude"/>        </methods>        </class>      </classes>    </test>  </suite> |

**Question #4) How to skip a @Test method from execution?**

**Answer:**You can use the below syntax inside @Test method to skip a test case from test execution.

Java



|  |  |
| --- | --- |
| 1 | throw new SkipException("Test Check\_Checkbox Is Skipped"); |

It will throw skip exception and @Test method will be ignored immediately from execution.

**Question #5) Can you arrange the below <testng.xml> tags from parent to child?**

**<test>  
<suite>  
<class>  
<methods>  
<classes>**

**Answer:** The <testng.xml> file will have the following structure.

* The parent tag in the <testng.xml> file is the <suite> tag.
* <suite> tag can include one or more <test> tags.
* <test> tag can include the <classes> tag.
* <classes> tag can include one or more <class> tags.
* <class> tag wraps the <methods> tag where we define the test methods to include or exclude.

Hence, the correct order of the TestNG tags would be.

**<suite>  
<test>  
<classes>  
<class>  
<methods>**

**Question #6) How to define the priority of @Test method? Also, mention its usage?**

**Answer:**

In your Selenium WebDriver project, you can set priority for TestNG @Test annotated methods as shown in the following example.

Java



|  |  |
| --- | --- |
| 1 | @Test(priority=0) |

Using priority, you can manage @Test method execution sequence as per your requirement. That means @Test method with priority = 0 will run 1st and @Test method with priority = 1 will execute 2nd and so on.

**Question #7) Can you specify any 6 assertions of TestNG to be used in a Selenium WebDriver software testing tool.**

**Answer:**There are multiple assertions available In TestNG but generally we use the following assertions in out test cases.

**1-** assertEquals  
**2-** assertNotEquals  
**3-** assertTrue  
**4-** assertFalse  
**5-** assertNull  
**6-** assertNotNull

**Question #8) Why soft assertion is used in Selenium WebDriver and TestNG automation project?**

**Answer:** TestNG soft assertion allows to continue the test execution even if the assertion is failed. That means once the soft assertion fails, remaining part of the <@Test> method is executed and the assertion failure is reported at the end of the <@Test> method.

**Question #9) How to apply regular expression in <testng.xml> file to find @Test methods containing “product” keyword?**

**Answer:**

Refer below example, here we’ve used a regular expression to find @Test methods containing keyword “product”.

XHTML



|  |  |
| --- | --- |
| 1  2  3 | <methods>       <include name=".\*product.\*"/>  </methods> |

# TestNG XML example to execute with package names

In testng.xml file we can specify the specific package name (s) which needs to be executed.  
In a project there may be many packages, but we want to execute only the selected packages.  
The below is the example testng.xml which will execute the specific packages.  
Example:

<?xml version="1.0" encoding="UTF-8"?>

<**suite** name="example suite 1" verbose="1" >

<**test** name="Regression suite 1" >

   <**packages**>

  <**package** name="com.first.example" />

  </**packages**> </**test**> </**suite**>

**Question #10) What are the time unit we specify in test cases and test suites? minutes? seconds? milliseconds? or hours? Give Example.**

**Answer:** Time unit we specify at @Test method level and test suite level which normally set in milliseconds unit.

**Question #11) List out the benefits of TestNG over Junit?**

**Answer:**

TestNG framework has following benefits over JUnit.

**1-** TestNG annotations are more logical and easier to understand.

**2-** Unlike JUnit, TestNG does not require to declare @BeforeClass and @AfterClass.

**3-** There is no method name constraint in Selenium TestNG framework.

**4-** TestNG supports three additional setups:

**4.1-** @Before/AfterSuite,

**4.2-** @Before/AfterTest, and

**4.3-** @Before/AfterGroup.

**5-** In Selenium TestNG projects, there is no need to extend any class.

**7-** In TestNG, it is possible to run Selenium test cases in parallel.

**8-** TestNG supports grouping of test cases which is not possible in JUnit.

**9-** Based on the group, TestNG allows you to execute the test cases.

**10-** TestNG permits you to determine the dependent test cases. Every test case is autonomous to other test cases.

**Question #12) What are the basic steps for drafting TestNG test cases?**

**Answer:**

Following are the most common steps for writing TestNG test cases.

