**How to Install TestNG in Eclipse IDE for Selenium WebDriver**

## How To Install TestNG In Eclipse

Before going through how to install TestNG plug in in Eclipse IDE. Let’s see what is TestNG and the pre-requisites we need to install TestNG in Eclipse IDE.

### ****What is TestNG?****

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing.

### ****Pre-requisites to Install TestNG Eclipse:****

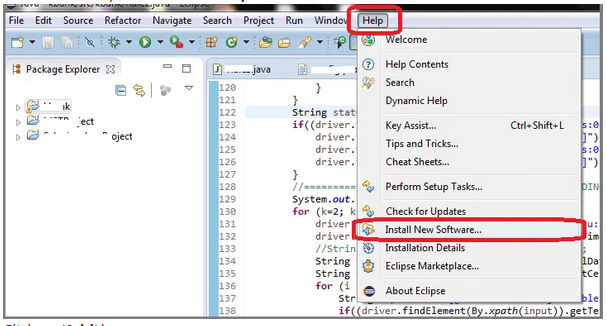
We need to have the following

i). **Selenium IDE**  
ii). **Active Internet Connection**

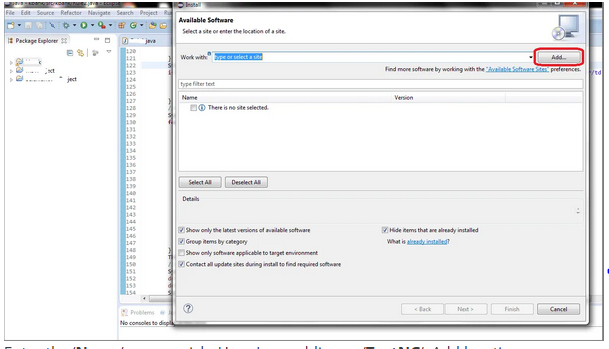
Follow the below steps to complete the installation.

Step 1: Open Eclipse IDE  
Step 2: Download and Install TestNG

1. Launch the Eclipse IDE. Go to ‘**Help**‘and click on ‘**Install New Software**‘



1. Click on ‘**Add**‘button

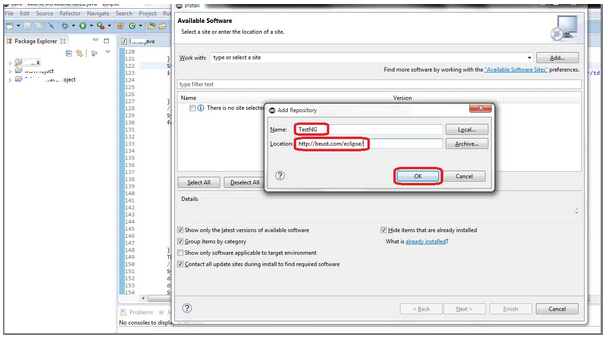


1. Enter the ‘**Name**‘as you wish. Here I am adding as ‘**TestNG**‘. Add location as “http://beust.com/eclipse/” and click on ‘**OK**‘ button.

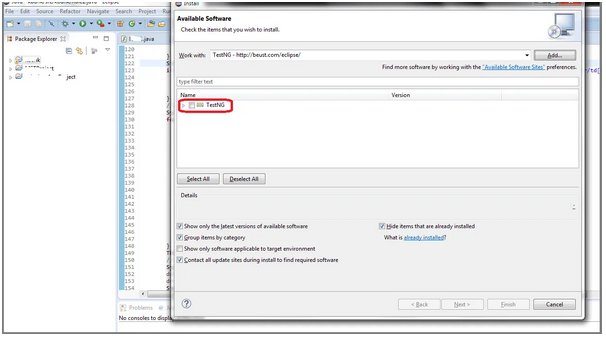
If the above link won’t work, use the beta version link “http://testng.org/eclipse-beta”

Note: You could always look back at TestNG site for the link (“<http://testng.org/doc/download.html>“)

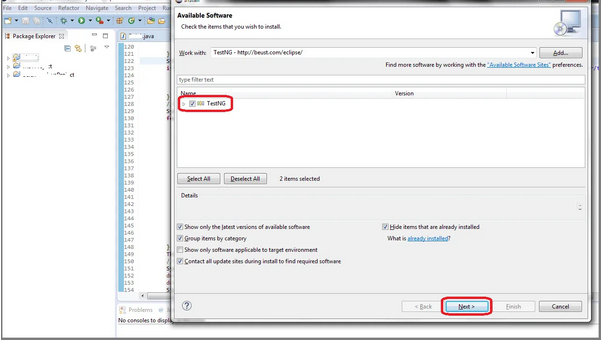
In the Available Software list, you could see **TestNG**option

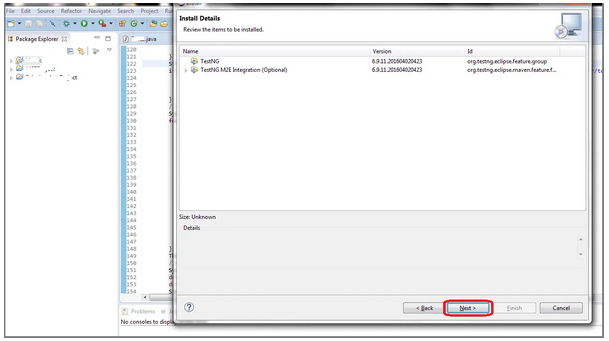


1. Just Click “**TestNG”** and press “**Next”** button.



1. Click on “**Next**” button

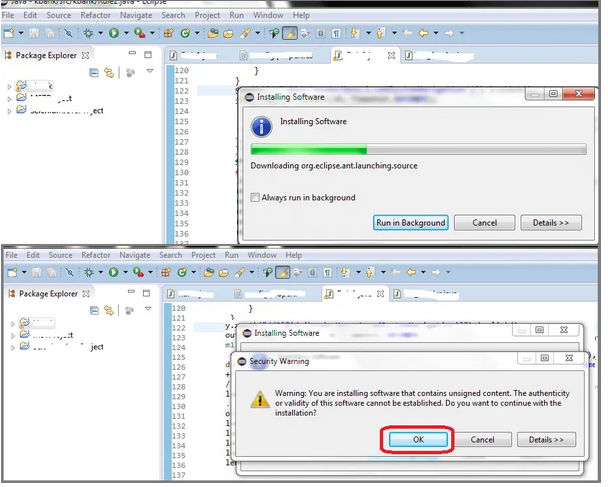




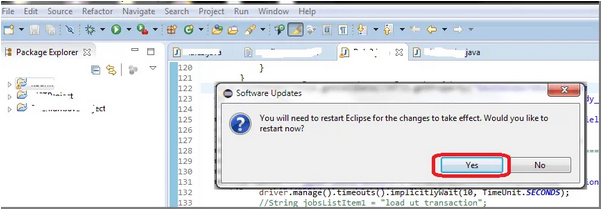
1. Click “**I accept the terms of the license agreement”** then click **Finish**.

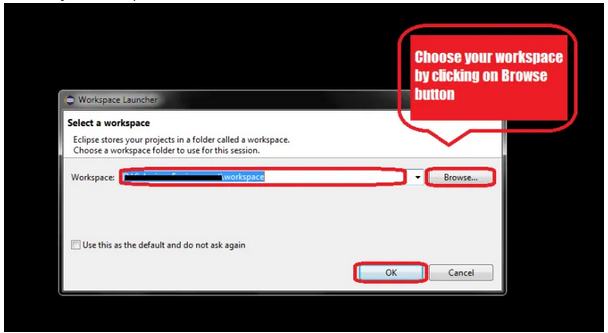


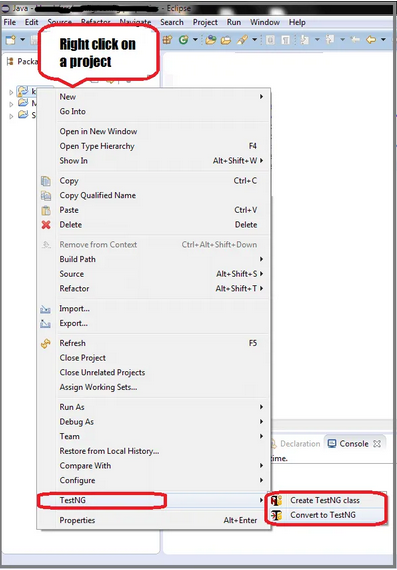
1. Installation is completely done. Just click “**Yes**“



1. Choose your workplace and click on “**OK**“.







After restart, Right click on your project to verify if TestNG was successfully installed or not. If it is successfully installed, you could see TestNG as shown below.

