# Junit4单元测试浅谈

**什么是junit?**

Junit是一个Java编程语言的单元测试框架。Junit在测试驱动的开发方面有很重要的发展，是起源junit的一个统称为XUnit的单元测试框架之一。

JUnit促进了“先测试后编码”的理念，强调建立测试数据的一段代码，可以先测试，然后再应用。这个方法就好比“测试一点，编码一点，编码一点，测试一点”增加了程序员的产量和程序的稳定性，可以减少程序员的压力和花费在排错上的时间。

**特点：**

JUnit是一个开放的资源框架，用于编写和运行测试。

提供注释来识别测试方法。

提供断言来测试预期结果。

提供测试运行来运行测试。

JUnit测试允许你编写代码更快，并能提高质量。

JUnit优雅简洁，没那么复杂，花费时间较少。

JUnit测试可以自动运行并且检查自身结果并提供及时反馈，所以也没必要 人工梳理测试结果的报告。

JUnit测试可以被组织为测试套件，包含测试用例，甚至其他测试套件。

JUnit在一个条中显示进度，如果运行良好则是绿色，如果运行失败则是红 色。

**Junit环境配置**

JUnit是Java的一个框架，所以最根本的需要是在你的机器里装有jdk，jdk版本要求1.5或1.5以上。从http://www.junit.org下载JUnit最新版本的压缩文件。

**什么是JUnit测试框架?**

JUnit是一个回归测试框架，被开发者用于实施对应用程序的单元测试，加快程序编制速度，同时提高编码的质量，JUnit测试框架能够轻松的完成以下任意两种结合；

Eclipse集成开发环境

Ant打包工具

Maven项目构建管理

特性：

JUnit测试框架具有以下重要特性：

测试工具

测试套件

测试运行器

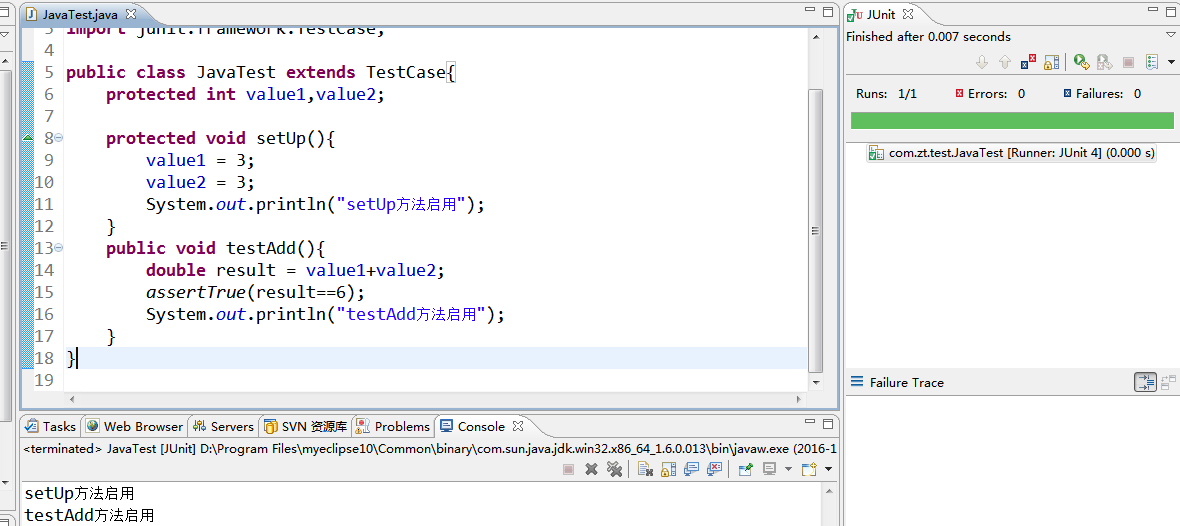
测试分类

测试工具：

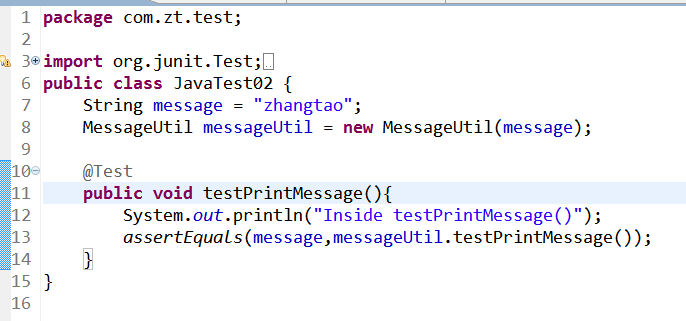
测试工具是一整套固定的工具用于基线测试。测试工具的目的是为了确保测试能够在共享且固定的环境中运行，因此保证测试结果的可重复性。它包括：