**1-** Write down the business logic of your test.

**2-** Add appropriate TestNG annotations in your code.

**3-** In <build.xml> or <testing.xml>, add the information about your test.

**4-** Run your TestNG project.

**Question #13) List out different ways to run TestNG?**

**Answer:**

You can run TestNG in the following ways.

**1-** Start directly from the Eclipse IDE, or

**2-** Run using the IntelliJ’s IDEA IDE.

**3-** Run with ant build tool.

**4-** Launch from the command line.

**Question #14) In TestNG how can you disable a test?**

**Answer:**

To disable the test case, you can use the following annotation.

* @Test(enabled = false).

**Question #15) Explain what does the test timeout mean in TestNG?**

**Answer:** The timeout test in TestNG is nothing but the time allotted to perform unit testing. If the unit test fails to finish in that specific time limit, TestNG will abandon further testing and mark it as a failed.

**Question #16) What is exception test and why is it used for?**

**Answer:**

TestNG provides an option for tracing the Exception handling of code. You can verify whether a code throws the desired exception or not. The expectedExceptions parameter is availed along with @Test annotation. Please refer the below example for clarity.

Java



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package com.techbeamers.testng.examples.exception;    import org.testng.annotations.Test;    public class TestRuntime {    @Test(expectedExceptions = { IOException.class })    public void exceptionTestOne() throws Exception {    throw new IOException();    }    @Test(expectedExceptions = { IOException.class, NullPointerException.class })    public void exceptionTestTwo() throws Exception {    throw new Exception();    }    } |

**Question #17) Explain what is parametric testing?**

**Answer:**

Parameterized testing lets the programmer re-run the same test with different values. TestNG allows you to pass parameters directly to the test methods using the two ways as given below.

**1-** With testing.xml.

**2-** With Data Providers.

**Question #18) Explain what does @Test(invocationCount=?) and (threadPoolSize=?) indicates?**

**Answer:**

**1- @Test (threadPoolSize=?):** The threadPoolSize attribute directs TestNG to create a thread pool to run the test method through multiple threads. With thread pool, the running time of the test method reduces considerably.

**2- @Test(invocationCount=?):** The invocation count refers to the no. of times TestNG should run the test method.

**Question #19) What are the different ways to produce reports for TestNG results?**

**Answer:**

TestNG offers following two ways to produce a report.

**Listeners:** For a listener class, the class has to implement the org.testng./TestListener interface. TestNG notifies these classes at runtime when the test enters into any of the below states.  
e.g. Test begins, finishes, skips, passes or fails.

**Reporters:** For a reporting class to implement, the class has to implement an org.testng/Reporter interface. When the whole suite run ends, these classes are called. When called, the object consisting the information of the whole test run is delivered to this class.

**Question #20) How does TestNG allow you to state dependencies?**

**Answer:**

TestNG supports two ways to declare the dependencies.

**1-** Using attributes dependsOnMethods in @Test annotations.

**2-** Using attributes dependsOnGroups in @Test annotations.

## [Execute Testng.xml using batch file](http://www.seleniumeasy.com/testng-tutorials/how-to-run-testng-xml-via-batch-file-example)

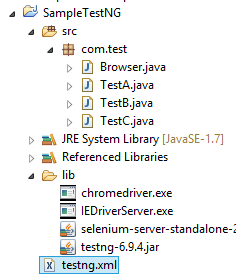
Now let us see how to invoke testng.xml from command line step by step:

**Step 1:** Create new project, In example we have created project 'SampleTestNG'

**Step 2:** Create new package. 'com.test' as example package created here.

**Step 3:** Create sample classes which has @Test methods. In the example below we have created four classes as below :

Step 4: Create a testing.xml and add the necessary field to run single testcase



7:For both batch file exicutions and command prompt exicutions use the below code

