TestnG - Dependencies

We will learn about the dependency feature in TestNG. Dependency allows users of TestNG to specify a dependency **method** or a dependency **group** for a test method. This will help in executing a set of tests to be executed before a test method. Method dependency only works if the depend-on-method is part of the same class or any of the inherited base class (that is, while extending a class).

* Dependency test
* Writing a multiple dependency test
* Dependency on group
* Using regular expressions
* Defining dependency through XML

Single Method Dependency

Below test class contains two test methods which print a message name onto the console when executed. Here, test method testOne depends on test method testTwo. This is configured by using the attribute dependsOnMethods while using the Test annotation as shown is the preceding code.

**package** pack3;

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

**public** **class** SimpleDependencyTest {

@Test(dependsOnMethods = { "testTwo" })

**public** **void** testOne() {

System.***out***.println("Test method one");

}

@Test

**public** **void** testTwo() {

System.***out***.println("Test method two");

}

}

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Test method two

Test method one

PASSED: testTwo

PASSED: testOne

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

Default test

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

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

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

Default suite

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

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

MultiDependency

Below test class contains three test methods which print a message name onto the console when executed. Here test method testOne depends on test methods testTwo and testThree. This is configured by using the attribute dependsOnMethods while using the Test annotation as shown in the preceding code.

**package** pack3;

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

**public** **class** MultiDependencyTest {

@Test(dependsOnMethods = { "testTwo", "testThree" })

**public** **void** testOne() {

System.***out***.println("Test method one");

}

@Test

**public** **void** testTwo() {

System.***out***.println("Test method two");

}

@Test

**public** **void** testThree() {

System.***out***.println("Test method three");

}

}

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Test method three

Test method two

Test method one

PASSED: testThree

PASSED: testTwo

PASSED: testOne

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

Default test

Tests run: 3, Failures: 0, Skips: 0

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

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

Default suite

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

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

Inherited Tests

Below test class contains two test methods which print a message name onto the console when executed. Here test method testThree depends on test method testOne. This is configured by using the attribute dependsOnMethods while using the Test annotation

**package** pack3;

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

**public** **class** InheritedTest **extends** SimpleDependencyTest {

@Test(dependsOnMethods = { "testOne" })

**public** **void** testThree() {

System.***out***.println("Test three method in Inherited test");

}

@Test

**public** **void** testFour() {

System.***out***.println("Test four method in Inherited test");

}

}

**package** pack3;

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

**public** **class** SimpleDependencyTest {

@Test(dependsOnMethods = { "testTwo" })

**public** **void** testOne() {

System.***out***.println("Test method one");

}

@Test

**public** **void** testTwo() {

System.***out***.println("Test method two");

}

}

testng.xml

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.InheritedTest"*></class>

</classes>

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

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

See below screenshot and console output



[RemoteTestNG] detected TestNG version 6.14.2

Test four method in Inherited test

Test method two

Test method one

Test three method in Inherited test

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

Suite

Total tests run: 4, Failures: 0, Skips: 0

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

Group Dependency

The dependency on the group is configured using the attribute dependsOnGroups while using

the Test annotation, as shown in the preceding code.

Method dependency only works with other methods that belong to the same class or in one of the inherited classes but not across different classes. In case you need a test method that exists in a separate class; you can achieve this by assigning the said test method to a group and configuring the dependent test method to be dependent on that group.

Example of dependsOnGroups defined in same class:

**package** pack3;

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

**public** **class** GroupDependency {

@Test(dependsOnGroups = { "grouptest" })

**public** **void** groupTestOne() {

System.***out***.println("Group Test method one");

}

@Test(groups = { "grouptest" })

**public** **void** groupTestTwo() {

System.***out***.println("Group test method two");

}

@Test(groups = { "grouptest" })

**public** **void** groupTestThree() {

System.***out***.println("Group Test method three");

}

}

testng.xml

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.GroupDependency"*></class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Group Test method three

Group test method two

Group Test method one

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

Suite

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

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



Example of dependsOnGroups defined in another class:

**package** pack3;

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

**public** **class** GroupDependency {

@Test()

