junit教程:JUNIT基本教程

疯狂代码 http://CrazyCoder.cn/ ĵ:http:/CrazyCoder.cn/Java/Article47006.html

junit是java中书写unit testframework目前些流行unit test工具大都都是在junit上扩展而来

Eclipse中配置junit

在要使用JUNITproject名上点击properties--java build path-libraries, 点击Add External JARs,把JUNIT包点上就行了. 并在需要测试项目上新建junit test

使用方法

1. 基本使用步骤Junit使用非常简单它基本使用步骤:

- 创建从junit.framework.TestCase派生unit test需要test

- 书写测试思路方法提供类似于如下签名测试思路方法:

public void testXXXXX;

- 编译书写完test 后编译所写test 类

- 运行启动junit test runner来运行这个test

Junit提供了2个基本test runner:界面和图形界面启动命令分别如下:

a 图形界面:

```
java junit.swingui.TestRunner XXXXX
```

b 界面:

```
java junit.textui.TestRunner XXXXX
```

2. 使用例子:

```
import junit.frmework.TestCase;
public  TestSample extends TestCa{
```

```
public void testMethod1{
assertTrue( true);
}
}
```

3. Up和tearDown这两个是junit framework中提供化和反化每个测试思路方法Up在每个测试思路方法前被负责化测试思路方法所需要测试环境；tearDown在每个测试思路方法被的后被负责撤销测试环境它们和测试思路方法关系可以描述如下:

测试开始 -> Up -> testXXXX -> tearDown ->测试结束

4. 使用例子:

```
import junit.frmework.TestCase;
public  TestSample extends TestCa{
protected void Up{
//化……
}
public void testMethod1{
assertTrue( true);
}
potected void tearDown{
//撤销化……
}
}
```

5. 区分fail、exception

- fail期望出现产生原因:assert出错(如assertFalse(true))；fail产生(如fail(……))

- exception不期望出现属于unit test运行时抛出异常它和普通代码运行过程中抛出runtime异常属于种类型

对于assert、fail等请参见junitjavadoc

6. 使用例子:

```
import junit.frmework.TestCase;
public  TestSample extends TestCa{
protected void Up{
//化……
}
public void testMethod1{
……
try{
boolean b= ……
assertTrue( b);
throw  Exception( “This is a test.”);
fail( “Unable po.”); //不可能到达
}catch(Exception e){
fail( “Yes, I catch u”); //应该到达点
}
……
}
potected void tearDown{
//撤销化……
}
}
```

7. 组装TestSuite运行更多test在junit中Test、TestCase和TestSuite 3者组成了composiste pattern通过组装自己TestSuite可以完成对添加到这个TestSuite中所有TestCase而且这些定义TestSuite还可以组装成更大TestSuite这样同时也方便了对于不断增加TestCase管理和维护

它另个好处就是可以从这个TestCase树任意个节点(TestSuite或TestCase)开始来完成这个节点以下所有TestCase提高了unit test灵活性

8. 使用例子:

```
  import junit.framework.Test;
import junit.framework.TestSuite;
public  TestAll{
public  TestAll{
```

```
//定义个suite对于junit作用可以视为类似于java应用
public  Test suite{
TestSuite suite =  TestSuite("Running all tests.");
suite.addTestSuite( TestCase1.);
suite.addTestSuite( TestCase2.);
 suite;
}
}
```

运行同运行单独个TestCase是样参见step 1 “运行”

9. 使用Ant junit task我们除了使用java来直接运行junit的外我们还可以使用junit提供junit task和ant结合来运行涉及几个主要ant task如下:

- <junit>定义个junit task

- <batchtest>位于<junit>中运行多个TestCase

- <test>位于<junit>中运行单个TestCase

- <formatter>位于<junit>中定义个测试结果输出格式

- <junitreport>定义个junitreport task

- <report>位于<junitreport>中输出个junit report

具体语法请参见相关文档

10. 使用例子:

```
<junit prsummary="yes" haltonfailure="no">
<path>
<path refid="path"/>
<pathelement location="${dist.junit}"/>
</path>
<formatter type="brief" usefile="false"/>
```

```
<formatter type="xml"/>
<batchtest todir="${doc.junitReport}">
<file dir="${dist.junit}" s="**/*Test." />
</batchtest>
</junit>
<junitreport todir="${doc.junitReport}">
<file dir="${doc.junitReport}">
< name="TEST*-*.xml"/>
</file>
<report format="frames" styledir="${junit.styleDir}" todir="${doc.junitReport}"/>
</junitreport>
```

检查表

junit使用并不很难然而要书写个好TestCase却并非易事个不好TestCase往往是既浪费了时间也起不了实际作用相反个好TestCase不仅可以很好指出代码中存在问题而且也可以作为代码更准确文档同时还在持续集成过程中起非常重要作用在此给出书写TestCase时需要注意几点:

- 测试独立性:次只测试个对象方便定位出错位置这有2层意思:个TestCase只测试个对象；个TestMethod只测试这个对象中个思路方法

- 给测试思路方法个合适名字

- 在assert中给出失败原因如:assertTrue(“... should be true”，.....)方便查错在这个例子中如果无法通过assertTrue那么给出消息将被显示在junit中每个assert都有第个参数是出错时显示消息原型

- 测试所有可能引起失败地方如:个类中频繁改动对于那些仅仅只含有getter/ter类如果是由IDE(如Eclipse)产生则可不测；如果是人工写那么最好测试下

- 在Up和tearDown中代码不应该是和测试思路方法相关而应该是全局相关如针对和测试思路方法A和B在Up和tearDown中代码应该是A和B都需要代码

- 测试代码组织:相同包区别目录这样测试代码可以访问被测试类protected变量/思路方法方便测试代码编写放在区别目录则方便了测试代码管理以及代码打包和发布个例子如下:

src <=源代码根目录

|---com
|---mod1
|---1
junit <=测试代码根目录
|---com
|---mod1
|---1

2009-1-8 1:59:20
疯狂代码 http://CrazyCoder.cn/