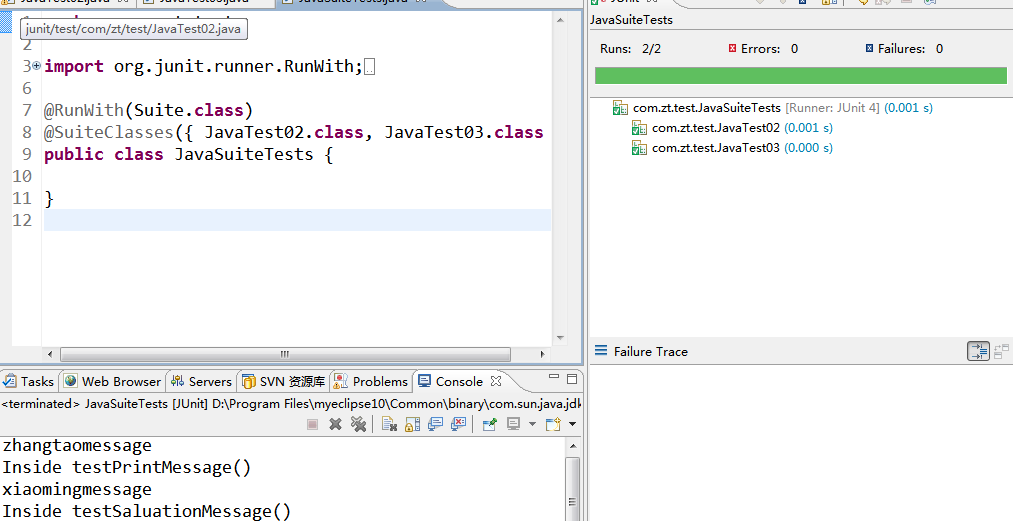
在所有测试调用指令发起前的setUp()方法

在测试方法运行后的tearDown()方法



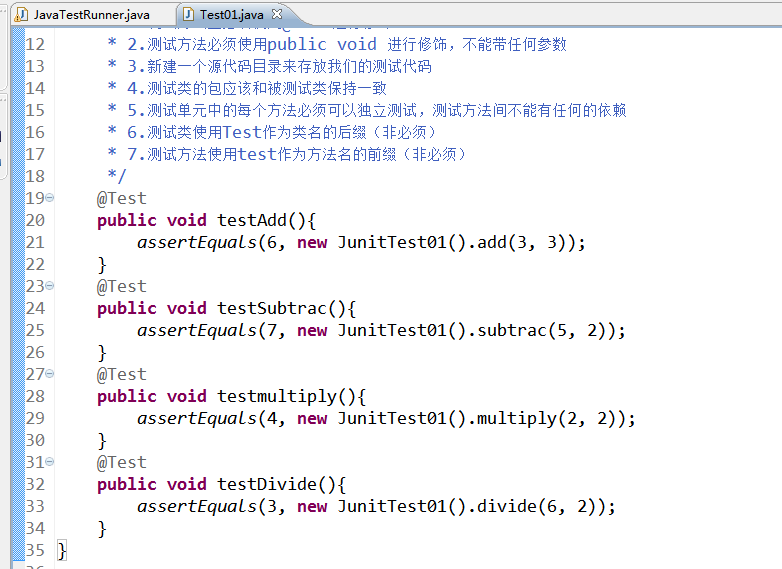
**测试套件：**

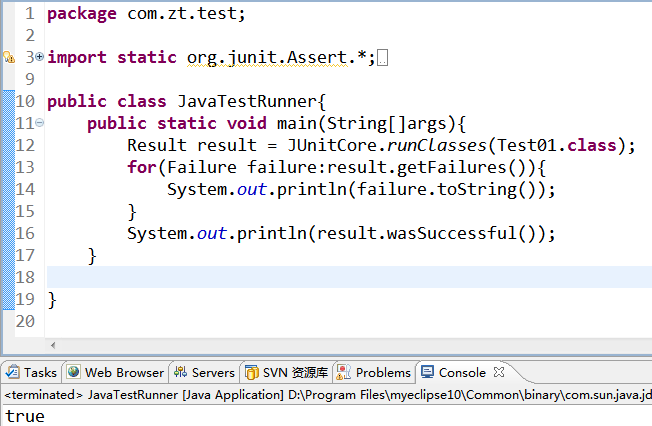
测试套件意味着捆绑几个测试案例并且同时运行。在junit中，@RunWith和@Suite都被用作运行测试套件。以下为使用JavaTest02和JavaTest03的测试分类：