**public** **void** groupTestOne() {

System.***out***.println("Group Test method one");

}

@Test(groups = { "grouptest" })

**public** **void** groupTestTwo() {

System.***out***.println("Group test method two");

}

@Test

**public** **void** groupTestThree() {

System.***out***.println("Group Test method three");

}

}

**public** **class** MultiDependencyTest {

@Test(dependsOnGroups= {"grouptest"})

**public** **void** testOne() {

System.***out***.println("Test method one");

}

@Test

**public** **void** testTwo() {

System.***out***.println("Test method two");

}

@Test

**public** **void** testThree() {

System.***out***.println("Test method three");

}

}

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

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

<suite name=*"Suite1"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.MultiDependencyTest"*></class>

</classes>

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

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

Run above testng.xml and below error is thrown

[RemoteTestNG] detected TestNG version 6.14.2

org.testng.TestNGException:

DependencyMap::Method "MultiDependencyTest.testOne()[pri:0, instance:pack3.MultiDependencyTest@3b0090a4]" depends on nonexistent group "grouptest"

at org.testng.DependencyMap.getMethodsThatBelongTo(DependencyMap.java:41)

at org.testng.internal.DynamicGraphHelper.createDynamicGraph(DynamicGraphHelper.java:76)

at org.testng.TestRunner.privateRun(TestRunner.java:618)

at org.testng.TestRunner.run(TestRunner.java:505)

at org.testng.SuiteRunner.runTest(SuiteRunner.java:455)

at org.testng.SuiteRunner.runSequentially(SuiteRunner.java:450)

at org.testng.SuiteRunner.privateRun(SuiteRunner.java:415)

at org.testng.SuiteRunner.run(SuiteRunner.java:364)

at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)

at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:84)

at org.testng.TestNG.runSuitesSequentially(TestNG.java:1208)

at org.testng.TestNG.runSuitesLocally(TestNG.java:1137)

at org.testng.TestNG.runSuites(TestNG.java:1049)

at org.testng.TestNG.run(TestNG.java:1017)

at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:114)

at org.testng.remote.RemoteTestNG.initAndRun(RemoteTestNG.java:251)

at org.testng.remote.RemoteTestNG.main(RemoteTestNG.java:77)

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

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

<suite name=*"Suite1"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.MultiDependencyTest"*></class>

<class name=*"pack3.GroupDependency"*></class>

</classes>

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

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

Run above testng.xml and below is the output

[RemoteTestNG] detected TestNG version 6.14.2

Test method three

Test method two

Group Test method one

Group Test method three

Group test method two

Test method one

PASSED: testThree

PASSED: testTwo

PASSED: groupTestOne

PASSED: groupTestThree

PASSED: groupTestTwo

PASSED: testOne

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

Default test

Tests run: 6, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 6, Failures: 0, Skips: 0

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



Using Regular Expressions

**package** pack3;

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

**public** **class** RegularExpressionTest {

@Test(dependsOnGroups = { "starts-with.\*" })

**public** **void** regularExpMethod() {

System.***out***.println("Dependent method");

}

@Test(groups = { "starts-with-one" })

**public** **void** startsWithMethodOne() {

System.***out***.println("Starts with method one");

}

@Test(groups = { "starts-with-two" })

**public** **void** startsWithMethodTwo() {

System.***out***.println("Starts with method two");

}

}

testng.xml

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

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

<suite name=*"Suite1"* time-out=*"500"*>

<test name=*"Test"*>

<groups>

<run>

<include name=*"starts-with.\*"*>

</include>

</run>

</groups>

<classes>

<class name=*"pack3.RegularExpressionTest"*></class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Starts with method one

Starts with method two

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

Suite1

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

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



Defining dependency through XML

In the below SimpleXmlDependency, test 'groupTestOne' depends on group 'testgroup' i.e on methods groupTestTwo() and groupTestThree()

**package** pack3;

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

**public** **class** SimpleXmlDependency {

@Test(dependsOnGroups= {"testgroup"})

**public** **void** groupTestOne() {

System.***out***.println("Group Test method one");

