Android：单元测试

leelit 于 2015-12-24 16:29:03 发布

前言

对于Android的单元测试一直有些弄不明白，虽然要进行单元测试并不复杂，但是其中的关系好像有点复杂，包括怎样在Eclipse进行单元测试，又怎样在Android Studio进行单元测试，怎样进行Local的测试而不需要运行真机或虚拟机，各种TestCase有什么不同，为什么Junit4好像很难work的样子…等等问题，我试着一一弄明白这些问题。这篇文章不涉及UI测试。

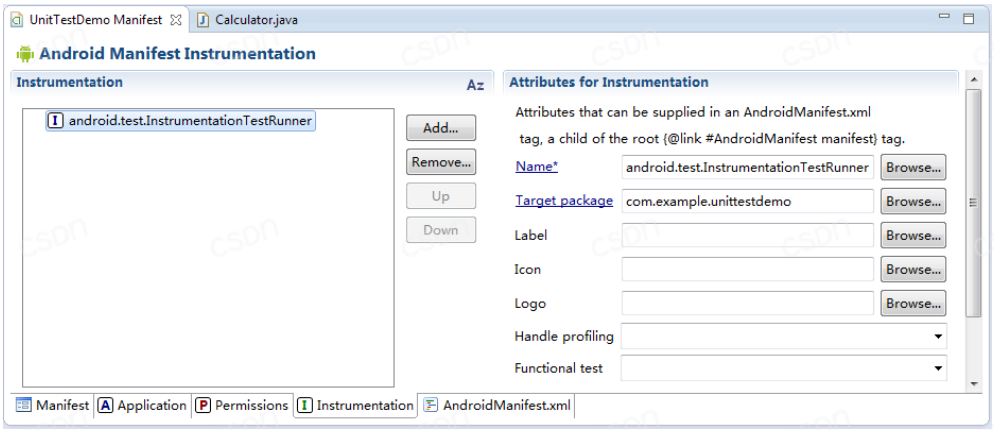
简介

> 单元测试是app的基础测试，为你的代码编写单元测试，可以很容易地验证程序的某个单元是否正确。当你改动代码之后，如果程序发生了错误，单元测试则可以帮助你快速发现这些错误。
> 单元测试以一种可重复的方式测试程序的最小单元，比如方法、类或者组件。为了验证特定代码单元的逻辑是否正确，你应该编写单元测试。通常情况下，单元测试与工程代码分离，测试只会影响到测试单元本身。而mocking框架可以将你的测试单元与其依赖分离。

ADT Eclipse进行单元测试

我们首先来看一下Eclipse怎样进行Android单元测试。

1、建立一个UnitTestDemo工程
2、为工程配置测试环境



然后在AndroidManifest文件Application节点下添加一条语句：

```
1 <uses-library android:name="android.test.runner" />
```

这样配置完成了

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="21" />

<instrumentation
    android:name="android.test.InstrumentationTestRunner"
    android:targetPackage="com.example.unittestdemo" >
</instrumentation>

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <uses-library android:name="android.test.runner" />

    <activity
        android:name=".MainActivity"
```

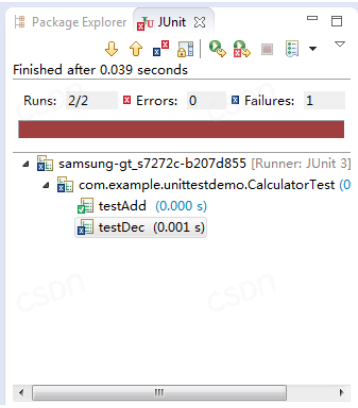
3、编写一个待测试的类，这个类我们会在后面多次用到。

```java
public class Calculator {

    public int add(int a, int b) {
        return a + b;
    }

    public int dec(int a, int b) {
        return a - b;
    }

}
```

4、为待测试类编写单元测试类，文件放在哪随意，最好把所有测试都放在新建的一个test包啦。

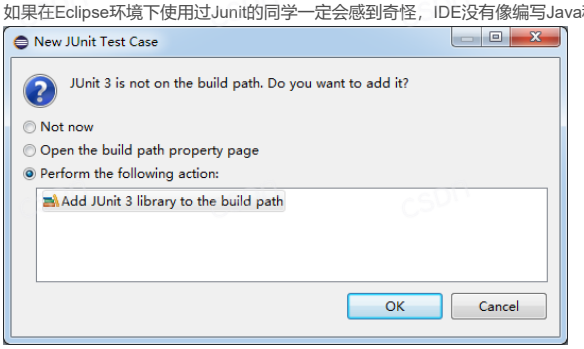
```java
import junit.framework.TestCase;

