**5.11 - JUnit tests**

**TestNG可以运行junit测试。所需要的只是在testng.classNames属性中指定junit测试类，并设置testng.junit属性为true。**

**<test name="Test1" junit="true">**

**<classes>**

**<!-- -->**

**这种情况下TestNG的行为类似jnit：**

**\* 类中所有以test\*开头的方法将被运行。**

**\* 如果测试类中有方法setUp(), 将在每次测试方法调用前被执行。**

**\* 如果测试类中有方法tearDown(),将在每次测试方法调用后被执行。**

**5.12 - JDK 1.4**

**TestNG也可以在JDK1.4下工作。在这种情况下，需要使用发布的jdk1.4的jar文件（名为testng-...-jdk14.jar）。唯一的差别是在于注解，jdk1.4下使用流行的XDoclet javadoc注解语法：**

**public class SimpleTest {**

**/\*\***

**\* @testng.before-class = "true"**

**\*/**

**public void setUp() {**

**// code that will be invoked when this test is instantiated**

**}**

**/\*\***

**\* @testng.test groups = "functest" dependsOnGroups = "group1,group2"**

**\*/**

**public void testItWorks() {**

**// your test code**

**}**

**}**

**javadoc语法的规则非常简洁，和jdk1.5注解的唯一差别是数组串数组需要特别写成单独的，逗号或空格分隔的字符串。虽然值周围的双引号是可选的，但还是建议在任何情况下都使用双引号，以保证将来迁移到jdk1.5时可以比较容易。**

**同样需要在<testng>的ant任务中指明sourcedir属性(或者在命令行中使用-sourcedir），以便testNG可以找到你的测试文件的源代码来解析javadoc注解。**

**这里是jdk1.4和jdk5注解的语法对照表：**

**（表格在blog中不好排版，不在这里发了，详细内容请参考官方文档的原文：http://testng.org/doc/documentation-main.html#jdk-14。）**

**更多jdk1.4的支持范例，请参考发行包中的test-14文件夹，这里包含全部的JDK 1.5测试对应的使用javadoc注解的内容。**

**5.13 - Running TestNG programmatically**

**在自己的程序中调用testNG也很简单:**

**TestListenerAdapter tla = new TestListenerAdapter();**

**TestNG testng = new TestNG();**

**testng.setTestClasses(new Class[] { Run2.class });**

**testng.addListener(tla);**

**testng.run();**

**这个范例创建了一个TestNG对象并运行测试类Run2。还增加了一个TestListener。你可以使用适配器类org.testng.TestListenerAdapter或自己实现org.testng.ITestListener。这个接口包含多个回调方法，使得可以追踪测试的开始，成功，失败等等。**

**类似的，可以使用testng.xml文件调用TestNG或者自己创建一个虚拟的testng.xml文件。为了做到这点，需要使用org.testng.xml包的类：XmlClass, XmlTest, 等等。每个类对应他们xml标签。**

**例如，假设你想创建下面的虚拟文件：**

**<suite name="TmpSuite" >**

**<test name="TmpTest" >**

**<classes>**

**<class name="test.failures.Child" />**

**<classes>**

**</test>**

**</suite>**

**你将使用下面的代码：**

**XmlSuite suite = new XmlSuite();**

**suite.setName("TmpSuite");**

**XmlTest test = new XmlTest(suite);**

**test.setName("TmpTest");**

**List<XmlClass> classes = new ArrayList<XmlClass>();**

**classes.add(new XmlClass("test.failures.Child"));**

**test.setXmlClasses(classes) ;**

**然后你可以将XmlSuite传递给TestNG：**

**List<XmlSuite> suites = new ArrayList<XmlSuite>();**

**suites.add(suite);**

**TestNG tng = new TestNG();**

**tng.setXmlSuites(suites);**

**tng.run();**

**完整的API请参考javadoc。**

**5.14 - BeanShell and advanced group selection**

**如果testng.xml中的<include>和<exclude>标签还不足够满足你的需要，你可以使用BeanShell表达式来决定是否需要将一个特定的测试方法包含在测试操作中。只需要在<test>标签下指定这个表达式：**

**<test name="BeanShell test">**

**<method-selectors>**

**<method-selector>**

**<script language="beanshell"><![CDATA[**

**groups.containsKey("test1")**

**]]></script>**

**</method-selector>**

**</method-selectors>**

**<!-- -->**

**当发现testng.xml中有<script>标签，TestNG将忽略当前<test>标签中的以后的组和方法的<include>和<exclude>标签：BeanShell表达式将是决定一个测试方法是否包含的唯一方法。**

**这里有一些BeanShell脚本的额外信息：**

**\* 必须返回boolean值。除了这个约束，任何有效的BeanShell代码都被容许.(例如，你可能想在工作日返回true而在周末返回false，这将容许你更加日期不同差异性的运行测试。**

**\* TestNG为了便利定义了以下变量：**

**java.lang.reflect.Method method: 当前测试方法**

**org.testng.ITestNGMethod testngMethod: 当前测试方法的描述**

**java.util.Map<String, String> groups: 当前测试方法所属组的Map**

**\* 你可能需要在你的表达式前后增加CDATA声明（如上面所示）以避免讨厌的xml转义字符**