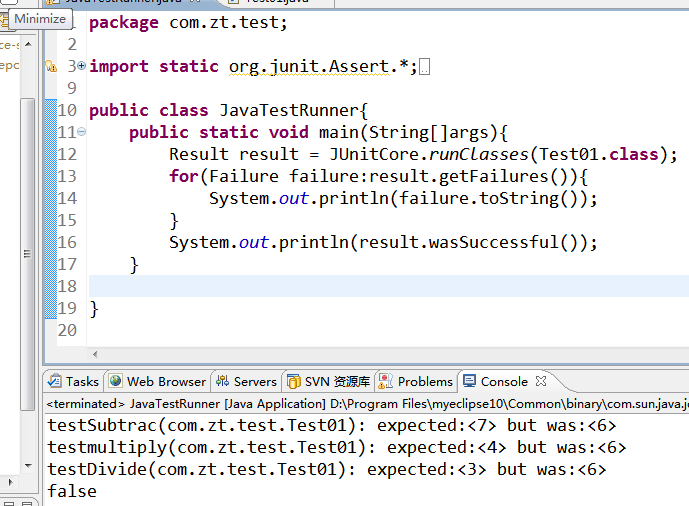


**测试运行器：**

用于执行测试案例。以下为假定测试分类成立的情况下的例子：







**断言测试：**

所有断言都包含在Assert类中

Public class Assert extends java.lang.Object

断言常用的方法：

void assertEquals(boolean expected,boolean actual)//检查两个变量或者等式是否平衡

void assertTrue(boolean expected,boolean actual)//检查条件为真

void assertFalse(boolean condition)//检查条件为假

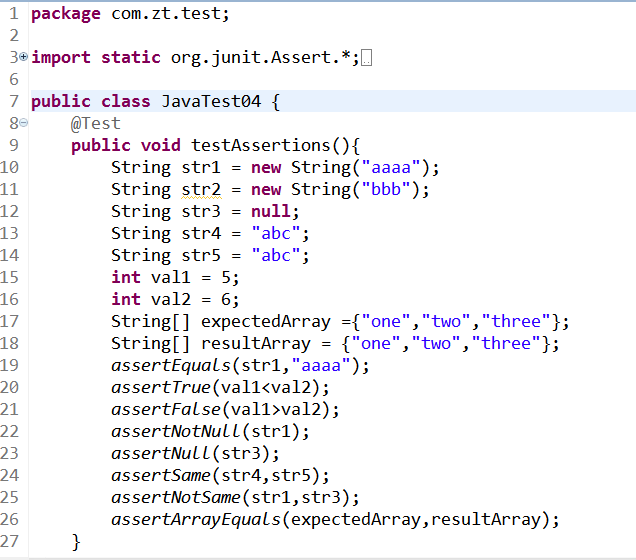
void assertNotNull(Object object)//检查对象不为空

