近期写单元测试计划，公司准备把junit3.8升级到junit4以上的版本，所以研究了一下，写下学习过程和心得体会，为了巩固学习成果，所以把学习心得写下来加深印象，也供需要的朋友查阅，少走弯路。

好了，废话不多说，直接开始：

假设我们要写一个整数除法和乘法的类，并且给他写测试用例：

1) 建立Math类

工具是eclipse3.3

Java代码

/\*\*

\* @author bulargy.j.bai

\* @创建时间：Mar 10, 2008

\* @描述：一个整数除法和乘法的工具类

\*/

public class Math {

public static int divide(int x,int y) {

return x/y;

}

public static int multiple(int x,int y) {

return x\*y;

}

}

2) 建立测试用例

选中需要建立测试用例的包，选择new->other->JUnit Test Case。

有5个方法可以选择：

setUp()方法在测试方法前调用，一般用来做测试准备工作。

tearDown()方法在测试方法后调用，一般作测试的清理工作。

setUpBeforeClass()方法在整个类初始化之后调用，一般用来做测试准备工作。

tearDownAfterClass()方法在整个类结束之前调用，一般作测试的清理工作。

constructor()为是否包含构造方法。

自动生成的代码如下：

Java代码

/\*\*

\* @author bulargy.j.bai

\* @创建时间：Mar 11, 2008

\* @描述：

\*/

public class MathTest {

@BeforeClass

public static void setUpBeforeClass() throws Exception {

}

@AfterClass

public static void tearDownAfterClass() throws Exception {

}

@Test

public void testDivide() {

fail("Not yet implemented");

}

@Test

public void testMultiple() {

fail("Not yet implemented");

}

}

说明：

@BeforeClass标签注释的方法用于在整个类测试过程的初始化后调用一次，@AfterClass标签注释的方法则是整个测试类结束之前调用一次。这2个标签的搭配可以避免使用@Before、@After标签组合在每个测试方法前后都调用的弊端，减少系统开销，提高系统测试速度。(不过对环境独立性要求较高的测试还是应当使用@Before、@After来完成)

@Test标签用来标注待测试的方法，按照类中声明的顺序执行。

我们在testDivide方法加入测试代码，分别测试三种情况：

a. 完全正确也没有可能出错的数据，如：9除3 结果必须等于3

b. 可能有问题的边缘数据，如：10除3 结果也必须等于3

c. 错误的数据，如：10除0 必须抛出异常

忽略testMultiple方法

代码如下：

Java代码

@Test(expected=ArithmeticException.class)

public void testDivide() {

assertEquals(3,Math.divide(9,3));

assertEquals(3,Math.divide(10,3));

Math.divide(10,0); //除数不能为0，会抛出异常

}

@Ignore("忽略乘法测试")

@Test

public void testMultiple() {

}

@Test(expected=ArithmeticException.class)

public void testDivide() {

assertEquals(3,Math.divide(9,3));

assertEquals(3,Math.divide(10,3));

Math.divide(10,0); //除数不能为0，会抛出异常

}

说明:

Junit4为测试方法增加了判断异常的方式，避免了以前还要通过try/catch块捕捉异常再抛出的复杂方式，简单的这样声明“@Test(expected=ArithmeticException.class)”Junit4就会检查此方法是否抛出ArithmeticException异常，如果抛出则测试通过，没抛出则测试不通过（@Test标签还有一些其他参数，例如超时测试@Test(timeout=1)这样，但是由于并不能准确反应实际时间，所以应用较少，经过我测试误差太大绝对不适合拿来做超时测试的）

@Ignore标签会告诉Junit4忽略它所标注的方法，例如数据库不可用时可以用此标注标注一些测试数据库连接的方法来避免测试失败。