1.介绍

TestNG是一个测试框架，旨在简化广泛的测试需求，从单元测试（隔离其他类的情况下测试一个类）到集成测试（测试由数个类组织的整个系统，几个包甚至多个外部框架，如应用服务器）。

编写测试通常是一个三步过程：

写你的测试业务逻辑，并在代码中插入TestNG的注释。

加入关于你测试的信息到testng.xml文件或者build.xml中（例如类名，要运行组，等等）

运行TestNG。

此文档中使用的概念如下：

一个测试套由一个XML文件表示。它可以包含一个或多个测试，并且由<suite>标记来定义。

测试是通过<test>标记来描述，并且可以包含一个或多个TestNG的类。

一个TestNG的类是包含至少一个TestNG的注释的Java类。它是由<class>标记表示，并且可以包含一个或多个测试方法。

测试方法是一个Java方法由@Test在源代码注释。

一个TestNG的测试可以通过@BeforeXXX和@AfterXXX注解，在一些确定点之前和之后执行一些Java的逻辑。

2.注解

这里是TestNG的提供及其属性注释的简要概述

|  |  |  |
| --- | --- | --- |
| **@BeforeSuite @AfterSuite @BeforeTest @AfterTest @BeforeGroups @AfterGroups @BeforeClass @AfterClass @BeforeMethod @AfterMethod** | | **Configuration information for a TestNG class:**   **@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. |
|  | alwaysRun | For before methods (beforeSuite, beforeTest, beforeTestClass and beforeTestMethod, but not beforeGroups): If set to true, this configuration method will be run regardless of what groups it belongs to.  For after methods (afterSuite, afterClass, ...): If set to true, this configuration method will be run even if one or more methods invoked previously failed or was skipped. |
|  | dependsOnGroups | The list of groups this method depends on. |
|  | dependsOnMethods | The list of methods this method depends on. |
|  | enabled | Whether methods on this class/method are enabled. |
|  | groups | The list of groups this class/method belongs to. |
|  | inheritGroups | If true, this method will belong to groups specified in the @Test annotation at the class level. |
|  | | |
| **@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 name equals to the name of this annotation.** |
|  | name | The name of this data provider. If it's not supplied, the name of this data provider will automatically be set to the name of the method. |
|  | parallel | If set to true, tests generated using this data provider are run in parallel. Default value is false. |
|  | | |
| **@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.** |
|  | value | An array of classes that extend org.testng.ITestNGListener. |
|  | | |
| **@Parameters** | | **Describes how to pass parameters to a @Test method.** |
|  | value | The list of variables used to fill the parameters of this method. |
|  | | |
| **@Test** | | **Marks a class or a method as part of the test.** |
|  | alwaysRun | If set to true, this test method will always be run even if it depends on a method that failed. |
|  | dataProvider | The name of the data provider for this test method. |
|  | dataProviderClass | The class where to look for the data provider. If not specified, the data provider will be looked on the class of the current test method or one of its base classes. If this attribute is specified, the data provider method needs to be static on the specified class. |
|  | dependsOnGroups | The list of groups this method depends on. |
|  | dependsOnMethods | The list of methods this method depends on. |
|  | description | The description for this method. |
|  | enabled | Whether methods on this class/method are enabled. |
|  | 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. |
|  | groups | The list of groups this class/method belongs to. |
|  | invocationCount | The number of times this method should be invoked. |
|  | invocationTimeOut | The maximum number of milliseconds this test should take for the cumulated time of all the invocationcounts. This attribute will be ignored if invocationCount is not specified. |
|  | priority | The priority for this test method. Lower priorities will be scheduled first. |
|  | successPercentage | The percentage of success expected from this method |
|  | 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). |
|  | timeOut | The maximum number of milliseconds this test should take. |
|  | threadPoolSize | The size of the thread pool for this method. The method will be invoked from multiple threads as specified by invocationCount.  Note: this attribute is ignored if invocationCount is not specified |

