**TAGS**

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

Above annotation support below parameters:

alwaysRun : for before methods always run irrespective of group

dependsOnGroups, dependsOnMethods, enabled, groups,

inheritGroups: If true, this method will belong to groups specified in the @Test annotation at the class level.

Ex:

@test(groups={“g1”,”g2”})

Public class WebTest{….

onlyForGroups: Only for @BeforeMethod and @AfterMethod

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**@DataProvider: returns Object[][]**

**name:** provider name**, parallel:** tests generated using this data provider are run in parallel if true

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**@Factory: returns Object[]**

**@Listeners: Defines listeners on a test class.**

**value:** An array of classes that extend org.testng.ITestNGListener.

**@Parameters: Describes how to pass parameters to a @Test method.**

Value: list of variables used to fill the parameters of this method.

**@Test**

**alwaysRun**, **dataProvider**, dataProviderClass, **dependsOnGroups**, dependsOnMethods, description, **enabled,** expectedExceptions, **groups, invocationCount,** invocationTimeOut, **priority,** successPercentage, **singleThreaded**, timeOut, **threadPoolSize**

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

|  |
| --- |
|  |

|  |
| --- |
| <**suite** name="Suite1" verbose="1" data-provider-thread-count="20"> |

|  |
| --- |
| <**test** name="Nopackage" preserve-order="false" > //random execute order |
| |  | | --- | | <**groups**> |  |  | | --- | | <**run**> |  |  | | --- | | <**exclude** name="brokenTests"  /> |  |  | | --- | | <**include** name="checkin.\*"  /> //regex in include, exclude, group names |  |  | | --- | | </**run**> |  |  | | --- | | </**groups**> | |

|  |
| --- |
| <**classes**> |

|  |
| --- |
| <**class** name="NoPackageTest" /> |
| |  | | --- | | <**methods**> |  |  | | --- | | <**include** name="testMethod" />  <**exclude** name=".\*page" /> |  |  | | --- | | </**methods**>  </**class**> | |

|  |
| --- |
| </**classes**> |
| //specify packages instead of class name |
| |  | | --- | | <**packages**> |  |  | | --- | | <**package** name="test.sample" /> |  |  | | --- | | </**packages**> | |  | |

|  |
| --- |
| </**test**> |

|  |
| --- |
| </**suite**> |

RUNNING testNG file:

java org.testng.TestNG testng1.xml [testng2.xml testng3.xml ...]

Return values from test methods:

|  |
| --- |
| <**suite** allow-return-values="true"> |

|  |
| --- |
|  |

|  |
| --- |
| or |

|  |
| --- |
|  |

|  |
| --- |
| <**test** allow-return-values="true"> |

Test-Groups

public class Test1 {

@Test(groups = { "functest", "checkintest" })

public void testMethod1() {}}

groups of groups “**MetaGroups**”:

<groups>

<define name="functest">

<include name="windows"/>

<include name="linux"/>

</define>

<define name="all">

<include name="functest"/>

<include name="checkintest"/>

</define>

<run>

<include name="all"/>

</run>

</groups>

**Define parameters:**

<suite name="My suite">

<parameter name="first-name" value="Cedric"/>

<test name="Simple example">

<-- ... -->

@Parameters({ "first-name" })

@Test

public void testSingleString(String firstName) {

System.out.println("Invoked testString " + firstName);

assert "Cedric".equals(firstName);

}

@Parameters("db")

@Test

public void testNonExistentParameter(@Optional("mysql") String db) { ... } // if db is not found set mysql value

**DataProvider**

@DataProvider(name = "test1", parallel = true)

public Object[][] createData1() {

return new Object[][] {

{ "Cedric", new Integer(36) },

{ "Anne", new Integer(37)},

};

}

@Test(dataProvider = "test1") or @Test(dataProvider = " test1", dataProviderClass = Provider.class)

public void verifyData1(String n1, Integer n2) {

System.out.println(n1 + " " + n2);

}

@DataProvider(name = "dp")

public Object[][] createData(**Method m**) {

System.out.println(m.getName()); **// print test method name**

return new Object[][] { new Object[] { "Cedric" }};

}

@Test(dataProvider = "dp")

public void test1(String s) {

}

**DEPENDS on USING XML**

<groups>

<dependencies>

<group name="c" depends-on="a b" />

<group name="z" depends-on="c" />

</dependencies>

</groups>

**FACTORY CLASS**

<**class** name="WebTestFactory" />

public class WebTestFactory {

@Factory

public Object[] createInstances() {

Object[] result = new Object[10];

for (int i = 0; i < 10; i++) {

result[i] = new WebTest(i \* 10);

}

return result;

}

}

public class WebTest {

private int m\_numberOfTimes;

public WebTest(int numberOfTimes) {

m\_numberOfTimes = numberOfTimes;

}

@Test

public void testServer() {

for (int i = 0; i < m\_numberOfTimes; i++) {

// access the web page

}

}

}

**OR RUNNING TestNG suites programatically**

TestNG testNG = new TestNG();

testNG.setTestClasses(WebTestFactory.class);

testNG.run();

**factory can receive privider**

@Factory(dataProvider = "dp")

public FactoryDataProviderSampleTest(int n) {

super(n);

}

@DataProvider

static public Object[][] dp() {

return new Object[][] {

new Object[] { 41 },

new Object[] { 42 },

};

}

@Ignore – equal to enabled false

<**suite** name="My suite" parallel="methods|tests|classes|instances" thread-count="5">

@Test(threadPoolSize = 3, invocationCount = 10, timeOut = 10000)

public void testServer() {

**Rerunning failed tests**

creates a file called testng-failed.xml

**java -classpath testng.jar;%CLASSPATH% org.testng.TestNG -d test-outputs testng.xml**

**java -classpath testng.jar;%CLASSPATH% org.testng.TestNG -d test-outputs test-outputs\testng-failed.xml**

**import org.testng.IRetryAnalyzer;**

**import org.testng.ITestResult;**

**retry on runtime**

public class MyRetry implements IRetryAnalyzer {

private int retryCount = 0;

private static final int maxRetryCount = 3;

@Override

public boolean retry(ITestResult result) {

if (retryCount < maxRetryCount) {

retryCount++;

return true;

}

return false;

}

}

import org.testng.Assert;

import org.testng.annotations.Test;

public class TestclassSample {

@Test(retryAnalyzer = MyRetry.class)

public void test2() {

Assert.fail();

}

}

There are several interfaces that allow you to modify TestNG's behavior. These interfaces are broadly called "TestNG Listeners". Here are a few listeners:

* IAnnotationTransformer ([doc](https://testng.org/doc/documentation-main.html#annotationtransformers), [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/IAnnotationTransformer.html))
* IAnnotationTransformer2 ([doc](https://testng.org/doc/documentation-main.html#annotationtransformers), [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/IAnnotationTransformer2.html))
* IHookable ([doc](https://testng.org/doc/documentation-main.html#ihookable), [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/IHookable.html))
* IInvokedMethodListener (doc, [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/IInvokedMethodListener.html))
* IMethodInterceptor ([doc](https://testng.org/doc/documentation-main.html#methodinterceptors), [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/IMethodInterceptor.html))
* IReporter ([doc](https://testng.org/doc/documentation-main.html#logging-reporters), [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/IReporter.html))
* ISuiteListener (doc, [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/ISuiteListener.html))
* ITestListener ([doc](https://testng.org/doc/documentation-main.html#logging-listeners), [javadoc](https://jitpack.io/com/github/cbeust/testng/master/javadoc/org/testng/ITestListener.html))

-Dtestng.mode.dryrun=true dry run