void assertNull(Object object)//检查对象为空

void assertSame(boolean condition)//检查两个相关对象是否指向同一个对象

void assertNotSame(boolean condition)//检查两个相关数组对象是否不指向同一个对象

void assertArrayEquals(expectedArray,resultArray)//检查两个数组是否相等



**JUnit注释**

JUnit中的这些注释为我们提供了测试方法的相关信息，哪些方法在测试方法前后应用，哪些方法将会在所有方法应用前后应用，哪些方法将会在执行中被忽略。

@Test

这个注释说明依附在JUnit的public void方法作为一个测试案例。

@Before

有些测试在运行前需要创造几个相似的对象。在public void方法加该注释是因为该方法需要在test方法前运行。

@After

如果你将外部资源在Before方法中分配，那么你需要在测试运行后释放他们。在public void方法加该注释是因为该方法需要在test方法后运行

@BeforeClass

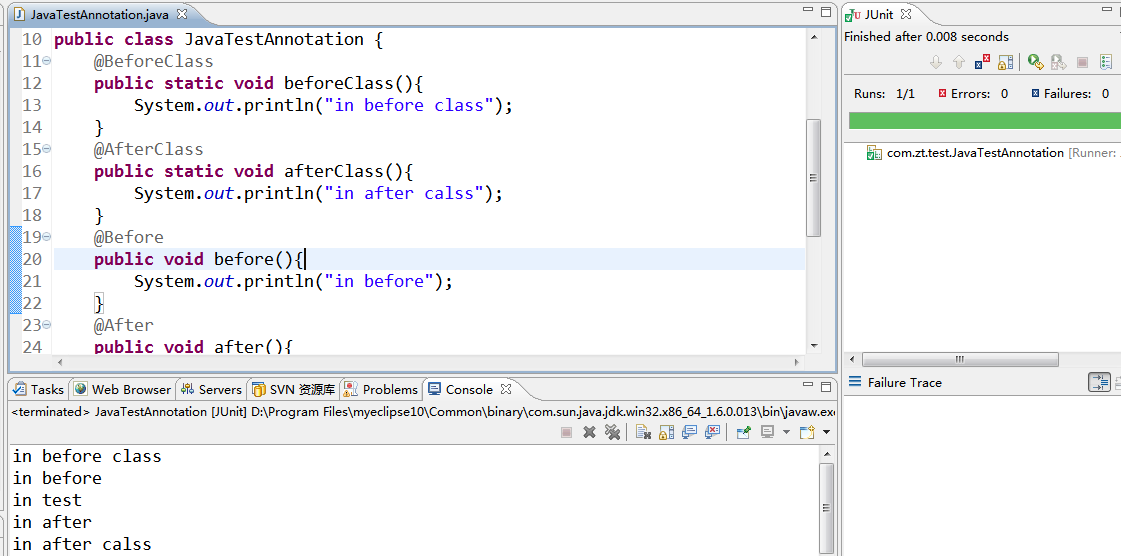
在public void方法加该注释是因为该方法需要在类中所有方法前运行

@AfterClass

它将会使方法在所有测试结束后执行。这个可以用来进行清理活动

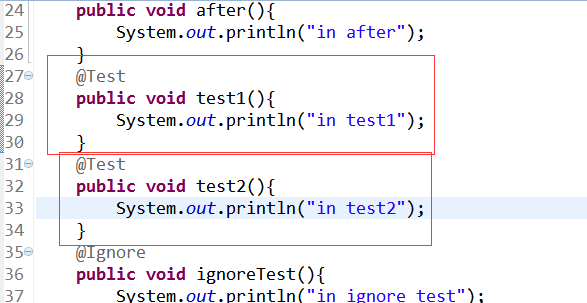
@Ignore

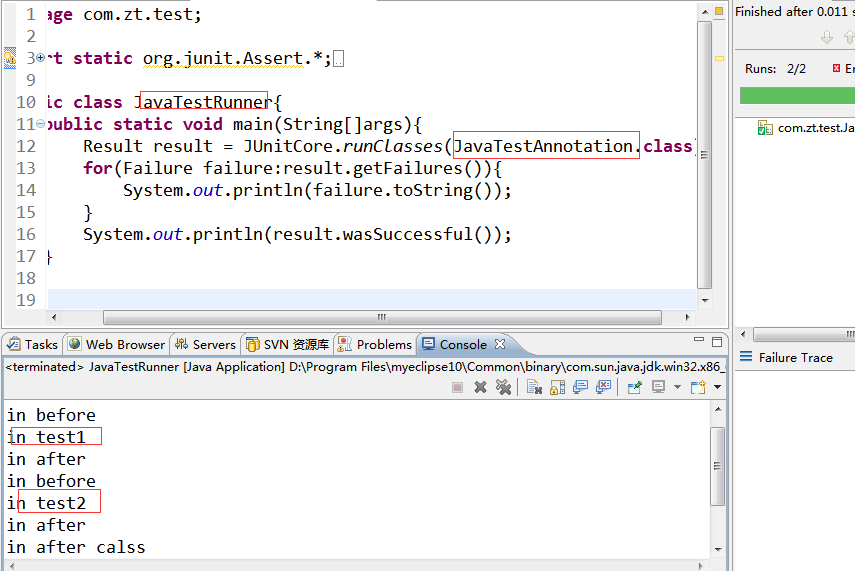
这个注释是用来忽略有关不需要执行的测试的



**JUnit执行过程：**

JUnit中的方法执行过程，即哪一个方法首先被调用，哪个方法在一个方法之后调用。





