***TEST NG: Testing Next Generation***

***< A Test Execution Engine for Java > <TEST RUNNER>***

TestNG is designed to cover all categories of tests:

* **unit,**
* **functional,**
* **end-to-end,**
* **integration.**

and it ***requires JDK 5 or higher***.

- TestNG is an **open source** Test Automation framework where **NG** stands for **Next Generation**

- TestNG inspired from JUnit but introduced some new functionality

- TestNG has **multiple classes and methods**

- for **Robust** Framework.

***TestNG Features:***

1. Annotations
2. Object Oriented features
3. Integrated classes.
4. Compile time Test code is separated from run time configuration/data.
5. Flexible run time configuration.
6. **test groups**.
7. Supports **Dependent test methods**, **parallel testing**, **load testing**, and **partial failure.**

***Benefits of TestNG:***

......................

i) It generates **HTML test reports**

ii) **TestNG Annotations** are easy to create test cases

iii) **Test Cases can be grouped and prioritized** more easily

iv) **Parallel testing** is possible

v) **Parametrization** test is possible.

- Run Tests in arbitrarily big thread pools

- Easy and Flexible TEST configuration.

- **data driven testing** with **@DataProvider annotation.**

***Installation:***

1. **As a Java Library (.jar)**
2. **As a Maven dependency.**

***TestNG annotations:***

***@Test***

- considers **method under** it as a **Testcase.**

- used with a Java void method and the method ***can be default as well as parameterised method.***

- one @Test with one java method only and each one of them is considered as a Test Method.

- for N no of Test methods, we need to use N no of @Test annotations.

***@BeforeMethod***

- applies to **all the methods in the class**, it **executes before each Test.**

- *runs before every Test Method within the same class.*

- write once run N no of times where N = No. @Test in the same class

- *need at least one @Test method to run, if no @Test in a class then*

***@BeforeMethod*** will not be executed.

***@AfterMethod***

- applies to all the methods in the class, it executes **after each Test**

-- *runs after every Test Method within the same class.*

- write once run N no of times where N = No. @Test in the same class

- need at least one @Test method to run, if no @Test in a class then ***@AfterMethod*** will not be executed.

***@BeforeClass***

- it will be called only one time for each class ie.. while loading the class

- applies to the respective class only not to any other class.

- *it executes before all the methods/Tests in the class.*

***@AfterClass***

- it will be called only one time for each class ie.. while exiting/closing the class

- applies to the respective class only not to any other class.

- it executes after all the Class/methods/Tests.

**Note:**

*@BeforeClass and @AfterClass need at least one @Test as similar to that of @BeforeMethod and @AfterMethod to execute the method under it or else it will be skipped.*

***@BeforeTest***

- *present in one class applies to all the classes/package in a project*

- Called before Class/methods/Test

- usually used to *initialize Selenium WebDriver in functional Test Automation.*

***@AfterTest***

- present in one class applies to all the classes/package in a project

- Called after Class/methods/Test.

***Note:***

*we**can have multiple @BeforeTest and @AfterTest in a project.*

1. ***The*** *order of preference will be entirely based on the name of the class* ***i.e. class name having alphabetically lesser alphabets starting from a to z will have higher order of preference of execution for the @BeforeTest and @AfterTest annotations irrespective of alphabets of the Test methods following these annotations.***
2. ***Multiple @BeforeTest and @AfterTest annotations are possible in a single class, In that scenario order of preference is based on the alphabetical order of the name of the methods followed by the annotations.***
3. ***@BeforeTest and @AfterTest annotations does not require any @Test in the same class but need in some other class for the execution.***

**@BeforeSuite**

-The annotated method will be run before all tests in this suite have run.

- **root annotation** / **starting point of Test Execution of a project**

**- Superior**

- executed before all other annotations

***@AfterSuite***

The annotated method will be run after all tests in this suite have run.

- end of the Suite /last point

- executed after all the annotations are executed.

- after Test NG has generated Test Report/ Output

- usually ***to flush the test report and cleanse the DB.***

**@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.

**@DataProvider**

Marks a method as **supplying data for a test method**.

The annotated method must return an Object[ ][ ], where each Object[ ] can be assigned the parameter list of the test method.