public class CalculatorTest extends TestCase {

    public void testAdd() {
        Calculator calculator = new Calculator();
        assertEquals(2, calculator.add(1, 1));
    }

    public void testDec() {
        fail("Not yet implemented");
    }

}
```


5、右击CalculatorTest类，Run As -> Android JUnit Test，启动真机或模拟器后即可看到测试结果了。



这样我们就完成了最简单的但也很实用的单元测试了。

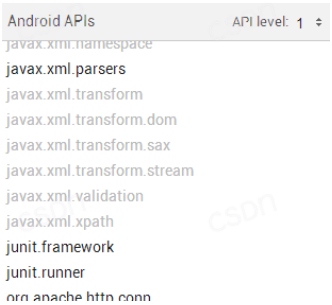
如果在Eclipse环境下使用过Junit的同学一定会感到奇怪，IDE没有像编写Java程序那样提示我们：



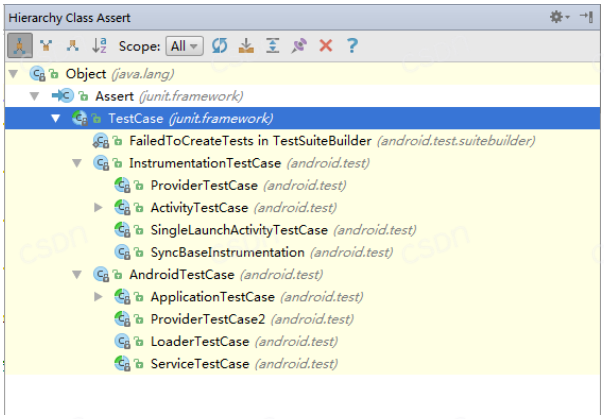
下一节来说明

认识

为什么我们Android工程只需要写了配置就可以进行单元测试，好像我们都没有导入Junit框架。原因是，Android SDK本身就内置了基于Junit3的测试框架。我们查下API历史，发现从API 1就已经内置了测试框架了。



基于Junit3，整个Android内置的测试框架如下图。



关于各种测试有何不同可以参考：

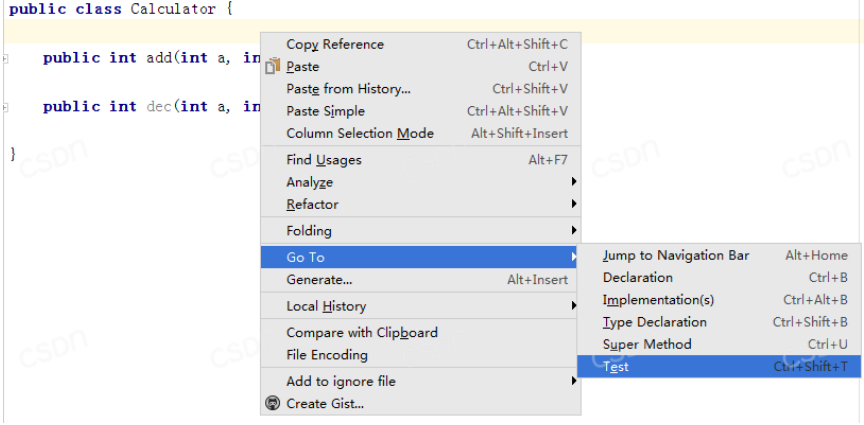
> http://developer.android.com/reference/android/test/package-summary.html
> http://stackoverflow.com/questions/2047261/using-android-test-framework/2055455#2055455

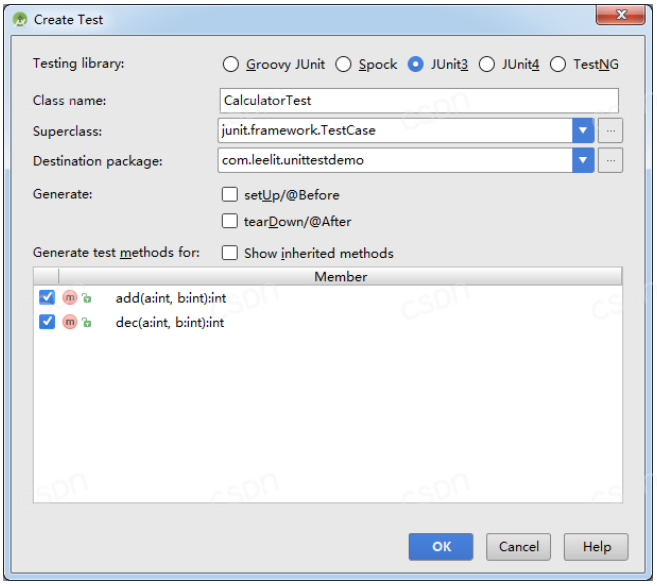