}

@Test(groups = { "testgroup" })

**public** **void** groupTestTwo() {

System.***out***.println("Group test method two");

}

@Test(groups = { "testgroup" })

**public** **void** groupTestThree() {

System.***out***.println("Group Test method three");

}

}

**Below is testng.xml to run '**SimpleXmlDependency'

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.SimpleXmlDependency"*>

</class>

</classes>

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

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

**Output:**

[RemoteTestNG] detected TestNG version 6.14.2

Group Test method three

Group test method two

Group Test method one

PASSED: groupTestThree

PASSED: groupTestTwo

PASSED: groupTestOne

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

Default test

Tests run: 3, Failures: 0, Skips: 0

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

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

Default suite

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

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

See below modified SimpleXmlDependency. There is no mention of 'dependsOnGroups' for method 'groupTestOne'

**package** pack3;

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

**public** **class** SimpleXmlDependency {

@Test(groups = { "dependentgroup" })

**public** **void** groupTestOne() {

System.***out***.println("Group Test method one");

}

@Test(groups = { "testgroup" })

**public** **void** groupTestTwo() {

System.***out***.println("Group test method two");

}

@Test(groups = { "testgroup" })

**public** **void** groupTestThree() {

System.***out***.println("Group Test method three");

}

}

Below testng.xml defined dependencies

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<groups>

<dependencies>

<group depends-on=*"testgroup"* name=*"dependentgroup"*></group>

</dependencies>

</groups>

<classes>

<class name=*"pack3.SimpleXmlDependency"*>

</class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Group Test method three

Group test method two

Group Test method one

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

Suite

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

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

The Factory Annotation

* What is factory
* Passing parameters to test classes
* The DataProvider annotation with the @Factory annotation
* The DataProvider or @Factory annotation
* Dependency with the @Factory annotation

Sometimes we may need to run a set of tests with different data values. To achieve this we may define a separate set of tests inside a suite in the testng XML and test the required scenario. The problem with this approach is that, if you get an extra set of data, you will need to redefine the test. TestNG solves this problem by providing the @Factory annotation feature.

**Factory in TestNG defines and creates tests dynamically at runtime.**

**It’s mandatory that a factory method should return an array of Object class i.e. Object [].**

**package** pack3;

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

**public** **class** SimpleTest {

@Test

**public** **void** testmethod1() {

System.***out***.println("Inside testMethod1");

}

}

**package** pack3;

**import** org.testng.annotations.Factory;

**import** pack3.SimpleTest;

**public** **class** SimpleTestFactory {

@Factory

**public** Object[] factoryMethod() {

Object s1 = **new** SimpleTest();

Object s2 = **new** SimpleTest();

Object ob1[] = **new** Object[2];

ob1[0] = s1;

ob1[1] = s2;

**return** ob1;

}

}

The preceding class defines a factory method inside it. A factory method is defined by declaring @Factory above the respective test method.

Run the factory by using below testng.xml

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.SimpleTestFactory"*>

</class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Inside testMethod1

Inside testMethod1

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

Suite

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

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

The test method from the SimpleTestFactory class was executed two times. The execution is based on the Object array returned by the factory method. As the said method returns two objects of the SimpleTest class, TestNG looks inside the specified returned object and executes all the test methods inside it. In this case, as there was only one test method, TestNG executes the respective test method.

**One of the main advantage of using the factory methods is that you can pass parameters to test classes while initializing them.**

See below

**package** pack3;

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

**public** **class** SimpleTest {

**int** param;

**public** SimpleTest(**int** param) {

**this**.param = param;

}

@Test

**public** **void** testmethod1() {

System.***out***.println("Inside testMethod1 and param + 1 = " + (param + 1));

}

@Test

**public** **void** testmethod2() {

System.***out***.println("Inside testMethod2 and param + 2 = " + (param + 2));

}

}

**package** pack3;

**import** org.testng.annotations.Factory;

**import** pack3.SimpleTest;

**public** **class** SimpleTestFactory {

@Factory

**public** Object[] factoryMethod() {

Object s1 = **new** SimpleTest(5);