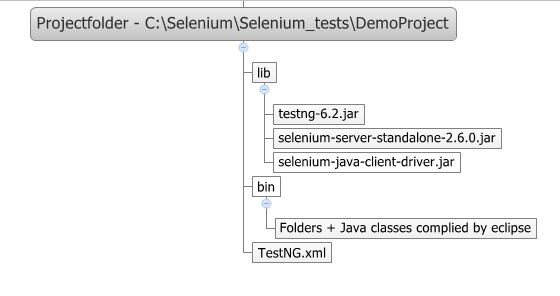
**set projectLocation=D:\AppiumHybridFrameWork\TestNg**

**cd %projectLocation%**

**set classpath=%projectLocation%\bin;%projectLocation%\Lib\\***

**java org.testng.TestNG %projectLocation%\testng.xml**

pause



# Parameterization in TestNG using testng.xml

TestNG allows the user to pass values to test methods as arguments by using parameter annotations through testng.xml file.

Some times it may be required for us to pass values to test methods during run time. Like we can pass user name and password through testng.xml instead of hard coding it in testmethods. or we can pass browser name as parameter to execute in specific browser.

**public** **class** ParameterPass {

@Parameters({"username" , "Password"})

@Test

**public** **void** parameter(String username , String Possword){

System.*out*.println("Pass the paramenter for USerName :" +username);

System.*out*.println("Pass the parameter for passwrod: " +Possword);

}

# }

Create a testing.xml file on the root folder

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<test name=*"Test"*>

<parameter name = *"username"* value = *"mohangowda"*/>

<parameter name =*"Password"* value =*"chandangowda"*/>

<classes>

<class name=*"CommandPrompt"*/>

<class name =*"ParameterPass"*/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

Output:

[TestNG] Running:

D:\AppiumHybridFrameWork\TestNg\testng.xml

Hello How are u

Pass the paramenter for USerName :mohangowda

Pass the parameter for passwrod: chandangowda

===============================================

Suite

Total tests run: 2, Failures: 0, Skips: 0

# Optional annotation in TestNG

As we know, we can pass parameter values to the test methods during run time from testng xml file by specifying [Parameters annotation to test method.](http://seleniumeasy.com/testng-tutorials/parameterization-in-testng)

To do this, we need to declare parameters tag in xml file using 'name' and 'value' attribute.Where the name attribute of the tag defines name of the parameter and the value attribute defines the value of the parameter.

If defined parameter is not found in your testng.xml file, The test method will receive the default value which is specified inside the @Optional annotation.

**Syntax to define Parameter :**

<parameter name="param" **value**="First parameter" />

**Syntax to define @Optional annotation**

@Parameters("browser")

@Test

**public** **void** **openBrowser**(@**Optional**("firefox") String value) { ... }

ex:

@Parameters({"secpar"})

@Test

**public** **void** secparam(@Optional("twinkle") String secpar){

System.*out*.println("The second para should be printed " +secpar);

}

Testing.xml does not have paramenter then it takes the default value of “Twinkle” in the above example.

<suite name=*"Suite"*>

<test name=*"Test"*>

<parameter name = *"username"* value = *"mohangowda"*/>

<parameter name =*"Password"* value =*"chandangowda"*/>

Here the parameter for secpar not defined so it takes twinkle as the default value

<classes>

<class name=*"CommandPrompt"*/>

<class name =*"ParameterPass"*/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

OutPut:

[TestNG] Running:

D:\AppiumHybridFrameWork\TestNg\testng.xml

Hello How are u

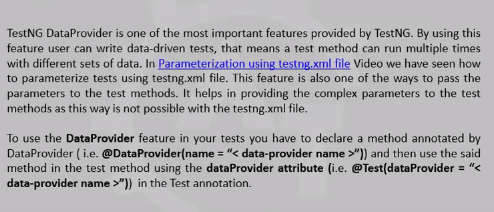
The second para should be printed twinkle