# TestNG Tutorial

## TestNG Tutorial – Complete Guide

In this TestNG Tutorial, you will learn how to work with the TestNG and implement it while working with Selenium.

**What is TestNG?**  
TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing. TestNG is a testing framework inspired from JUnit and NUnit. It has more functionalities compared to JUnit and NUnit and it makes more powerful and easier to use.

This TestNG Tutorial covers the following topics in detail with examples.

1. [TestNG Introduction](https://www.softwaretestingmaterial.com/testng-introduction/)
2. [TestNG Annotations And Benefits](https://www.softwaretestingmaterial.com/testng-annotations/)
3. [To Create TestNG.xml file](https://www.softwaretestingmaterial.com/create-testng-xml-file)
4. [Ignore Tests in TestNG](https://www.softwaretestingmaterial.com/how-to-ignore-testng-test/)
5. [Skip Test In TestNG](https://www.softwaretestingmaterial.com/how-to-skip-testng-test/)
6. [Groups](https://www.softwaretestingmaterial.com/testng-groups/)
7. [Exception](https://www.softwaretestingmaterial.com/testng-exception/)
8. [Dependencies](https://www.softwaretestingmaterial.com/testng-dependencies/)
9. [Parameterized Tests Using XML](https://www.softwaretestingmaterial.com/testng-parameterization-using-xml/)
10. [Parameterized Tests Using Data Providers](https://www.softwaretestingmaterial.com/testng-parameterization-using-dataproviders/)
11. [Parallel Execution in TestNG](https://www.softwaretestingmaterial.com/parallel-test-execution-testng/)
12. [Asserts](https://www.softwaretestingmaterial.com/testng-asserts/)
13. [Soft Assert](https://www.softwaretestingmaterial.com/soft-assert/)
14. [Listeners](https://www.softwaretestingmaterial.com/testng-listeners/)
15. [IRetryAnalyzer – How To Run Failed Test Cases](https://www.softwaretestingmaterial.com/run-failed-test-cases-using-testng/)
16. [TestNG Reports](https://www.softwaretestingmaterial.com/testng-reports/)
17. [How To Run TestNG Using Command Prompt](https://www.softwaretestingmaterial.com/run-testng-using-command-prompt/)

# TestNG Introduction

## TestNG Introduction:

Definition of TestNG as per its documentation is as follows:

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing.

Writing a test is typically a three-step process:

* Write the business logic of your test and insert TestNG annotations in your code.
* Add the information about your test (e.g. the class name, the groups you wish to run, etc…) in a testng.xml file or in build.xml.
* Run TestNG.xml.

TestNG is a framework for Java, so the very first requirement is to have JDK installed in your machine.

Verify Java Installation in your system

java version.PNG

Output:

2.PNG

**Set Java Environment:**

Set the **JAVA\_HOME** environment variable to point to the base directory location, where Java is installed on your machine. For example,

Set the environment variable **JAVA\_HOME** to C:\Program Files\Java\jdk1.8.0\_101

Append the string **C:\Program Files\Java\jdk1.8.0\_101\bin** at the end of the system variable, ‘**Path**‘.

TestNG can be invoked in three different ways:

* Using command prompt
* Using testng.xml file
* Using ANT

# TestNG Annotations And Benefits

## TestNG Annotations:

In this post, we see the list of TestNG Annotations. Here is a quick overview of the annotations available in TestNG.

**@Test:**Marks a class or a method as a part of the test.

**@BeforeMethod:** A method which is marked with this annotation will be executed before every **@test** annotated method.

**@AfterMethod:** A method which is marked with this annotation will be executed after every **@test** annotated method.

**@BeforeClass:** A method which is marked with this annotation will be executed before ***first @Test*** method execution. It runs only once per class.

**@AfterClass:**A method which is marked with this annotation will be executed after all the test methods in the current class have been run

**@BeforeTest:** A method which is marked with this annotation will be executed before ***first @Test*** annotated method.

**@AfterTest:**A method which is marked with this annotation will be executed when **all @Test** annotated methods complete the execution of those classes which are inside <test> tag in testng.xml file.

**@BeforeSuite:** A method which is marked with this annotation will run **only once before** all tests in the suite have run

**@AfterSuite:** A method which is marked with this annotation will run **once after** execution of all tests in the suite have run

**@BeforeGroups:**This annotated method will run **before the first test run** of that specific group.

**@AfterGroups:** This annotated method will run **after all test methods** of that group completes its execution.

Some other TestNG Annotations, we need to discuss here are mentioned below:

**@Parameters:**This annotation is used to pass parameters to test methods.

**@DataProvider:**If we use @DataProvider annotation for any method that means you are using that method as a data supplier. The configuration of @DataProvider annotated method must be like it always return Object[][] which we can use in @Test annotated 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.

**@Factory:**Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].

**@Listeners:**This annotation is used with test class. It helps in writing logs and results.

.

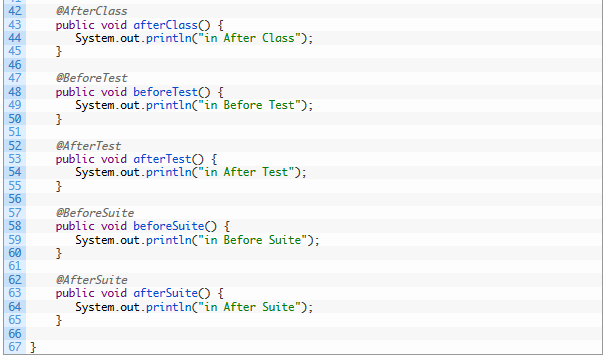
### Benefits of Using TestNG Annotations:

Following are some of the benefits of using annotations:

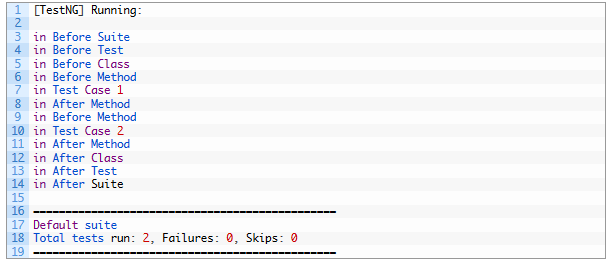
* TestNG identifies the methods it is interested in, by looking up annotations. Hence, method names are not restricted to any pattern or format.
* We can pass additional parameters to annotations.
* Annotations are strongly typed, so the compiler will flag any mistakes right away.
* Test classes no longer need to extend anything (such as TestCase, for JUnit 3).

**Let’s see the order of methods called using the below script:**

****

****

**Console output:**



**Execution process is as follows:**

First of all, beforeSuite() method is executed only once.

Lastly, the afterSuite() method executes only once.

Even the methods beforeTest(), beforeClass(), afterClass(), and afterTest() methods are executed only once.

beforeMethod() method executes for each test case but before executing the test case.

afterMethod() method executes for each test case but after executing the test case.

In between beforeMethod() and afterMethod(), each test case executes.

