TESTNG - PARAMETERIZED TEST

Another interesting feature available in TestNG is *parametric testing*. In most cases, you'll come across a scenario where the business logic requires a hugely varying number of tests. *Parameterized tests* allow developers to run the same test over and over again using different values.

TestNG lets you pass parameters directly to your test methods in two different ways:

- With testng.xml
- With Data Providers

Passing Parameters with *testng.xml*

With this technique, you define the simple parameters in the *testng.xml* file and then reference those parameters in the source files. Let us have an example to demonstrate how to use this technique to pass parameters.

Create Test Case Class

- Create a java test class, say, ParameterizedTest1.java.
- Add test method parameterTest to your test class. This method takes a string as input parameter.
- Add the annotation *@Parameters " myName "* to this method. The parameter would be passed a value from testng.xml, which we will see in the next step.

Create a java class file named ParameterizedTest1.java in **C:\>TestNG_WORKSPACE**.

```
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class ParameterizedTest1 {
   @Test
   @Parameters("myName")
   public void parameterTest(String myName) {
      System.out.println("Parameterized value is : " + myName);
   }
}
```

Create testng.xml

Create testng.xml in **C:\>TestNG_WORKSPACE** to execute test cases.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Suite1">
   <test name="test1">

      <parameter name="myName" value="manisha"/>

      <classes>
         <class name="ParameterizedTest1" />
      </classes>

   </test>
</suite>
```

> *We can also define the parameters at the <suite> level. Suppose we have defined myName at both <suite> and <test> levels. In such cases, regular scoping rules apply. It means that any class inside <test> tag will see the value of parameter defined in <test>, while the classes in the rest of the testng.xml file will see the value defined in <suite>.*

Compile the test case class using `javac`.

```
C:\TestNG_WORKSPACE>javac ParameterizedTest1.java
```

Now, run testng.xml, which will run the *parameterTest* method. TestNG will try to find a parameter named *myName* first in the <test> tag, and then, if it can't find it, it searches in the <suit> tag that encloses it.

```
C:\TestNG_WORKSPACE>java -cp "C:\TestNG_WORKSPACE" org.testng.TestNG testng.xml
```

Verify the output.

```
Parameterized value is : manisha

===============================================
Suite1
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

TestNG will automatically try to convert the value specified in testng.xml to the type of your parameter. Here are the types supported:

- String
- int/Integer
- boolean/Boolean
- byte/Byte
- char/Character
- double/Double
- float/Float
- long/Long
- short/Short

Passing Parameters with *Dataproviders*

When you need to pass complex parameters or parameters that need to be created from Java *complexobjects*, *objectsreadfromapropertyfileoradatabase*, *etc.* , parameters can be passed using Data providers. A Data Provider is a method annotated with *@DataProvider*. This annotation has only one string attribute: its name. If the name is not supplied, the data provider's name automatically defaults to the method's name. A data provider returns an array of objects.

The following examples demonstrate how to use data providers. The first example is about @DataProvider using Vector, String or Integer as parameter, and the second example is about @DataProvider using object as parameter.

Example 1

Here, the @DataProvider passes Integer and Boolean as parameter.

Create Java class

Create a java class called PrimeNumberChecker.java. This class checks if the number is prime.

Create this class in **C:\>TestNG_WORKSPACE**.

```
public class PrimeNumberChecker {
   public Boolean validate(final Integer primeNumber) {

      for (int i = 2; i < (primeNumber / 2); i++) {
         if (primeNumber % i == 0) {
            return false;
         }
      }
      return true;
   }
}
```

Create Test Case Class

- Create a java test class, say, ParamTestWithDataProvider1.java.
- Define the method primeNumbers, which is defined as a Data provider using the annotation. This method returns an array of objects.
- Add the test method testPrimeNumberChecker to your test class. This method takes an Integer and Boolean as input parameters. This method validates if the parameter passed is a prime number.
- Add the annotation *@TestdataProvider =" test1 "* to this method. The attribute dataProvider is mapped to "test1".

