**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

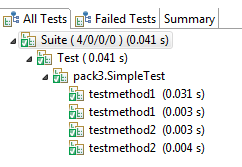
Inside testMethod2 and param + 2 = 12

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

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

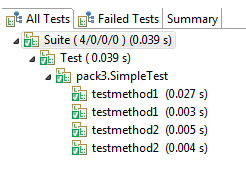
Inside testMethod2 and param2 = 20

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

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

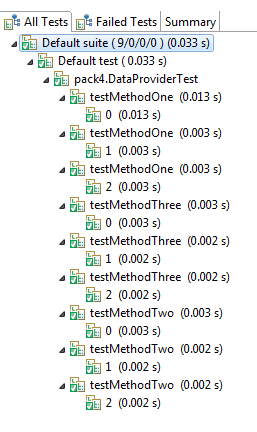
===============================================

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

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

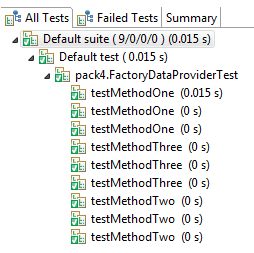
===============================================

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

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

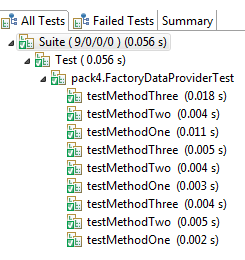
This is after test method

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

Suite

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

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



Example showing both @Factory and @DataProvider

Below is sample LoginTest class with 2 Test methods i.e. Login and call\_conference\_test

**package** TestNG;

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

**public** **class** LoginTest {

@Test()

**public** **void** Test\_Login() {

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

}

@Test()

**public** **void** call\_conference\_test() {

System.***out***.println("Inside call conference");

}

}

When above LoginTest is run below is the result:

[RemoteTestNG] detected TestNG version 6.14.2

Inside Logintest

Inside call conference

PASSED: Test\_Login

PASSED: call\_conference\_test

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

Default test

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

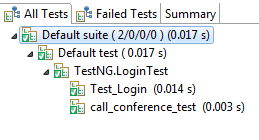
===============================================

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

Default suite

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

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



Below class has two test methods having it’s own DataProviders i.e. dp1 and dp2

**package** TestNG;

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

**public** **class** LogoutTest {

@Test(dataProvider = "dp1", dataProviderClass = DataProviderFactory.**class**)

**public** **void** Test\_Logout1(**int** id, String password) {

System.***out***.println("id = " + id);

System.***out***.println("Password = " + password);

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

}

@Test(dataProvider = "dp2", dataProviderClass = DataProviderFactory.**class**)

**public** **void** Test\_Logout2(**int** id, String password) {

System.***out***.println("id = " + id);

System.***out***.println("Password = " + password);

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

}

}

When above LogoutTest class is run below is the output:

[RemoteTestNG] detected TestNG version 6.14.2

id = 1

Password = Jigar

Inside Logintest

id = 2

Password = Mehta

Inside Logintest

id = 3

Password = Learning

Inside Logintest

id = 11

Password = TestNG

Inside Logintest

id = 12

Password = is

Inside Logintest

id = 13

Password = easy

Inside Logintest

PASSED: Test\_Logout1(1, "Jigar")

PASSED: Test\_Logout1(2, "Mehta")

PASSED: Test\_Logout1(3, "Learning")

PASSED: Test\_Logout2(11, "TestNG")

PASSED: Test\_Logout2(12, "is")

PASSED: Test\_Logout2(13, "easy")

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

Default test

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

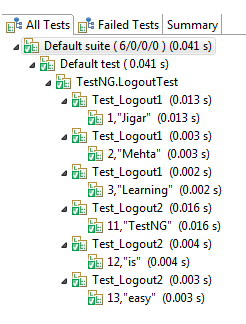
===============================================

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

Default suite

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

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



Below FactoryClass has @Factory method that creates and returns an object of LoginTest and LogoutTest. Hence when FactoryClass is run from TestNg.xml all methods in LoginTest and LogoutTest is run. Since Logout internally call dataproviders hence those methods are run that many times.