# How To Create TestNG XML File And Execute TestNG.XML File

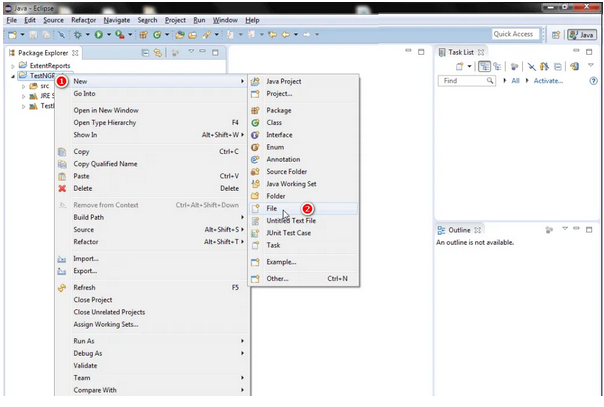
## ****Create TestNG XML file:****

In TestNG framework, we need to create T**estng xml** file to create and handle multiple test classes. We do configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc in the xml file.

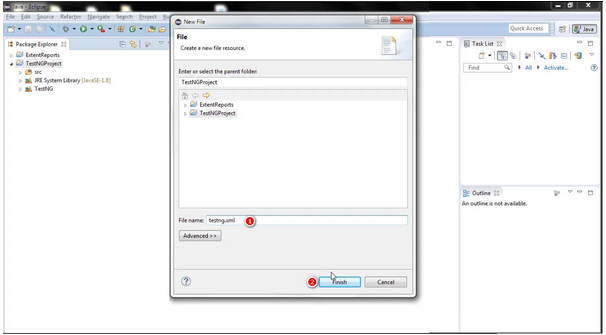
**Steps to create TestNG XML file**

**Step 1: Create testng xml file**

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



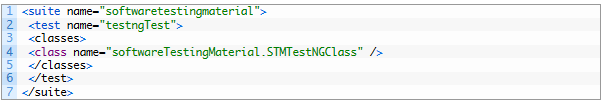
ii. In New file wizard, add file name as ‘**testng xml**‘ as shown in below given image and click on **Finish** button.



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

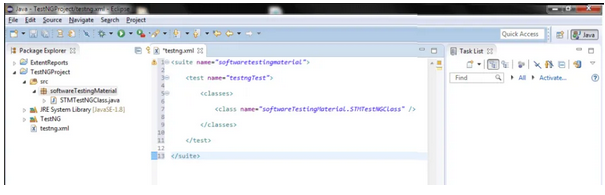
**Step 2 : Write xml code:**

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



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

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



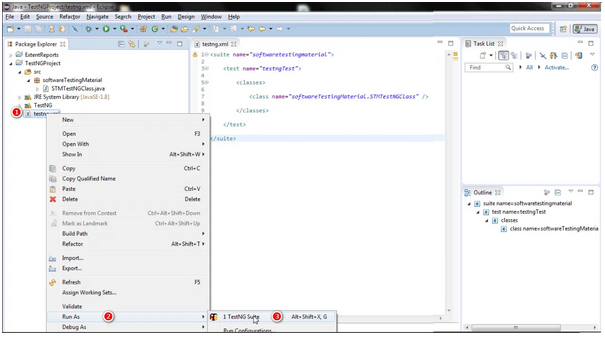
The hierarchy in the testng xml file is very simple to understand.

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.

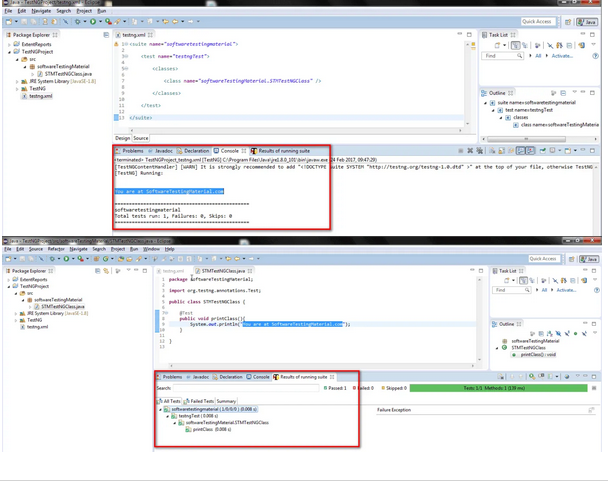
E.g. **Package Name** is “softwareTestingMaterial”, **Test Case Name** is “STMTestNGClass”. So the C**lass Name** should be softwareTestingMaterial.STMTestNGClass

**Step 3: Execute a testng xml**

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



Once the execution is done, you could view test result under the TestNg console.



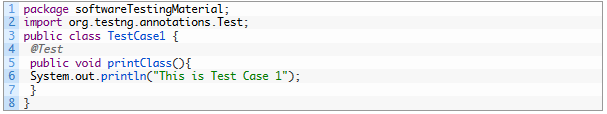
**How To Ignore TestNG Test | TestNG Tutorial**

At times we may face some situations where our code is not ready and the test cases written to test a particular method may fail. In such cases, we could use TestNG annotation *@Test (enabled = false).* This TestNG annotation allows us to ignore testng test.

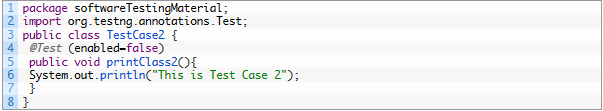
If a test method is annotated with *@Test (enabled = false)*, then the test case that is not ready to test is ignored.

Now, let’s see how to ignore TestNG test using a script.

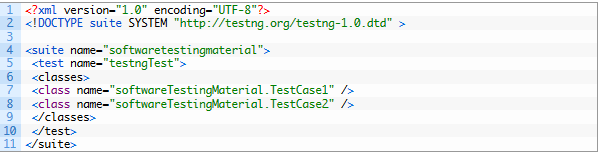
**Script – Test Case 1:**



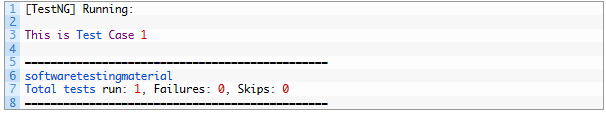
**Script – Test Case 2:** *I would like to ignore this test from the test execution*



**testNg.xml file:**



**Console Output:**



# Skip TestNG Test | TestNG Tutorial

## How To Skip TestNG Test:

Let’s see how to skip TestNG test deliberately. Sometimes we may face a situation where our test cases might not be ready and we need to skip those tests from running. One way of skipping a test method is by using throw new SkipException() exception.

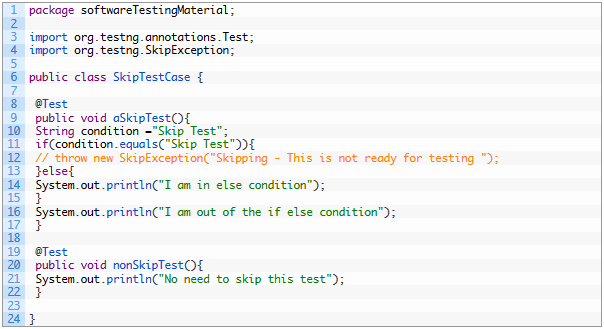
**Scenario:** Skip TestNG Test, If a condition met else continue execution.

Let see a sample WebDriver test case example where I have placed SkipException() inside if condition to Intentionally skip that test.

**Note:** Once SkipException() thrown, remaining part of that test method will not be executed and control will goes directly to next test method execution.

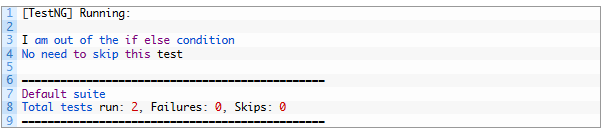
**Scenario 1:**

Let me show you how the below program works by commenting the “throw new SkipException()” in the if condition.