观察以上的输出，这是JUnit执行过程

beforClass()方法首先执行，并且只执行一次

afterClass()方法最后执行，并且只执行一次

before()方法针对每一个测试用例执行，但是在执行测试用例之前

after()方法针对每一个测试用例执行,但是是在执行测试用例之后

在before()方法和after()方法之间，执行每一个测试用例

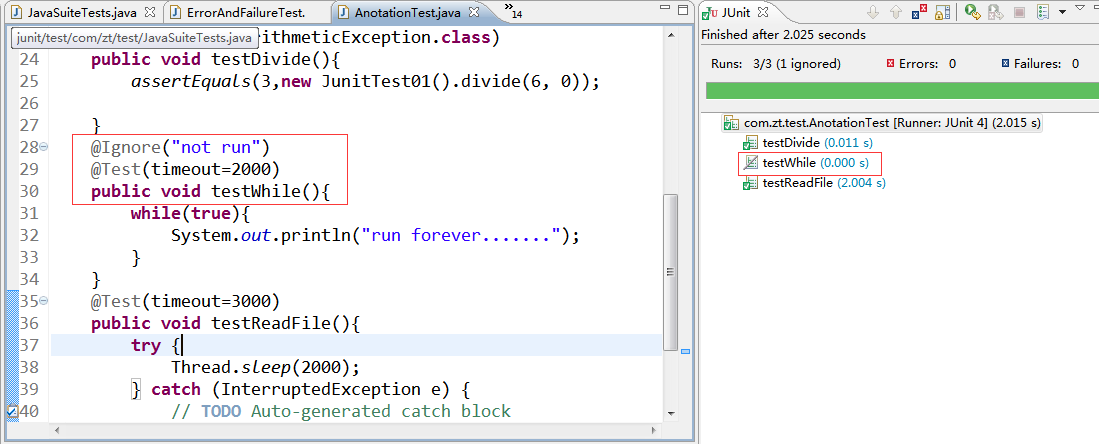
**忽略测试：**

有时可能会发生我们的代码还没准备好的情况，这时测试用例去测试这个方法或代码的时候会造成失败。@Ignore注释会在这种情况时帮助我们

一个含有@Ignore注释的测试方法将不会被执行

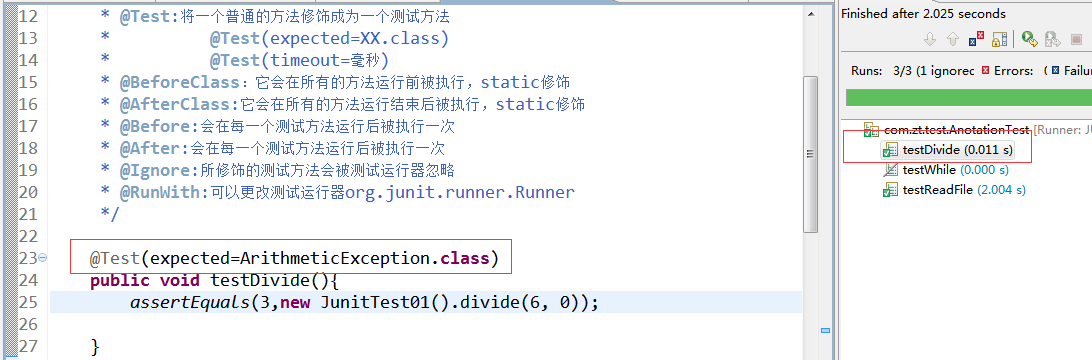
如果一个测试类有@Ignore注释，则它的测试方法将不会执行

下面这个例子用了测试时间限制



**JUnit异常测试:**

Junit用代码处理提供了一个追踪异常的选项。你可以测试代码是否它抛出了想要得到的异常。Excepted参数和@Test注释一起使用。现在看看以下示例：



**参数化测试：**

JUnit 4引入了一个新的功能参数化测试。参数化测试允许开发人员使用不同的值反复运行同一个测试。你将遵循5个步骤来创建参数化测试

1.[用@RunWith(Parameterized.class)来注释test类](mailto:用@RunWith(Parameterized.class)来注释test类)

2.创建一个由@Parameters注释的公共的静态方法，它返回一个对象的集合（数组）来作为测试数据集合

3.创建一个公共的构造函数，它接受和一行测试数据相等同的东西

4.为每一列测试数据创建一个实例变量

5.用实例变量作为测试数据的来源来创建你的测试用例

