Junit：一款免费开源的测试套件。用于验证程序的运行结果是否符合预期。本文档仅仅记录在maven工程下使用Junit的要点；pom.xml文件新增依赖：

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

## @Before测试前准备

写在方法上，测试前准备，被@Before注释的方法将会在@Test注释的方法之前运行，一般做准备基础数据，实验数据等工作。

## @Test测试

写在方法上，测试代码写在被@Test注释的方法中，测试类中的测试方法可以有多个，junit会全部执行，但不会按照顺序执行，若需要按要求执行，可参考@FixMethodOrder(后面讲)。方法中编写测试代码会用到Assert类

### Assert

|  |  |
| --- | --- |
| ~.assertArrayEquals(expecteds,actuals); | 比较期望值数组与实际值数组是否相等 |
| ~.assertEquals(expected,actual); | 比较期望值与实际值是否相等 |
| ~.assertNotEquals(expected,actual); | 比较期望值与实际值是否不相等 |
| ~.assertTrue(condition); | 比较实际值是否为true，为true则通过测试 |
| ~.assertFalse(condition); | 比较实际值是否为false，为false则通过测试 |
| ~.assertNull(object); | 比较对象是否为null |
| ~.assertNotNull(object); | 比较对象是否不为null |
| ~.assertSame(expected,actual); | 比较对象是否一样 |
| ~.assertNotSame(unexpected,actual); | 比较对象是否不一样 |

@Test方法支持时间验证——程序在指定时间内正确跑完算通过。例子如下

@Test(timeout = 10)

public void \_2() {

Double dou = 0.0;

for (int i = 0; i < 1000000; i++) {

dou += RandomUtils.nextDouble(0,100);

}

}

### 异常测试

若希望@Test方法时产生一个Exception异常表示测试通过，使用@Test的expected属性。

@Test(timeout = 10000,expected = ***NullPointerException.class***)

public void \_2() {

Double i = null;

Double j = 10.2;

Double k = i+j;

}

## @FixMethoder(MethodSorters.JVM)

写在测试类上，

MethodSorters.JVM：按jvm生成的hash值执行@Test

MethodSorters.NAME\_ASCENDING：按方法名字顺序执行@Test

MethodSorters.DEFAULT：默认方法，不可预期执行顺序

## @Ignore跳过@Test方法

写在方法上，被@Test注释的方法上加上@Ignore，junit执行时将跳过执行此方法。

## @After测试后收尾

写在方法上，测试后收尾，被@After注释的方法将会在@Test注释的方法之后运行，一般做：关闭数据连接，清理日志，缓存等工作

## 测试套件

批量运行测试类。

1. 创建一个被public修饰的空类，此类不带有任何方法属性，且具备一个不带有任何参数的构造函数；它将作为测试套件的入口；用@RunWith(Suite.class)和@Suite.SuiteClasses({...})修饰此空类。
2. 将需要测试的测试类放入@Suite.SuiteClasses({...})中

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

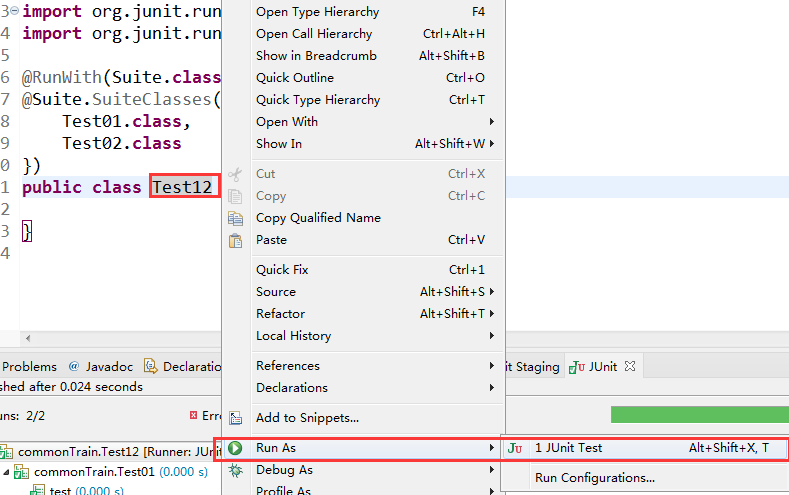
@Suite.SuiteClasses({***Test01.class, Test02.class***})

public class Test12 {

//类中不写任何属性及方法

}

1. 使用junit执行空类即可



## 参数化测试

自定义一组数据，交给@Test程序反复跑，看结果是否正确。

1. 测试类必须用@RunWith(Parameterized.class)修饰
2. 测试类定义的属性，除定义被测类外还定义与被测类方法中参数一样的属性，如果被测方法有返回值，也必须定义上去；定义构造函数。
3. 定义生成实验数据的函数，用@Parameterized.Parameters修饰。
4. 在@Test方法中，直接传入测试类中定义的属性，然后运行。

例子如下：

@RunWith(Parameterized.class)

public class CalculateTest {

//被测类中有多少参数，多少返回结果，就在类中定义多少

private Integer a;//参数1

private Integer b;//参数2

private Integer c;//返回值

private Calculate calculate;//被测试的类

//构造函数（必需）

public CalculateTest(Integer a,Integer b,Integer c) {

this.a = a;

this.b = b;

this.c = c;

}

//实验数据

@Parameterized.Parameters

public static Collection<Object[]> primeNumbers() {

return Arrays.asList(new Object[][] {

{1,2,3},

{10,20,30},

{12,21,33}

});

}

//初始化

@Before

public void init() {

this.calculate = new Calculate();

}

//测试

@Test

public void test() {

assertEquals(c, calculate.add(a, b));

}

}