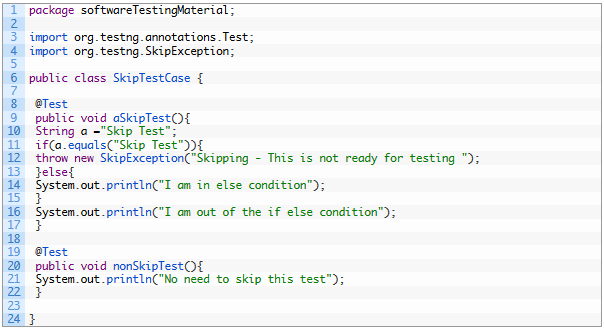
In the output console, we could see **“out of loop”** from first test method “aSkipTest” and No need to skip this test from second test method **“nonSkipTest”**

**Output Console:**

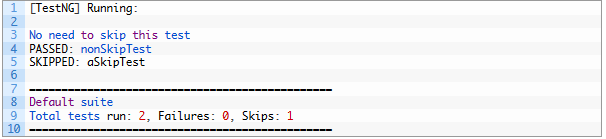


**Scenario 1:**

I will uncomment the “throw new SkipException()” in the if condition.



**Output Console:**



Skip exception thrown and the remaining part of the first test method **“aSkipTest”** not executed and control reached to second test method **“nonSkipTest”** and printed the value as **“No need to skip this test”**

# TestNG Groups | TestNG Tutorial

## TestNG Groups:

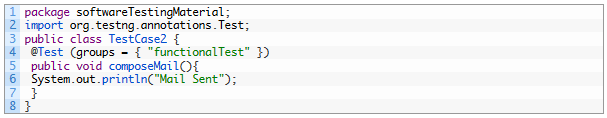
TestNG allows you to perform sophisticated groupings of test methods. Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups (or regular expressions) while excluding another set.  This gives you maximum flexibility in how you partition your tests and doesn’t require you to recompile anything if you want to run two different sets of tests back to back.

Groups are specified in your testng.xml file and can be found either under the <test> or <suite> tag. Groups specified in the <suite> tag apply to all the <test> tags underneath.

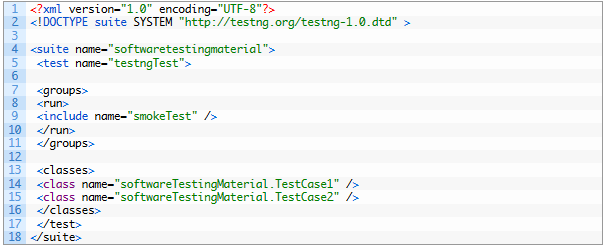
**Script – Test Case 1:**



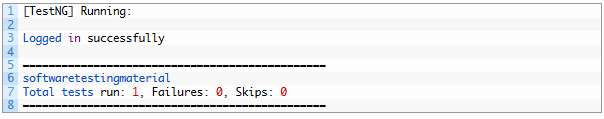
**Script – Test Case 2:**



**testng.xml:**



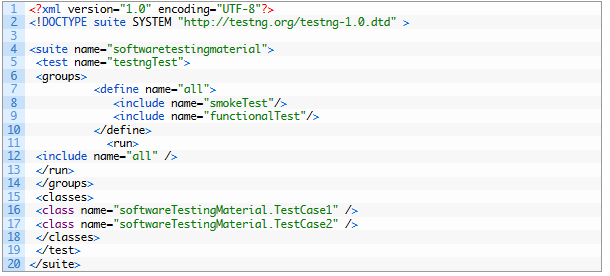
**Console Output:**



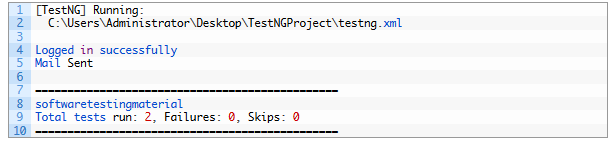
## ****Group of Groups in TestNG Groups:****

Groups can also include other groups. These groups are called *MetaGroups*. For example, you might want to define a group *all* that includes *smokeTest*and functionalTest. Let’s modify our testng.xml file as follows:

**testng.xml – Group of Groups:**



**Console Output:**



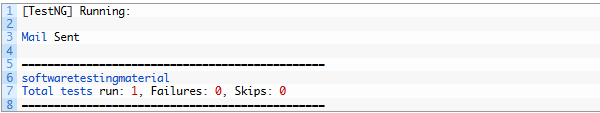
## ****Groups Exclusion:****

TestNG allows you to include groups as well as exclude them. You can ignore a group by using the <exclude> tag as shown below:

For example, it is quite usual to have tests that temporarily break because of a recent change, and you don’t have time to fix the breakage yet. However, you do want to have clean runs of your functional tests, so you need to deactivate these tests but keep in mind they will need to be reactivated.



**Console Output:**



|  |
| --- |
|  |

You can also disable tests on an individual basis by using the “enabled” property available on both @Test and @Before/After annotations.

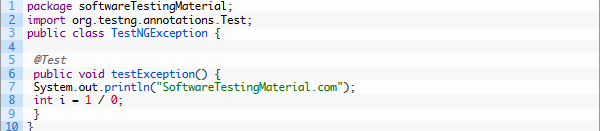
# TestNG Exception | TestNG Tutorial

## TestNG Exception:

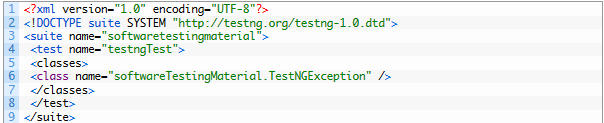
In this post, I will show how to use TestNG Exception (i.e., **expectedExceptions)** parameter along with the @Test annotation. TestNG provides an option of tracing the exception handling of code.

Firstly, we see a basic program without using TestNG Exception.

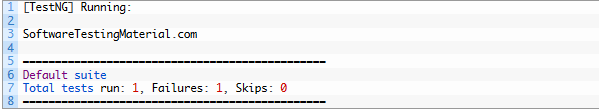
Created a java class, say, TestNGException.java and added an error condition in the method testException().



**testng.xml:**



**Console Output:**



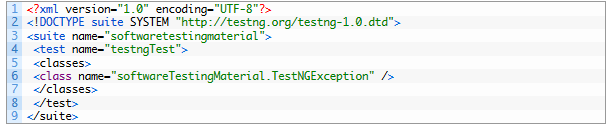
As you can see from the test results, testException() method was marked as failed by TestNG during execution.

Now let’s see TestNG Exception in action.

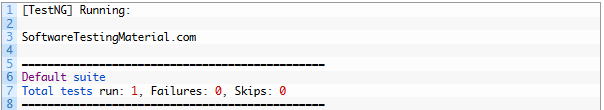
The expected exception to validate while running the below test is mentioned using the **expectedExceptions** attribute value while using the @Test annotation.



**testng.xml:**



**Console Output:**



As you can see from the test results, testException() method was marked as passed by TestNG during execution.

# TestNG Dependencies

## TestNG Dependencies

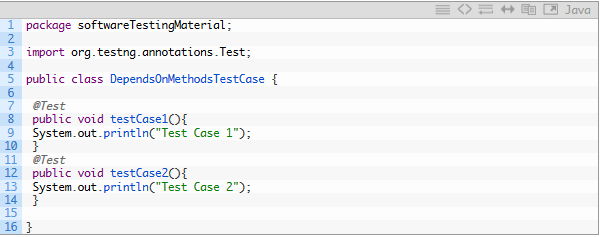
Sometimes, you may need to invoke methods in a test case in a certain order. Here comes TestNG Dependencies into the picture.

TestNG allows you to specify dependencies either with annotations or in XML.