Create a java class file named ParamTestWithDataProvider1.java in **C:\>TestNG_WORKSPACE**.

```
import org.testng.Assert;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class ParamTestWithDataProvider1 {
   private PrimeNumberChecker primeNumberChecker;

   @BeforeMethod
   public void initialize() {
      primeNumberChecker = new PrimeNumberChecker();
   }

   @DataProvider(name = "test1")
   public static Object[][] primeNumbers() {
      return new Object[][] {{2, true}, {6, false}, {19, true}, {22, false}, {23, true}};
   }

   // This test will run 4 times since we have 5 parameters defined
   @Test(dataProvider = "test1")
   public void testPrimeNumberChecker(Integer inputNumber, Boolean expectedResult) {
      System.out.println(inputNumber + " " + expectedResult);
      Assert.assertEquals(expectedResult,
      primeNumberChecker.validate(inputNumber));
   }
}
```

Create testng.xml

Create a testng.xml **C:\>TestNG_WORKSPACE** to execute Test cases.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Suite1">
   <test name="test1">
```

```
      <classes>
         <class name="ParamTestWithDataProvider1" />
      </classes>
   </test>
</suite>
```

Compile the Test case class using javac.

```
C:\TestNG_WORKSPACE>.javac ParamTestWithDataProvider1.java PrimeNumberChecker.java
```

Now, run testng.xml.

```
C:\TestNG_WORKSPACE>java -cp "C:\TestNG_WORKSPACE" org.testng.TestNG testng.xml
```

Verify the output.

```
   2 true
   6 false
   19 true
   22 false
   23 true

===============================================
   Suite1
   Total tests run: 5, Failures: 0, Skips: 0
===============================================
```

Example 2

Here, the @DataProvider passes Object as parameter.

Create Java class

Create a java class Bean.java, which is a simple object with get/set methods, in **C:\>TestNG_WORKSPACE**.

```
public class Bean {
   private String val;
   private int i;

   public Bean(String val, int i){
      this.val=val;
      this.i=i;
   }

   public String getVal() {
      return val;
   }

   public void setVal(String val) {
      this.val = val;
   }

   public int getI() {
      return i;
   }

   public void setI(int i) {
      this.i = i;
   }
}
```

Create Test Case Class

- Create a java test class, say, ParamTestWithDataProvider2.java.
- Define the method primeNumbers, which is defined as a data provider using annotation. This method returns an array of object.
- Add the test method testMethod to your test class. This method takes an object bean as parameter.
- Add the annotation *@TestdataProvider =" test1 "* to this method. The attribute dataProvider is mapped to "test1".

Create a java class file named ParamTestWithDataProvider2.java in **C:\>TestNG_WORKSPACE**.

```
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class ParamTestWithDataProvider2 {
   @DataProvider(name = "test1")
   public static Object[][] primeNumbers() {
      return new Object[][] { { new Bean("hi I am the bean", 111) } };
   }

   @Test(dataProvider = "test1")
   public void testMethod(Bean myBean) {
      System.out.println(myBean.getVal() + " " + myBean.getI());
   }
}
```

Create testng.xml

Create testng.xml in **C:\>TestNG_WORKSPACE** to execute test cases.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Suite1">
   <test name="test1">
      <classes>
         <class name="ParamTestWithDataProvider2" />
      </classes>
   </test>
</suite>
```

Compile the test case class using javac.

```
C:\TestNG_WORKSPACE>javac ParamTestWithDataProvider2.java Bean.java
```

Now, run testng.xml.

```
C:\TestNG_WORKSPACE>java -cp "C:\TestNG_WORKSPACE" org.testng.TestNG testng.xml
```

Verify the output.

```
   hi I am the bean 111

===============================================
   Suite1
   Total tests run: 1, Failures: 0, Skips: 0
===============================================
```