===============================================

Suite

Total tests run: 3, Failures: 0, Skips: 1

**Data Provider**



**public** **class** DataProviderTest {

@Test(dataProvider="Authentication")

**public** **void** dataprvider(String uname){

System.*out*.println(uname);

}

@DataProvider(name = "Authentication")

**public** **static** Object[][] credentials(){

**return** **new** Object[][]{

{"mohan"},{"chandan"}};

}

mohan

chandan

PASSED: dataprvider("mohan")

PASSED: dataprvider("chandan")

===============================================

Default test

Tests run: 2, Failures: 0, Skips: 0

===============================================

# Exception Test in TestNG

In TestNG we use expectedException with @Test annotation and we need to specify the type of exceptions that are expected to be thrown when executing the test methods.

A below example which throws an exception called “ArithmeticException” when dividing two numbers with denominator value as 0.

# @Test(expectedExceptions=ArithmeticException.class)

# **public** **void** **dividedByZeroExample1**()

# {

# **int** e = 1/0;

# }

when we execute the above code, the test method "dividedByZeroExample1" will return as “Passed” as we are handling the exceptions and the Test Method "dividedByZeroExample2" will return output as "Failed" with exception as '“java.lang.ArithmeticException: / by zero”

PASSED: test

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

# Skip Exception Test in TestNG

Using TestNG, we have multiple ways to Skip a test based on our requirement. We can Skip complete test without executing it or we can Skip a test when a specific condition is not satisfied.

In TestNG, *@Test(enabled=false)* annotation is used to skip a test case if it is not ready to test. We don't need to import any additional statements.

As in JUnit, TestNG will not show you the other test method as Skipped or Ignored. It will not consider that case method at all when the annotation is mentioned as “@Test(enabled=false)”

And We can Skip a test by using TestNG [*Skip Exception*](http://testng.org/javadoc/org/testng/SkipException.html) if we want to Skip a particular Test.  
Syntax:

**throw** **new** SkipException("message");

And we can also perform a ***CONDITIONAL Skip***, i.e.. We can have a condition to check and then SKIP test if condition is not satisfied. For Example, if there is no data available to perform a test, we can skip the test instead of making that test as failed.

Let us see all the above three cases by taking three simple tests.

Please find the below sample program.

**package** packOne;

**import** org.testng.SkipException;

**import** org.testng.annotations.Test;

**public** **class** **SkipExample** {

@Test(enabled=**false**)

**public** **void** **testCaseEnabling**(){

System.out.println("I'm Not Ready, please don't execute me");

}

@Test

**public** **void** **testCaseSkipException**(){

System.out.println("Im in skip exception");

**throw** **new** SkipException("Skipping this exception");

}

@Test

**public** **void** **testCaseConditionalSkipException**(){

System.out.println("Im in Conditional Skip");

**if**(!DataAvailable)

**throw** **new** SkipException("Skipping this exception");

System.out.println("Executed Successfully");

}

}

Once i execute the above program the output should of the program should look as below . Test *testCaseConditionalSkipException* will be skipped only when data is not available

# Preserver Order in Testng

In TestNg bydefault the preserve-order attribute will be set to 'true', this means, TestNG will run your tests in the order they are found in the XML file.

Create three classes as  
ClassOne.java  
ClassTwo.java  
ClassThree.java

Now we will define the xml file with preserve-order attribute for tests and set the value as 'false'.