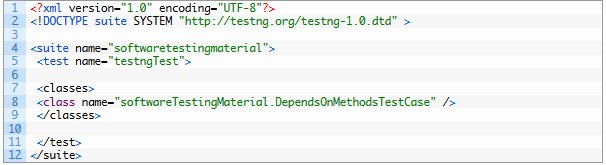
**First, we see Dependencies with annotations:**

TestNG allows you to specify dependencies either with:

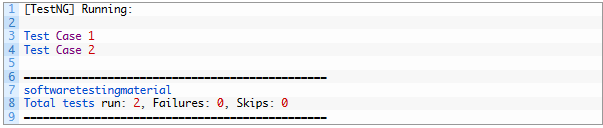
* Using attribute *dependsOnMethods* in @Test annotations, OR.
* Using attribute *dependsOnGroups* in @Test annotations.



**testng.xml**

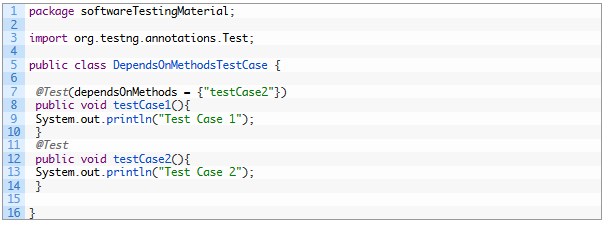


**Console Output:**



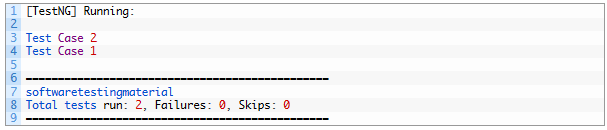
Now we add the dependsOnMethods attribute to the @Test Annotations and execute the same program.

**See the script below.**

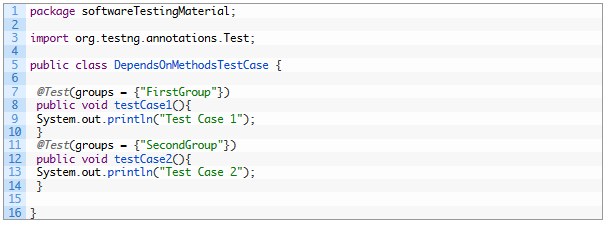


Execute the same testng.xml which was placed above and see the difference in Console Output

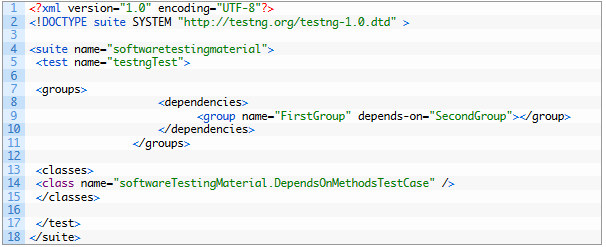
**Console Output:**



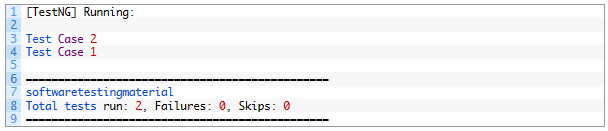
**Let’s see Dependencies with XML:**



**testng.xml**



**Console Output:**



**TestNG Parameterization Using XML**

**TestNG Parameterization Using XML:**

*Parameterized tests* allow developers to run the same test over and over again using different values.

There are two ways to set these parameters:

* *with testng.xml*
* *with Data Providers*

Let’s see passing parameters with testng.xml:

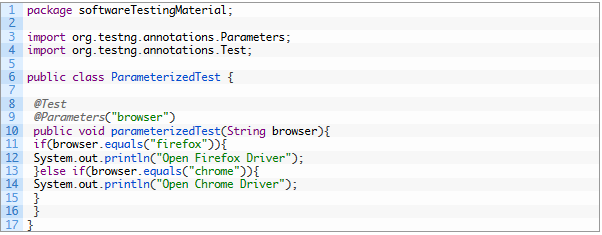
With this technique, we could define the parameters in the *testng.xml* file and then reference those parameters in the source files.

Create a java test class, say, *ParameterizedTest.java*

Add test method *parameterizedTest()* to your test class. This method takes a string as input parameter

Add the annotation @Parameters(“browser”) to this method. The parameter would be passed a value from testng.xml, which we will see in the next step.

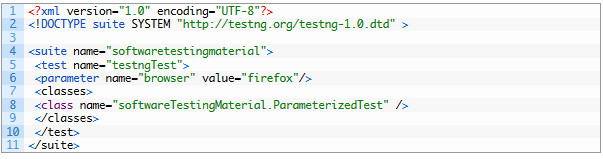
Java



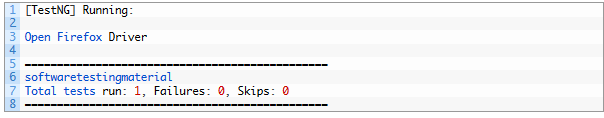
**testng.xml16.PNG**

Here, name attribute represents the parameter name and value represents the value of that parameter. We could use this parameter to the test method *parameterizedTest* by using @Parameters(“browser”) as mentioned in the above step.

Now let’s see the complete *testng.xml* file



**Console Output:**



TestNG will automatically try to convert the value specified in testng.xml to the type of your parameter. Here are the types supported:

* String
* int/Integer
* boolean/Boolean
* byte/Byte
* char/Character
* double/Double
* float/Float
* long/Long
* short/Short

# TestNG Parameterization Using Data Providers

## ****TestNG Parameterization Using Data Providers****

*Parameterized tests* allow developers to run the same test over and over again using different values.

There are two ways to set these parameters:

* with *testng.xml*
* with Data Providers

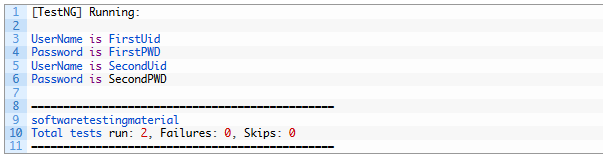
Let’s see passing parameters using DataProviders:

Specifying parameters in testng.xml might not be sufficient if you need to pass complex parameters, or parameters that need to be created from Java (complex objects, objects read from a property file or a database, etc…). In this case, you can use a Data Provider to supply the values you need to test.  A Data Provider is a method on your class that returns an array of objects.  This method is annotated with @DataProvider:

Java



**Console Output:**



# Parallel Test Execution In TestNG [Parallel Execution & MultiThreading]

## Parallel Test Execution In TestNG

There are situations where we want to run multiple tests with same or different browsers at the same time. In such cases, we can use “parallel” attribute in testng.xml to accomplish parallel test execution in TestNG

The parallel attribute of suite tag can accept four values:

tests – All the test cases inside <test> tag of testng.xml file will run parallel

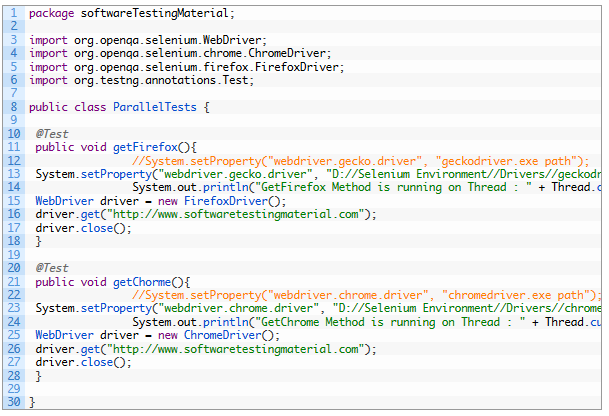
classes – All the test cases inside a java class will run parallel

methods – All the methods with @Test annotation will execute parallel

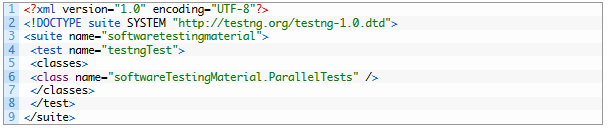
instances – Test cases in same instance will execute parallel but two methods of two different instances will run in different thread.

let us look at example of Parallel Test execution in TestNG.

