TestNG Annotations

In this post we will learn common TestNG annotations including @Test, @BeforeMethod, @AfterMethod, @BeforeClass, @AfterClass, @BeforeGroups, @AfterGroups, @BeforeSuite, @AfterSuite, @BeforeTest & @AfterTest. Further posts next in tutorial explains in detail other popular annotations including [@Parameters](http://websystique.com/java/testing/testng-parameters-example/) & [@DataProvider](http://websystique.com/java/testing/testng-dataprovider-example/) annotations.

TestNG Annotations can be categorized based on there usage level(basic,advanced) and how frequently they are used.

**1) Basic Annotations:** Annotations which are most frequently used on day to day basis.

* @Test, @BeforeClass, @AfterClass, @BeforeMethod, @AfterMethod

**2) Advanced Annotations:** Annotations which are used on special situations/setup/configurations.

* Group Annotations : @BeforeGroups, @AfterGroups
* Suite related Annotations : @BeforeSuite, @AfterSuite, @BeforeTest, @AfterTest

Each annotation also comes with several attributes which can be used to alter the test execution behavior.

Let’s go through annotations in details:

Basic Annotations

First thing first, let’s create a sample class whose methods will be unit-tested using testNG.

Below is the sample class we will use in this example.

|  |
| --- |
| package com.websystique.testng;    public class Calculator {        public double add(double a, double b){          return a+b;      }        public double subtract(double a, double b){          return a-b;      }        public double multiply(double a, double b){          return a\*b;      }    } |

Nothing special about this Calculator class, which provides methods for common arithmetic operations. We will unit test each of these methods.

|  |
| --- |
| package com.websystique.testng;    import java.lang.reflect.Method;    import org.testng.Assert;  import org.testng.annotations.AfterClass;  import org.testng.annotations.AfterMethod;  import org.testng.annotations.BeforeClass;  import org.testng.annotations.BeforeMethod;  import org.testng.annotations.Test;    public class TestCalculator {        Calculator calculator;        @BeforeClass      public void beforeClass(){          //Ideal place to perform some setup which is shared among all tests.          //E.g. Initializing DB connection, setting environment properties          System.out.println("@BeforeClass: I run only once, before first test start.");          calculator = new Calculator();      }        @AfterClass      public void afterClass(){          //Ideal place to perform some cleanup of setup which is shared among all tests.          System.out.println("@AfterClass: I run only once, after all tests have been done.\n");          calculator = null;      }        @BeforeMethod      public void beforeEachTestMethod(Method method){//Parameter are optional          //May perform some initialization/setup before each test.          //E.g. Initializing User whose properties may be altered by actual @Test          System.out.println("\n@BeforeMethod: I run before each test method. Test to be executed is : "+method.getName());      }        @AfterMethod      public void afterEachTestMethod(Method method){//Parameter are optional          //May perform cleanup of initialization/setup after each test.          System.out.println("@AfterMethod: I run after each test method. Test just executed is : "+method.getName()+"\n");      }        @Test      public void testAdd(){          System.out.println("@Test add");          Assert.assertEquals(calculator.add(2, 3), 5.0);      }        @Test      public void testSubtract(){          System.out.println("@Test subtract");          Assert.assertTrue(calculator.subtract(5, 3) > 1, "Subtract test failed");      }        @Test      public void testMultiply(){          System.out.println("@Test multiply");          Assert.assertEquals(calculator.multiply(5, 3) , 15.0);      }    } |

Run above tests class using **mvn clean test** or TestNG eclipse plugin as mentioned in [introduction post](http://websystique.com/java/testing/testng-hello-world-example/)