**package** TestNG;

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

**public** **class** FactoryClass {

@Factory()

**public** Object[] FactoryAnnotationTransformerExample1() {

System.***out***.println("I'm in constructor method");

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

ob1[0] = **new** LoginTest();

ob1[1] = **new** LogoutTest();

**return** ob1;

}

}

Below is DataProviderFactory class file

**package** TestNG;

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

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

**public** **class** DataProviderFactory {

@DataProvider()

**public** Object[][] dp1() {

Object o1[][] = **new** Object[3][2];

o1[0][0] = 1;

o1[0][1] = "Jigar";

o1[1][0] = 2;

o1[1][1] = "Mehta";

o1[2][0] = 3;

o1[2][1] = "Learning";

**return** o1;

}

@DataProvider

**public** Object[][] dp2() {

Object o1[][] = **new** Object[3][2];

o1[0][0] = 11;

o1[0][1] = "TestNG";

o1[1][0] = 12;

o1[1][1] = "is";

o1[2][0] = 13;

o1[2][1] = "easy";

**return** o1;

}

@DataProvider()

**public** Object[][] dp3() {

Object o1[][] = **new** Object[3][2];

o1[0][0] = 21;

o1[0][1] = "How's";

o1[1][0] = 22;

o1[1][1] = "Your";

o1[2][0] = 23;

o1[2][1] = "Day";

**return** o1;

}

@DataProvider(name="get\_Constructor\_Parameter")

**public** Object[][] get\_Constructor\_Parameter() {

**return** **new** Object[][] {{"a"},{"b"},{"c"}};

}

}

Run FactoryClass using below TestNg.xml

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

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

<suite name=*"Suite1"* thread-count=*"5"*>

<test name=*"Test1"*>

<classes>

<class name=*"TestNG.FactoryClass"*></class>

</classes>

</test>

</suite>

Output:

[RemoteTestNG] detected TestNG version 6.14.2

I'm in constructor method

Inside Logintest

Inside call conference

id = 1

Password = Jigar

Inside Logintest

id = 2

Password = Mehta

Inside Logintest

id = 3

Password = Learning

Inside Logintest

id = 11

Password = TestNG

Inside Logintest

id = 12

Password = is

Inside Logintest

id = 13

Password = easy

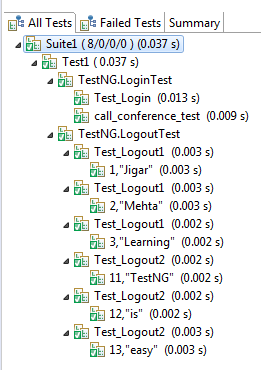
Inside Logintest

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

Suite1

Total tests run: 8, Failures: 0, Skips: 0

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



**@Factory with @DataProviders**

**package** TestNG;

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

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

**public** **class** FactoryClass {

@Factory(dataProvider="dp1",dataProviderClass=DataProviderFactory.**class**)

**public** Object[] FactoryAnnotationTransformerExample1(**int** id, String name) {

System.***out***.println("I'm in constructor method");

Object[] ob1 = **new** Object[1];

System.***out***.println("Id = " + id + " Name = " + name);

ob1[0] = **new** LoginTest();

**return** ob1;

}

@Test

**public** **void** sampletest() {

System.***out***.println("inside sample test");

}

}

**package** TestNG;

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

**public** **class** LoginTest {

**public** LoginTest() {

System.***out***.println("constructor...");

}

@Test()

**public** **void** LoginTest1() {

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

}

/\*

\* @Test() public void call\_conference\_test() {

\* System.out.println("Inside call conference"); }

\*/}

**package** TestNG;

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

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

**public** **class** DataProviderFactory {

@DataProvider()

**public** Object[][] dp1() {

Object o1[][] = **new** Object[3][2];

o1[0][0] = 1;

o1[0][1] = "Jigar";

o1[1][0] = 2;

o1[1][1] = "Mehta";

o1[2][0] = 3;

o1[2][1] = "Learning";

**return** o1;

}

@DataProvider

**public** Object[][] dp2() {

Object o1[][] = **new** Object[3][2];

o1[0][0] = 11;

o1[0][1] = "TestNG";

o1[1][0] = 12;

o1[1][1] = "is";

o1[2][0] = 13;

o1[2][1] = "easy";

**return** o1;

}

@DataProvider()

**public** Object[][] dp3() {

Object o1[][] = **new** Object[3][2];

o1[0][0] = 21;

o1[0][1] = "How's";

o1[1][0] = 22;

o1[1][1] = "Your";

o1[2][0] = 23;

o1[2][1] = "Day";

**return** o1;

}

@DataProvider(name="get\_Constructor\_Parameter")

**public** Object[][] get\_Constructor\_Parameter() {

**return** **new** Object[][] {{"a"},{"b"},{"c"}};

}

}

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

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

<suite name=*"Suite1"* thread-count=*"5"*>

<test name=*"Test1"*>

<classes>

<class name=*"TestNG.FactoryClass"*></class>

</classes>

</test>

</suite>

Output:

[RemoteTestNG] detected TestNG version 6.14.2

I'm in constructor method

Id = 1 Name = Jigar

constructor...

I'm in constructor method

Id = 2 Name = Mehta

constructor...

I'm in constructor method

Id = 3 Name = Learning

constructor...

inside sample test

Inside Logintest

Inside Logintest

Inside Logintest

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

Suite1

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

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