Object s2 = **new** SimpleTest(10);

Object ob1[] = **new** Object[2];

ob1[0] = s1;

ob1[1] = s2;

**return** ob1;

}

}

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.SimpleTestFactory"*>

</class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Inside testMethod1 and param + 1 = 6

Inside testMethod1 and param + 1 = 11

Inside testMethod2 and param + 2 = 7

Inside testMethod2 and param + 2 = 12

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

Suite

Total tests run: 4, Failures: 0, Skips: 0

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



Another example of @Factory annotation

**package** pack3;

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

**public** **class** SimpleTest {

**int** param1;

**int** param2;

**public** SimpleTest(**int** param1, **int** param2) {

**this**.param1 = param1;

**this**.param2 = param2;

}

@Test

**public** **void** testmethod1() {

System.***out***.println("Inside testMethod1 and param1 = " + param1);

}

@Test

**public** **void** testmethod2() {

System.***out***.println("Inside testMethod2 and param2 = " + param2);

}

}

**package** pack3;

**import** org.testng.annotations.Factory;

**import** pack3.SimpleTest;

**public** **class** SimpleTestFactory {

@Factory()

**public** Object[] factoryMethod() {

Object s1 = **new** SimpleTest(5, 10);

Object s2 = **new** SimpleTest(15, 20);

Object ob1[] = **new** Object[2];

ob1[0] = s1;

ob1[1] = s2;

**return** ob1;

// or use below Array literal

// return new Object[] { new SimpleTest(5, 10), new SimpleTest(15, 20) };

}

}

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack3.SimpleTestFactory"*>

</class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Inside testMethod1 and param1 = 5

Inside testMethod1 and param1 = 15

Inside testMethod2 and param2 = 10

Inside testMethod2 and param2 = 20

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

Suite

Total tests run: 4, Failures: 0, Skips: 0

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



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

**package** pack4;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.DataProvider;

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

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

@BeforeClass

**public** **void** BeforeClassMethod() {

System.***out***.println("This is before class method");

}

@AfterClass

**public** **void** AfterClassMethod() {

System.***out***.println("This is after class method");

}

@BeforeTest

**public** **void** BeforeTestMethod() {

System.***out***.println("This is before test method");

}

@AfterTest

**public** **void** AfterTestMethod() {

System.***out***.println("This is after test method");

}

@DataProvider

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

**return** **new** Object[][] { { 0 }, { 1 }, { 2 } };

}

@Test(dataProvider = "dataMethod")

**public** **void** testMethodOne(**int** param) {

System.***out***.println("Test method one output: " + param);

}

@Test(dataProvider = "dataMethod")

**public** **void** testMethodTwo(**int** param) {

System.***out***.println("Test method two output: " + param);

}

@Test(dataProvider = "dataMethod")

**public** **void** testMethodThree(**int** param) {

System.***out***.println("Test method two output: " + param);

}

}

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack4.DataProviderTest"*>

</class>

</classes>

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

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

Output

[RemoteTestNG] detected TestNG version 6.14.2

This is before test method

This is before class method

Test method one output: 0

Test method one output: 1

Test method one output: 2

Test method two output: 0

Test method two output: 1

Test method two output: 2

Test method two output: 0

Test method two output: 1

Test method two output: 2

This is after class method

This is after test method

PASSED: testMethodOne(0)

PASSED: testMethodOne(1)

PASSED: testMethodOne(2)

PASSED: testMethodThree(0)

PASSED: testMethodThree(1)

PASSED: testMethodThree(2)

PASSED: testMethodTwo(0)

PASSED: testMethodTwo(1)

PASSED: testMethodTwo(2)

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

Default test

Tests run: 9, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 9, Failures: 0, Skips: 0

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



**package** pack4;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.DataProvider;

**import** org.testng.annotations.Factory;

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

**public** **class** FactoryDataProviderTest {

**private** **int** param;

@BeforeClass

**public** **void** BeforeClassMethod() {

System.***out***.println("This is before class method");

}

@AfterClass

**public** **void** AfterClassMethod() {

System.***out***.println("This is after class method");