In the below program, I took two methods. First methods opens Firefox driver and navigate to http://www.softwaretestingmaterial.com and closes the browser. Second methods opens Chrome driver and navigate to the same URL and closes the browser.

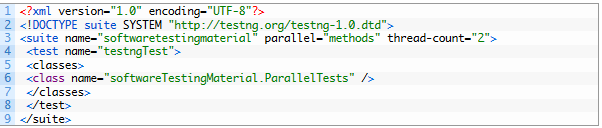


testng.xml file without mentioning **parallel** attribute:



After running the testng.xml using the above mentioned code, first you could see the firefox browser in action and then you can see the chrome driver in action.

To run both the browsers in parallel, use the below code in your testng.xml file.



Once you run the testng.xml using the above code, you could see both the browsers in action at a time.

Here in the above testng.xml file, I have passed parallel=methods and thread-count=2 at the suite level. I would like to execute selenium scripts in parallel in different threads. Most of the times, these two methods will execute in different threads. Thread Id may vary on every run. Here we are just passing thread count but we are not assigning any thread id, assigning thread id will be taken care by your system processor.

# Testng Asserts – How To Use Assertions In TestNG

## TestNG Asserts

TestNG Asserts are the most frequently used methods while creating Selenium Scripts. Let’s see Assertions in TestNG and where to use them

TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Let’s see a basic example using TestNG Asserts.

We do verify the title of the webpage using TestNG Asserts.

Here I do take two test conditions. In the first condition, I take a title value correctly and use assertEquals statement and in the second condition, I take incorrect title value to deliberately throw the exception.

**Step 1:** Open Gmail

**Step 2:** Verify whether the title matches to the given String. If it matches, go to the email field and type the given text in the sendKeys method else it throws an exception

**Note:** In this step, I took the ***actualTitle*** value correctly in the below script. So it goes to the email field and type the given text.

**Step 3:** Again open gmail

**Step 4:** Verify whether the title matches to the given String. If it matches, go to the email field and type the given text in the sendKeys method else it throws an exception

**Note:** In this step, I took incorrect ***actualTitle*** value. Here due to assertion fails, it throws an exception.



### Different TestNG Asserts Statements:

***Assert.assertEquals(String actual,String expected) :*** Asserts that two Strings are equal. If they are not, an AssertionError is thrown.

**Parameters:**  
actual – the actual value  
expected – the expected value

***Assert.assertEquals(String actual,String expected, String message)* :** Asserts that two Strings are equal. If they are not, an AssertionError, with the given message, is thrown.

**Parameters:**  
actual – the actual value  
expected – the expected value  
message – the assertion error message

***Assert.*assertEquals(boolean actual,boolean expected) :** Asserts that two booleans are equal. If they are not, an AssertionError is thrown.

**Parameters:**  
actual – the actual value  
expected – the expected value

***Assert.assertTrue(condition)* :** Asserts that a condition is true. If it isn’t, an AssertionError is thrown.

**Parameters:**  
condition – the condition to evaluate

***Assert.assertTrue(condition, message) :****Asserts that a condition is true. If it isn’t, an AssertionError, with the given message, is thrown.*

**Parameters:**  
condition – the condition to evaluate  
message – the assertion error message

***Assert.assertFalse(condition) :*** Asserts that a condition is false. If it isn’t, an AssertionError is thrown.

**Parameters:**  
condition – the condition to evaluate

***Assert.assertFalse(condition, message) :*** Asserts that a condition is false. If it isn’t, an AssertionError, with the given message, is thrown.

**Parameters:**  
condition – the condition to evaluate  
message – the assertion error message

**How To Use Soft Assert In TestNG**

Before knowing what is Soft Assert, first let’s see what is an Assert and what is the disadvantage in using Assert and why we are moving to Soft Assert.

Asserts are used to perform validations in the test scripts.

There are two types of Assert:

1. Hard Assert
2. Soft Assert

When an assert fails the test script stops execution unless handled in some form. We call general assert as Hard Assert

**Hard Assert** – Hard Assert throws an *AssertException* immediately when an assert statement fails and test suite continues with next *@Test*

The disadvantage of Hard Assert – It marks method as fail if assert condition gets failed and the remaining statements inside the method will be aborted.

To overcome this we need to use Soft Assert. Let’s see what is Soft Assert.

**Soft Assert** – Soft Assert collects errors during *@Test*. Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

If there is any exception and you want to throw it then you need to use *assertAll()* method as a last statement in the @Test and test suite again continue with next *@Test* as it is.

We need to create an object to use Soft Assert which is not needed in Hard Assert.

Let’s see a practical example:

Here I took two methods namely *softAssert()* and *hardAssert()*.

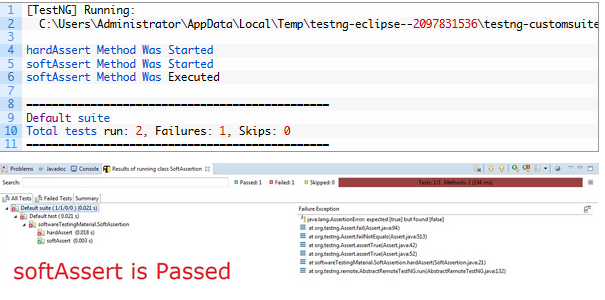
In the *softAssert()* method, I have used SoftAssert class and intentionally passing value false in the assertTrue() method to make it fail

In the*hardAssert()*method, I simply used Assert and intentionally passing parameter value false in the assertTrue() method to make it fail



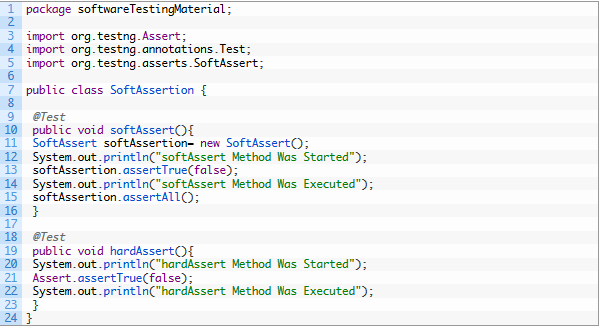
Execute the above script and see the console output. You could see only one failure.

**Console Output:**



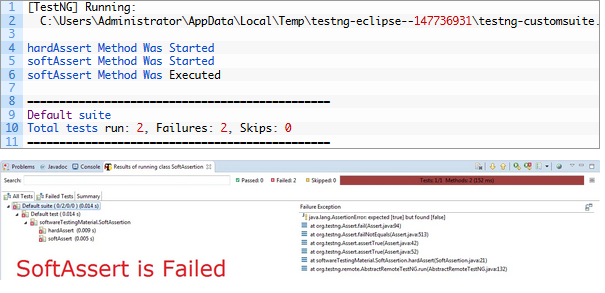
Below script is same as the first one but just added *assertAll()* method in the end of the first method (i.e., *softAssert()*).

Note:  If you forget to call *assertAll()* at the end of your test, the test **will pass** even if any assert objects threw exceptions as shown in the above example. So don’t forget to add *assertAll()*



Execute the above script and see the console output. There are two failures here. Second failure is due to *assertAll()* method

Console output:



**TestNG Listeners – Selenium WebDriver**

In this post, we see TestNG listeners. Listeners “listen” to the event defined in the selenium script and behave accordingly. The main purpose of using listeners is to create logs. There are many types of listeners such as WebDriver Listeners and TestNG Listeners.

Here in this post, we see TestNG Listeners. Using TestNG listeners we could generate logs and customize TestNG Reports.

Let’s see how to implement TestNG Listeners.