3.testng.xml

您可以通过几种不同的方式调用TestNG：

用testng.xml文件

通过ant

通过命令行

本章节描述testng.xml的格式

下面是testng.xml文件的一个示例

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

<suite name="Suite1" verbose="1" >

<test name="Nopackage" >

<classes>

<class name="NoPackageTest" />

</classes>

</test>

<test name="Regression1">

<classes>

<class name="test.sample.ParameterSample"/>

<class name="test.sample.ParameterTest"/>

</classes>

</test>

</suite>

你可以指定一个包的名字,而不是多个类名

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

<suite name="Suite1" verbose="1" >

<test name="Regression1" >

<packages>

<package name="test.sample" />

</packages>

</test>

</suite>

在这个例子中，TestNG的查找包test.sample中所有类并且将仅仅保留具有TestNG注解的类。

您还可以指定组和方法，包括和排除：

<test name="Regression1">

<groups>

<run>

<exclude name="brokenTests" />

<include name="checkinTests" />

</run>

</groups>

<classes>

<class name="test.IndividualMethodsTest">

<methods>

<include name="testMethod" />

</methods>

</class>

</classes>

</test>

你也可以在testng.xml中定义新的组，并且通过属性指定其他详细信息，如通过并发跑测试，使用多少线程，运行JUnit测试等

默认情况下，TestNG根据从XML文件中出现测试用例的顺序执行，如果你想以不可预知的顺序运行，设置属性preserve-order为false

<test name="Regression1" preserve-order="false">

<classes>

<class name="test.Test1">

<methods>

<include name="m1" />

<include name="m2" />

</methods>

</class>

<class name="test.Test2" />

</classes>

</test>

4.运行TestNG

TestNG可以用不同方式来调用:

命令行

Ant

Eclipse

IntelliJ’s IDEA

本节仅介绍如何从从命令行调用TestNG的。

假设在你的类路径中有TestNG，调用TestNG的最简单的方法如下：

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

您需要至少指定一个XML文件，描述您要运行TestNG的套件。另外，以下命令行开关可用：

| Command Line Parameters | | |
| --- | --- | --- |
| **Option** | **Argument** | **Documentation** |
| -configfailurepolicy | skip|continue | Whether TestNG should continue to execute the remaining tests in the suite or skip them if an @Before\* method fails. Default behavior is skip. |
| -d | A directory | The directory where the reports will be generated (defaults to test-output). |
| -dataproviderthreadcount | The default number of threads to use for data providers when running tests in parallel. | This sets the default maximum number of threads to use for data providers when running tests in parallel. It will only take effect if the parallel mode has been selected (for example, with the -parallel option). This can be overridden in the suite definition. |
| -excludegroups | A comma-separated list of groups. | The list of groups you want to be excluded from this run. |
| -groups | A comma-separated list of groups. | The list of groups you want to run (e.g. "windows,linux,regression"). |
| -listener | A comma-separated list of Java classes that can be found on your classpath. | Lets you specify your own test listeners. The classes need to implement [org.testng.ITestListener](http://testng.org/javadocs/org/testng/ITestListener.html) |
| -methods | A comma separated list of fully qualified class name and method. For example com.example.Foo.f1,com.example.Bar.f2. | Lets you specify individual methods to run. |
| -methodselectors | A comma-separated list of Java classes and method priorities that define method selectors. | Lets you specify method selectors on the command line. For example: com.example.Selector1:3,com.example.Selector2:2 |
| -parallel | methods|tests|classes | If specified, sets the default mechanism used to determine how to use parallel threads when running tests. If not set, default mechanism is not to use parallel threads at all. This can be overridden in the suite definition. |
| -reporter | The extended configuration for a custom report listener. | Similar to the -listener option, except that it allows the configuration of JavaBeans-style properties on the reporter instance.  Example: -reporter com.test.MyReporter:methodFilter=\*insert\*,enableFiltering=true  You can have as many occurences of this option, one for each reporter that needs to be added. |
| -sourcedir | A semi-colon separated list of directories. | The directories where your javadoc annotated test sources are. This option is only necessary if you are using javadoc type annotations. (e.g."src/test" or "src/test/org/testng/eclipse-plugin;src/test/org/testng/testng"). |
| -suitename | The default name to use for a test suite. | This specifies the suite name for a test suite defined on the command line. This option is ignored if the suite.xml file or the source code specifies a different suite name. It is possible to create a suite name with spaces in it if you surround it with double-quotes "like this". |
| -testclass | A comma-separated list of classes that can be found in your classpath. | A list of class files separated by commas (e.g. "org.foo.Test1,org.foo.test2"). |
| -testjar | A jar file. | Specifies a jar file that contains test classes. If a testng.xml file is found at the root of that jar file, it will be used, otherwise, all the test classes found in this jar file will be considered test classes. |
| -testname | The default name to use for a test. | This specifies the name for a test defined on the command line. This option is ignored if the suite.xml file or the source code specifies a different test name. It is possible to create a test name with spaces in it if you surround it with double-quotes "like this". |
| -testnames | A comma separated list of test names. | Only tests defined in a <test> tag matching one of these names will be run. |
| -testrunfactory | A Java classes that can be found on your classpath. | Lets you specify your own test runners. The class needs to implement [org.testng.ITestRunnerFactory](http://testng.org/javadocs/org/testng/ITestRunnerFactory.html). |
| -threadcount | The default number of threads to use when running tests in parallel. | This sets the default maximum number of threads to use for running tests in parallel. It will only take effect if the parallel mode has been selected (for example, with the -parallel option). This can be overridden in the suite definition. |
| -xmlpathinjar | The path of the XML file inside the jar file. | This attribute should contain the path to a valid XML file inside the test jar (e.g. "resources/testng.xml"). The default is "testng.xml", which means a file called "testng.xml" at the root of the jar file. This option will be ignored unless -testjar is specified. |