<!DOCTYPE suite SYSTEM "[http://testng.org/testng-1.0.dtd">](http://testng.org/testng-1.0.dtd)

[<](http://testng.org/testng-1.0.dtd)**[suite](http://testng.org/testng-1.0.dtd)** name="Preserve order test runs">

  <**test** name="Regression 1" preserve-order="false">

    <**classes**>

      <**class** name="com.pack.preserve.ClassOne"/>

      <**class** name="com.pack.preserve.ClassTwo"/>

      <**class** name="com.pack.preserve.ClassThree"/>

    </**classes**>

  </**test**>

</**suite**>

As we have set the preserve-order attribute to false, test will not be executed in order. They will get executed in an unpredictable order. We have ClassOne, ClassTwo and ClassThree defined in xml, but the order that they executed are ClassOne, ClassThree, and ClassTwo. Check the output below:

# Log4J Logging

During the running of test case user wants some information to be logged in the console. Information could be any detail depends upon the purpose.

* A complete test steps to replicate the scenario
* Issue, Description of the failure or reason for the failed test case
* Time stamp for the developers to investigate the issue in detail

## How to do it…

1) [Download JAR files](http://toolsqa.wpengine.com/selenium-webdriver/download-log4j/) of Log4j and [Add Jars](http://toolsqa.wpengine.com/selenium-webdriver/add-log4j-jars/) to your project library. You can download it from [here](http://logging.apache.org/). That’s all about configuration of Apache POI with eclipse. Now you are ready to write your test.

2) Create a new **XML** file – **log4j.xml** and place it under the Project root folder.

3) Paste the following code in the **log4j.xml** file.

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/" debug="false">

<appender name="fileAppender" class="org.apache.log4j.FileAppender">

<param name="Threshold" value="INFO" />

<param name="File" value="logfile.log"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d %-5p [%c{1}] %m %n" />

</layout>

</appender>

<root>

<level value="INFO"/>

<appender-ref ref="fileAppender"/>

</root>

</log4j:configuration>

Then create a class and paste as below

**private** **static** Logger *Log* = Logger.*getLogger*(ReporterLogs.**class**.getName());

@Test

**public** **static** **void** test() {

DOMConfigurator.*configure*("log4j.xml");

*driver* = **new** FirefoxDriver();

*Log*.info("New driver instantiated");

*driver*.manage().timeouts().implicitlyWait(10, TimeUnit.*SECONDS*);

*Log*.info("Implicit wait applied on the driver for 10 seconds");

*driver*.get("http://www.store.demoqa.com");

*Log*.info("Web application launched");

// Our first step is complete, so we produce a main event log here for our reports.

Reporter.*log*("Application Lauched successfully | ");

*driver*.findElement(By.*xpath*(".//\*[@id='account']/a")).click();

*Log*.info("Click action performed on My Account link");

*driver*.findElement(By.*id*("log")).sendKeys("testuser\_1");

*Log*.info("Username entered in the Username text box");

*driver*.findElement(By.*id*("pwd")).sendKeys("Test@123");

*Log*.info("Password entered in the Password text box");

*driver*.findElement(By.*id*("login")).click();

*Log*.info("Click action performed on Submit button");

// Here we are done with our Second main event

Reporter.*log*("Sign In Successful | " );

/\*driver.findElement(By.id("account\_logout"));

Log.info("Click action performed on Log out link");

\*/

*driver*.quit();

*Log*.info("Browser closed");

// This is the third main event

Reporter.*log*("User is Logged out and Application is closed | ");

 }

# TestNG Parameters

It allows us to automatically run a test case multiple times with different input and validation values.

Create a Testng class with the parameters

Parameters({“username” , “Password”})

@Test

@Parameters({"userName" , "Password"})

**public** **void** Login(String userName1 , String Password1) {

System.*out*.println(" The username of the employee" +userName1);

System.*out*.println("The Password of the employee"+Password1);

Create a testing.xml and Pass the parameter

<parameter name = “userName" , value = “mohan>

<parameter name ="Password" , value =”ajinke”})

<suite name=*"Suite"* parallel=*"false"*>

<test name=*"Test"*>

<parameter name=*"userName"* value=*"testuser\_1"*/>

<parameter name=*"Password"* value=*"Test@123"*/>

<classes>

<class name=*"com.javatpoint.Parameterization"*/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->