**Step 1:** Create a Class “***ListenerTestNG***” to implement ***ITestListener*** methods

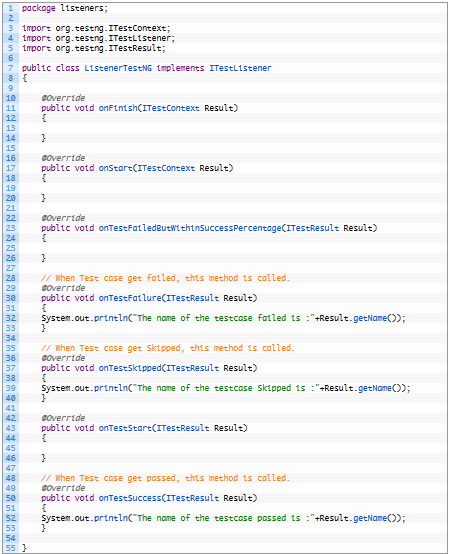




Mouse hover on *ITestListener and import ITestListener* Interface andmouse hover on *ListenerTestNG*and click on “*Add unimplemented methods*” to add multiple unimplemented methods (without body)



Here, I am implementing *onTestSuccess*, *onTestSkipped*, *onTestFailure*methods.

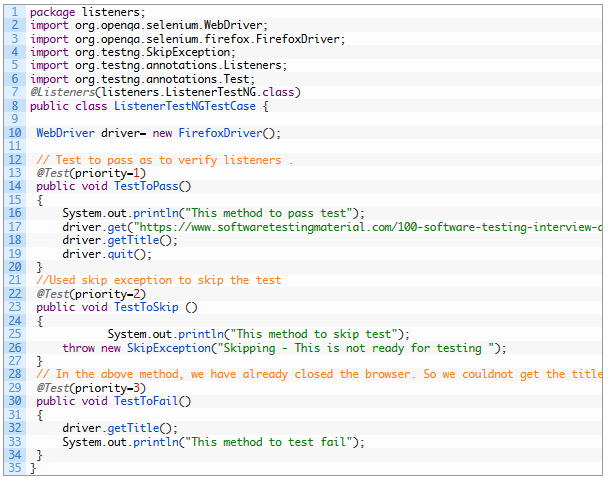


**Step 2:** Create another Class “*ListenerTestNGTestCase*” and write a script (which ever script you prefer). Else copy paste the below mentioned script.

**Step 3:** Add the listeners annotation (*@Listeners*) in the Class “*ListenerTestNGTestCase*”

5.PNG

The complete “*ListenerTestNGTestCase*” class after adding Listener annotation is mentioned below:



**Step 4:** Execute the “*ListenerTestNGTestCase*” class. Methods in class “ListenerTestNG” are called automatically according to the behavior of methods annotated as @Test.

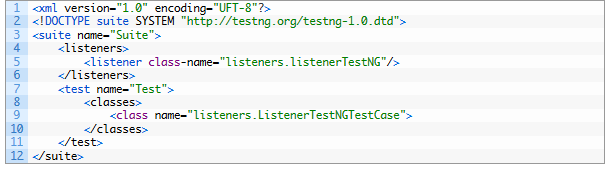
**Step 5**: Verify the Output in the console. You could find the logs in the console.

**If you want to use listeners in multiple classes.**

Add the below lines of code in the TestNG.xml file

9.PNG

Final testng.xml file will be like this:



Execute it by right clicking on testng.xml and run as TestNG Suite

# How To Run Failed Test Cases Using TestNG In Selenium WebDriver

## Run Failed Test Cases Using TestNG in Selenium WebDriver:

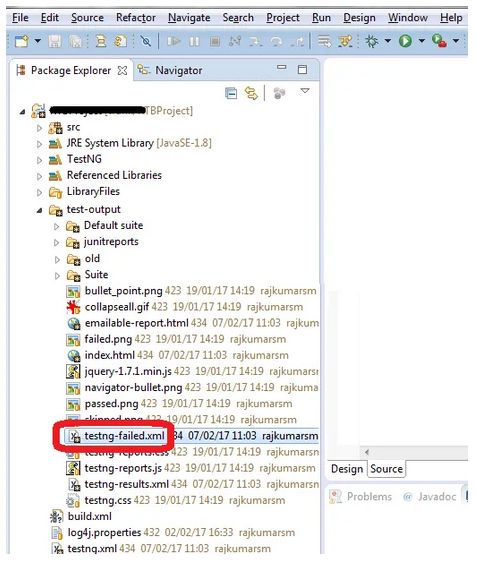
At times, test cases may fail while running automated test scripts.  The reason may be anything (say, Network issue, System issue or browser issue) but as an automation tester, you need to execute the test scripts again. Here is a solution to run failed test cases using TestNG in Selenium.

We could execute the failed test cases in two ways.

**Case 1: Execute failed test cases using TestNG in Selenium – By using “testng-failed.xml”**

**Steps To follow:**

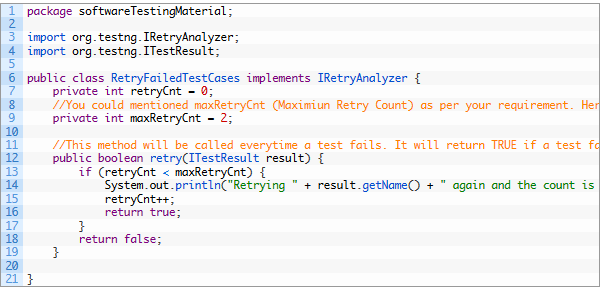
1. After the first run of an automated test run. Right click on Project – Click on Refresh
2. A folder will be generated named “test-output” folder. Inside “test-output” folder, you could find “testng-failed.xml”
3. Run “testng-failed.xml” to execute the failed test cases again.



**Case 2: Execute failed test cases using TestNG in Selenium – By Implementing TestNG IRetryAnalyzer.**

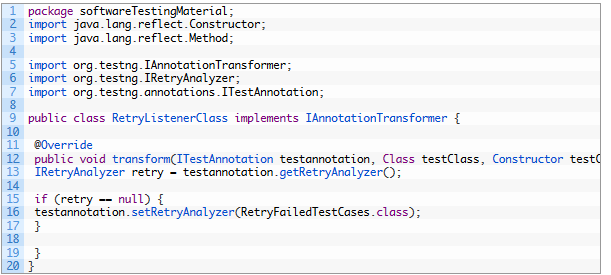
Create a class to implement IRetryAnalyzer. Here I am creating a class (say, RetryFailedTestCases) and implementing IRetryAnalyzer.

**RetryFailedTestCases implements IRetryAnalyzer:**



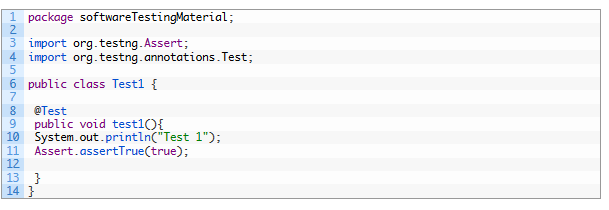
Let’s create another class ‘RetryListenerClass’ by Implementing ‘IAnnotationTransformer’ interface. transform method is called for every test during test run. A simple implementation of this ‘IAnnotationTransformer’ interface can help us set the ‘setRetryAnalyzer’ for ‘ITestAnnotation’. Add the above class name (RetryFailedTestCases.class) in the below program. This interface does its work in run time by adding annotation to the test methods.

**RetryListenerClass implements IAnnotationTransformer:**

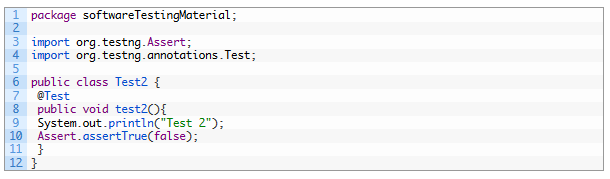


Let us see the example by executing simple tests below. Here I took two test cases say Test1 and Test2.