本文档可以通过调用TestNG的不带任何参数来获得。

你也可以把参数放入一个文本文件，例如c：\ command.txt，并告诉TestNG使用那个文件来检索它的参数：

C：>more C：\ command.txt

-d test-output testng.xml

C：> java org.testng.TestNG @c：\ command.txt

此外，TestNG的可以在通过命令行Java虚拟机的命令行上被传递的性能，例如

java -Dtestng.test.classpath="c:/build;c:/java/classes;" org.testng.TestNG testng.xml

下面是TestNG能理解的属性：

| System properties | | |
| --- | --- | --- |
| **Property** | **Type** | **Documentation** |
| testng.test.classpath | A semi-colon separated series of directories that contain your test classes. | If this property is set, TestNG will use it to look for your test classes instead of the class path. This is convenient if you are using the package tag in your XML file and you have a lot of classes in your classpath, most of them not being test classes. |

例：

java org.testng.TestNG -groups windows,linux -testclass org.test.MyTest

**ant任务和testng.xml让你启动TestNG的降低更多的参数（包含方法，指定参数，等...），所以你应考虑使用只当你试图了解TestNG的命令行，你想快速启动和运行。**

重要信息：命令行标志指定哪些测试，应运行，如果您还指定了文件的testng.xml将被忽略，与-includegroupids和-excludedgroups例外，这将覆盖所有的组夹杂物/排除在发现的testng.xml 。

5.测试方法，测试类和测试组

5.1测试方法

测试方法是添加了@Test标注。@Test注释的方法存在返回值时将被忽略，除非你设置allow-return-values为true在你的testng.xml：

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

or

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

5.2 测试组

**TestNG允许你执行的测试方法复杂的分组。您不仅可以声明方法都属于组，但您也可以指定包含其他组的组。然后TestNG的可以调用，并要求包括一组特定的群体（或正则表达式）的同时排除另一组。这给了你在你如何分割你的测试最大的灵活性，如果你想运行两套不同的测试背靠背不需要你重新编译什么。**

public class Test1 {

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

public void testMethod1() {

}

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

public void testMethod2() {

}

@Test(groups = { "functest" })

public void testMethod3() {

}

}

<test name="Test1">

<groups>

<run>

<include name="functest"/>

</run>

</groups>

<classes>

<class name="example1.Test1"/>

</classes>

</test>

5.6 参数

测试方法不必是无参的。可以在每个测试方法的使用任意数量参数，你可以指示TestNG的为你传递正确的参数通过@Parameters注解。

有两种方法来设置这些参数：使用testng.xml或编程。

5.6.1 通过testng.xml传参

如果你的参数使用的是简单的值，你可以在testng.xml指定它们：

@Parameters({ "first-name" })

@Test

public void testSingleString(String firstName) {

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

assert "Cedric".equals(firstName);

}

在这段代码中，我们指定的Java方法的参数firstName 应该收到称为firstName 的XML参数的值。此XML参数定义在testng.xml中

<suite name="My suite">

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

<test name="Simple example">

<-- ... -->

同样的技术可以使用在@Before/After and @Factory 注解

@Parameters({ "datasource", "jdbcDriver" })

@BeforeMethod

public void beforeTest(String ds, String driver) {

m\_dataSource = ...; // look up the value of datasource

m\_jdbcDriver = driver;

}

这个时候，两个java 参数 ds 和driver将分别接收属性datasource和jdbc-driver的值  
参数可以声明为可选的使用@Optional注释：

@Parameters("db")

@Test

public void testNonExistentParameter(@Optional("mysql") String db) { ... }

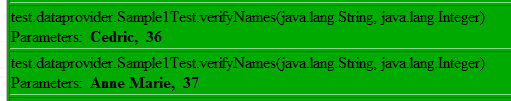
如果testng.xml中找不到名为”db”的参数，你的测试方法将采用默认值通过@Optional 注解 ：”mysql”。

@Parameters 注解能被放置在如下所指的位置：

任何注解为 @Test,@Before/After or @Factory

5.6.3 报告中的参数

**参数用于调用你的测试方法被展示由TestNG生成的HTML报告。这是一个例子:**



5.9 类层面的注解

@Test

public class Test1 {

public void test1() {

}

public void test2() {

}

}

@Test

public class Test1 {

public void test1() {

}

@Test(groups = "g1")

public void test2() {

}

}