Following is the output of above test run:

|  |
| --- |
| @BeforeClass: I run only once, before first test start.    @BeforeMethod: I run before each test method. Test to be executed is : testAdd  @Test add  @AfterMethod: I run after each test method. Test just executed is : testAdd      @BeforeMethod: I run before each test method. Test to be executed is : testMultiply  @Test multiply  @AfterMethod: I run after each test method. Test just executed is : testMultiply      @BeforeMethod: I run before each test method. Test to be executed is : testSubtract  @Test subtract  @AfterMethod: I run after each test method. Test just executed is : testSubtract    @AfterClass: I run only once, after all tests have been done.    PASSED: testAdd  PASSED: testMultiply  PASSED: testSubtract    ===============================================      Default test      Tests run: 3, Failures: 0, Skips: 0  ===============================================      ===============================================  Default suite  Total tests run: 3, Failures: 0, Skips: 0  =============================================== |

Let’s go through each used annotation in detail and try to understand each line in above output:

**@BeforeClass, @AfterClass**

Method annotated with @BeforeClass will be executed only once, before the first test method in this class starts to run. It’s a good place to do any **initialization or configuration setup** which is **common to all test methods** and eventually may be used in those test methods. Note how we created an instance of calculator in @BeforeClass method which is used throughout all tests method. Method annotated with @AfterClass will be executed only once but only after all tests case have been executed. It provides a handle to do some cleanup after all tests in this class have been executed. You can see in above output that @BeforeClass and @AfterClass were executed only once and that too at very beginning and at very end.

**@BeforeMethod, @AfterMethod**

Method annotated with @BeforeMethod will be executed before each test method. Similarly @AfterMethod method will be executed after each test method execution. @BeforeMethod annotated method is a good place to initialize some setup which can be used and eventually altered by the test method. Similarly @AfterMethod annotated method is a handy place to cleanup the setup created in @BeforeMethod and updated by @Test method itself.

Optionally, you can provide some specific arguments in @BeforeMethod and @AfterMethod methods in order to get some info related to the test in execution. In our example, we have declared an argument Method which eventually helps us to find the name of the test in execution.

**@Test**

Methods annotated with @Test are actual test methods, serving as unit test in above example. In @Test methods, we call real methods and assert the output/behavior from actual method for success/failure.

Advanced Annotations

**1) Group Annotations:**

TestNG allows us to group several tests together. You can group certain tests based on what behavior/aspect they are actually testing. You may have a scenario where few tests belong to a certain group(say database) and other ones belong to other group(say security) and yet another one belong to other group(say UI). With this approach you may decide to execute only certain group of test and skip other ones(let’s say there was a regression on database related code, so only execute database related tests). Several Tests can belong to a group and a test can be part of several groups.

**@BeforeGroups, @AfterGroups**

Method annotated with @BeforeGroups gets executed only once for a group before any of the test of that group executes. Method annotated with @AfterGroups gets executed only once for a group only after all of the tests of that group finished execution.

[TestNG Groups Example](http://websystique.com/java/testing/testng-groups-example/)  
Post containing full example code for testNG grouping related aspects.

**2) Suite Annotations:**

Every Test you run using TestNG runs as part of a suite. A Suite in testNG is represented by one XML file, usually named as **testng.xml**. **<suite>** tag is the first tag in XML file which represents a suite and it may contains one or more **<test>** tags. Each <test> tag in turn may contain one or more **<classes>**, **<packages>**, **<groups>** tags and more.

You can provide a Suite level configuration/setup and/or a Test [**<test>** tag] level configuration setup using following annotations:

**@BeforeSuite, @AfterSuite**

Method annotated with @BeforeSuite is executed only once, before any of the tests method execution in a given suite. It is an ideal place to do some setup/initialization for environments which will be shared among all tests of this suite. Method annotated with @AfterSuite gets executed only once for a suite only after all of the tests of that suite finished execution.

**@BeforeTest, @AfterTest**

Method annotated with @BeforeTest is executed only once, before any of the tests method execution in a given **<test>** tags which may contain multiple classes and packages. It is an ideal place to do some setup/initialization for environments which will be shared among all tests defined in this **<test>** tag. Method annotated with @AfterTest gets executed only once for a given **<test>** tag, and only after all of the tests of that **<test>** tag finished execution.