**Testcase 1:**



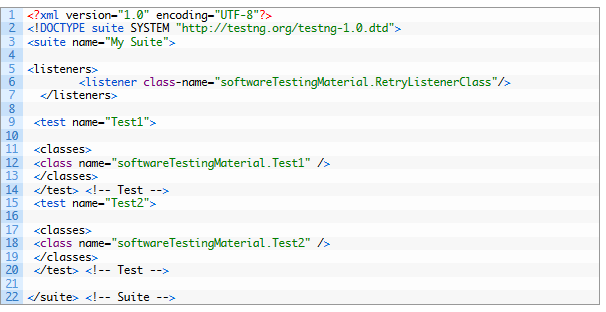
**Testcase 2:**



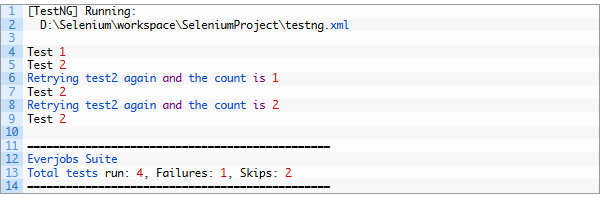
As per the lines of code in Test2, it will fail. So it (Test2) will be executed 2 times (we took the maxRetryCnt as 2) in Retry Class. First lets include below mentioned Listener to testng.xml file. Below mentioned syntax is to add Listener for RetryListnereClass

16.PNG

Final testng.xml file should looks like below:



Execute the testng.xml. Here is the output which I got. You could see in the below mentioned result that the Test 2 is executed three times as we have mentioned ‘maxRetryCnt = 2’. Even though we have just 2 tests, we could find total test runs are 4 in the result.



This way we could run failed test cases using TestNG in Selenium.

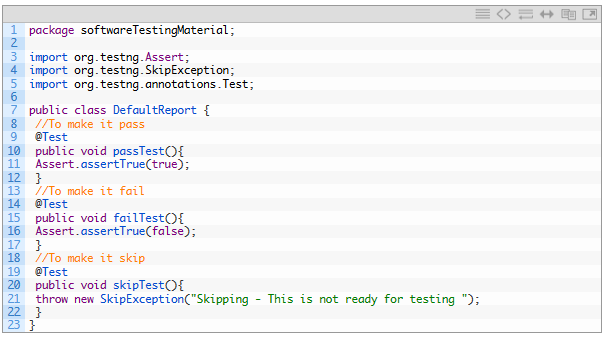
# How To Generate TestNG Reports

## TestNG Reports:

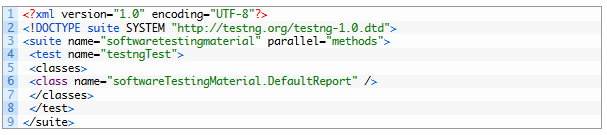
TestNG Reports come in to the picture once we execute the test cases using TestNG. Once we execute test cases using TestNG, it will generate a default HTML report. Let’s see this process of generating TestNG Reports in detail.

I have created a basic script and mentioned below.

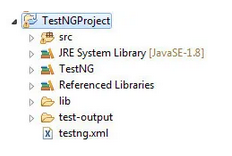
It has three methods namely passTest, failTest and skipTest with @Test annotation.



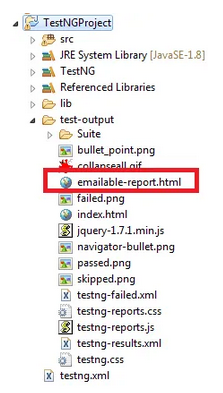
**TestNG.xml file**



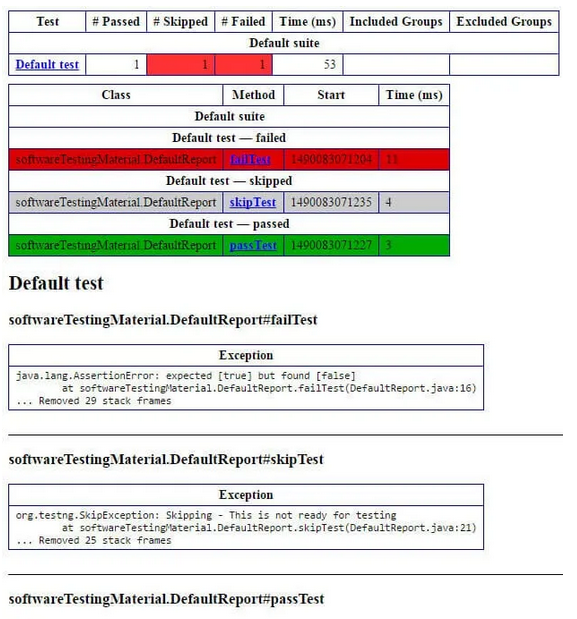
Execute the testng.xml file and refresh the project. You could see your project similar to the below mentioned image.



Navigate to ‘test-output’ folder. Now you should find a report ”**emailable-report.html**‘. This is the default report generated by TestNG.



Open **emailable -report.html**using any browser of your choice. Report will be like as shown below:



# How To Run TestNG Using Command Prompt

## ****Goal:**** To Run TestNG using Command Prompt:

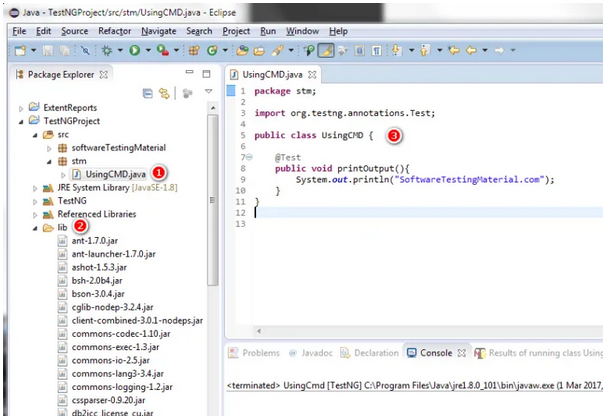
Steps to achieve our goal – Executing TestNG using Command Prompt:

1. Open Eclipse and create a Java class
2. Write a Java program
3. Convert the Java Program into TestNG
4. Open command prompt
5. Run the TestNG using command prompt

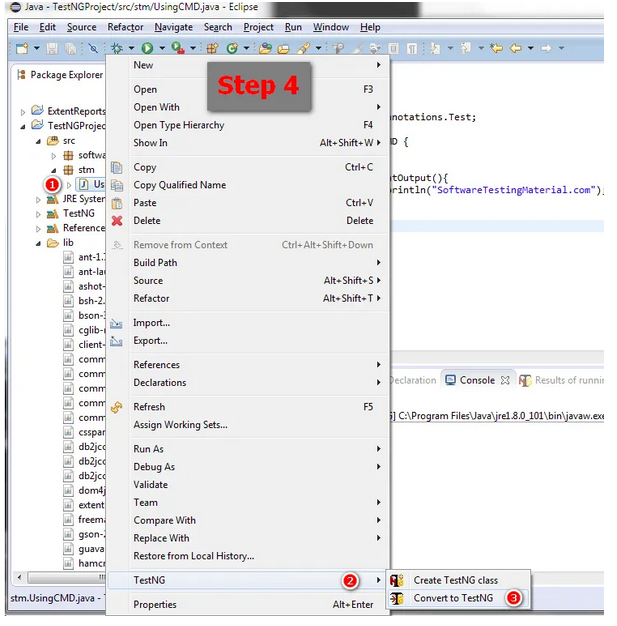
**Step i:** Open Eclipse and create a Java class

**Step ii.** Keep all the library files in a folder (here I create a folder name “lib”)

**Step iii:** Write a Java program

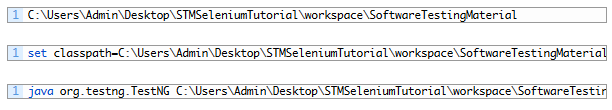


**Step iv:** Convert the Java Program into TestNG



**Step v:** Open command prompt

**Step vi:** Run the TestNG using command prompt



**Also you could run TestNG using Batch file (.bat file)**

Copy the below code and place it in a notepad and save the file using .bat extension