The @Test method that wants to receive data from this DataProvider needs to use a dataProvider attribute which is assigned with the name of the @DataProvider, and if the name attribute is not assigned for @DataProvider then name of the method followed by @DataProvider will be used as name attribute for @Test.

**@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**

Defines listeners on a test class.

**@Parameters**

Describes how to pass parameters to a @Test method.

***Hierarchy of annotations:***

@BeforeSuite

@BeforTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

***Attributes used with annotations***

***@Test***

1. **description**

to add description to the Test

Ex: @Test(description = "any statement")

1. **priority**

prioritisation of Test: parameter - "priority" - a Number

*priority can be negative number also*, but industry standard starts from1, lowest Number executes first which means it has highest priority.

Ex: @Test(priority = 1)

1. **enabled**

is a boolean: true /false

it’s a parameter to disable(false) test cases.

Ex: @Test(enabled=false)

1. **groups**

to group the test cases for respective groups.

Ex: @Test(groups={“SMOKE”,”REGRESSION”})

1. **invocationCount**

It should be a positive integer.

If the value is 0 or any negative integer Test Runs will be 0.

*Same Test will be run N no of times provided N is a positive integer.*

1. **invocationTimeOut**

The maximum number of milliseconds that the total number of invocations on this test method should take.

*This annotation will be ignored if the attribute invocationCount is not specified on this method.*

If it hasn't returned after this time, it will be marked as a FAIL.

So ideally speaking your invocationTimeout value should be calculated as :

*Time taken per invocation \* Number of invocations.*

**Note:**

The invocationTimeOut should be at least 1 or above (i.e. range 1 to +N)

If it is 0 or negative integer it will be ignored.

This attribute will not be ignored if invocationCount is not specified.

1. **timeOut**

The maximum number of milliseconds the test(each test in case of invocationCount) should take.

Timeout can be configured at two levels:

**Suite level** – This will be applicable for all the tests in the said TestNG test suite.

**Ex:**

<suite name="Time test Suite" time-out="500" verbose="1" >

  <test name="Timeout Test" >

    <classes>

      <class name="com.howtodoinjava.test.TimeoutSuite" />

    </classes>

  </test>

</suite>

**Test level** – This will be applicable for the said test method and will override the timeout period if configured at the suite level.

**Ex:**

@Test(timeOut = 500)

1. **successPercentage**

The percentage of success expected from the method.

For example: @Test(successPercentage=60, invocationCount=5), in this annotation **success percentage** is 60% and invocation count is 5, that means out of 5 times if at least 3 times ((⅗)\*100= 60) the test method gets passed, it would be considered as passed.

1. **dependsOnMethods / dependsOnGroups**

Test case will be executed only when the method on which it is depending on will be executed with success.

Ex: @Test(dependsOnMethods = {"Method Name"})

similar to it we have dependsOnGroups - Test will be executed when Group on which it is depending is executed with success.

Ex: @Test(dependsOnGroups = {"Group 1","Group 2"})

***NOTE:*** dependsOnMethods / dependsOnGroups --> "**HARD DEPENDENCY**"

1. **alwaysRun** = true --> "***SOFT DEPENDENCY***"

Ex: @Test(alwaysRun = true)

1. **expectedExceptions**

The **list of exceptions that a test method is expected to throw.**

*If no exception or a different than one on this list is thrown, this test will be marked a failure.*

*Ex:*

@Test(expectedExceptions=ArithmeticException.**class**)

**Note:**

multiple exceptions for the same Test case can be set as below:

@Test(expectedExceptions={ArithmeticException.**class**,FileNotFoundException.**class**})

1. **singleThreaded**

If set to true, all the methods on this test class are guaranteed to run in the same thread, even if the tests are currently being run with parallel="methods".

This attribute can only be used at the class level and it will be ignored if used at the method level.

**Note:** this attribute used to be called sequential (now deprecated).

1. **threadPoolSize**

The size of the thread pool for the method.

The method will be invoked from multiple threads as specified by invocationCount.   
**Note:** this attribute is ignored if invocationCount is not specified

1. **dataProvider**

used to fetch data to a test case from a Data Provider method where data is of type Object[][].

Ex:

@Test(dataProvider=”Name of the Data Provider”)

**Note:**

If for @DataProvider , name attribute is not used than,

Name of the method followed by @DataProvider should be used at the place of Name of the Data Provider.