}

@BeforeTest

**public** **void** BeforeTestMethod() {

System.***out***.println("This is before test method");

}

@AfterTest

**public** **void** AfterTestMethod() {

System.***out***.println("This is after test method");

}

@Factory(dataProvider = "dataMethod")

**public** FactoryDataProviderTest(**int** param) {

**this**.param = param;

}

@DataProvider

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

**return** **new** Object[][] { { 0 }, { 1 }, { 2 } };

}

@Test()

**public** **void** testMethodOne() {

System.***out***.println("Test method one output: " + param);

}

@Test

**public** **void** testMethodTwo() {

System.***out***.println("Test method two output: " + param);

}

@Test

**public** **void** testMethodThree() {

System.***out***.println("Test method three output: " + param);

}

}

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"*>

<classes>

<class name=*"pack4.FactoryDataProviderTest"*>

</class>

</classes>

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

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

Output

[RemoteTestNG] detected TestNG version 6.14.2

This is before test method

This is before class method

Test method one output: 1

This is before class method

Test method one output: 0

This is before class method

Test method one output: 2

Test method three output: 1

Test method three output: 0

Test method three output: 2

Test method two output: 1

This is after class method

Test method two output: 0

This is after class method

Test method two output: 2

This is after class method

This is after test method

PASSED: testMethodOne

PASSED: testMethodOne

PASSED: testMethodOne

PASSED: testMethodThree

PASSED: testMethodThree

PASSED: testMethodThree

PASSED: testMethodTwo

PASSED: testMethodTwo

PASSED: testMethodTwo

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

Default test

Tests run: 9, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 9, Failures: 0, Skips: 0

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



Many people get confused when they read about the DataProvider and @Factory annotations – what to use when? and what is better?

DataProvider: A test method that uses DataProvider will be executed a multiple number of times based on the data provided by the DataProvider. The test method will be executed using the same instance of the test class to which the test method belongs.

Factory: A factory will execute all the test methods present inside a test class using a separate instance of the respective class.

**running a dependency test sequentially**

**package** pack4;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.DataProvider;

**import** org.testng.annotations.Factory;

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

**public** **class** FactoryDataProviderTest {

**private** **int** param;

@BeforeClass

**public** **void** BeforeClassMethod() {

System.***out***.println("This is before class method");

}

@AfterClass

**public** **void** AfterClassMethod() {

System.***out***.println("This is after class method");

}

@BeforeTest

**public** **void** BeforeTestMethod() {

System.***out***.println("This is before test method");

}

@AfterTest

**public** **void** AfterTestMethod() {

System.***out***.println("This is after test method");

}

@Factory(dataProvider = "dataMethod")

**public** FactoryDataProviderTest(**int** param) {

**this**.param = param;

}

@DataProvider

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

**return** **new** Object[][] { { 0 }, { 1 }, { 2 } };

}

@Test(dependsOnMethods= {"testMethodTwo"})

**public** **void** testMethodOne() {

System.***out***.println("Test method one output: " + param);

}

@Test()

**public** **void** testMethodTwo() {

System.***out***.println("Test method two output: " + param);

}

@Test

**public** **void** testMethodThree() {

System.***out***.println("Test method three output: " + param);

}

}

**To run the dependent methods according to the sequence, they are supposed to run a configuration attribute group-by-instance, which is set to true.**

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

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

<suite name=*"Suite"* time-out=*"500"*>

<test name=*"Test"* group-by-instances=*"true"*>

<classes>

<class name=*"pack4.FactoryDataProviderTest"*>

</class>

</classes>

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

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

Output:

[RemoteTestNG] detected TestNG version 6.14.2

This is before test method

This is before class method

Test method three output: 1

Test method two output: 1

Test method one output: 1

This is after class method

This is before class method

Test method three output: 0

Test method two output: 0

Test method one output: 0

This is after class method

This is before class method

Test method three output: 2

Test method two output: 2

Test method one output: 2

This is after class method

This is after test method

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

Suite

Total tests run: 9, Failures: 0, Skips: 0

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