因为不涉及UI测试，需要用到Context就继承AndroidTestCase，有个getContext方法；不需要直接就继承TestCase和Junit3一样使用。

Android Studio进行单元测试

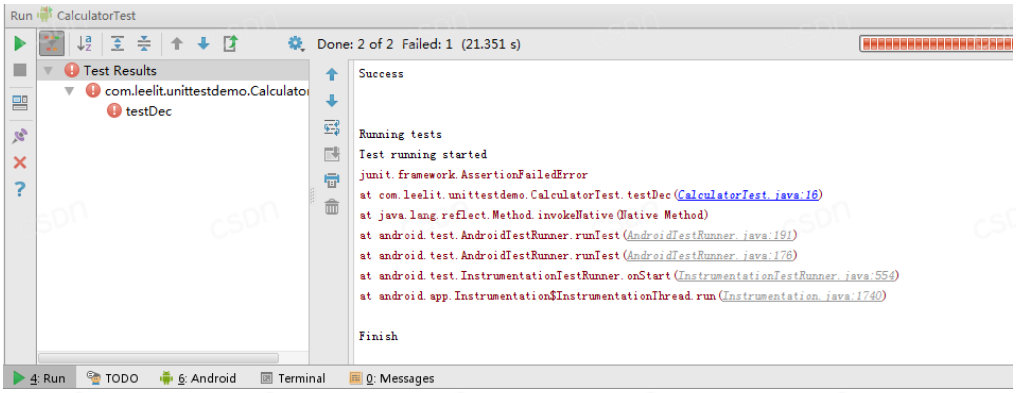
有了上面的认识，我们知道了Android SDK本身就内置了以Junit3为基础的Android测试框架，而Android Studio本身就默认帮我们配置好了测试环境，所以直接就可以上测试了。

1、建立一个UnitTestDemo Module
2、编写一个待测试的类，直接拷贝上面的。
3、自动生成测试类，生成的代码位置androidTest目录下，编辑测试代码。





4、右击CalculatorTest类，Run，启动真机或模拟器后即可看到测试结果了。



解决了Eclipse和AS进行基本单元测试的问题，还有两个问题值得探讨：

1、每次测试都要启动真机或者模拟器，有没办法进行本地测试；

2、难道就不能用Junit4吗？

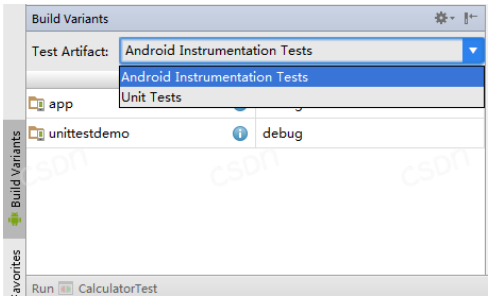
本地测试与Junit4

首先来了解一下两个概念：

> Local tests：代码被编译成运行在本地JVM的单元测试，使用Local tests无需依赖Android框架，但可以通过mock注入依赖。
> Instrumented(Instrumentation) tests：运行在真机或者模拟器的单元测试，这些测试可以获得真机环境，比如context。当难以通过mock注入Android框架的依赖时，就可以使用这种测试。

现在我们知道上面提到的两个例子都是Instrumentation tests。通过继承AndroidTestCase，我们就能分分钟得到context对象啦。不过使用Instrumentation Test每次都要启动一下真机或者模拟器，消耗较多资源，理论上来讲Local Test可以大幅减少测试时间。

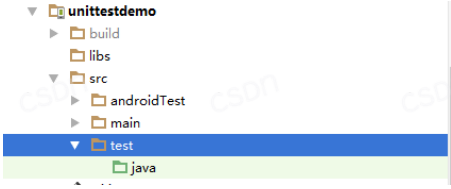
1、配置Local Test环境，选择Unit Tests



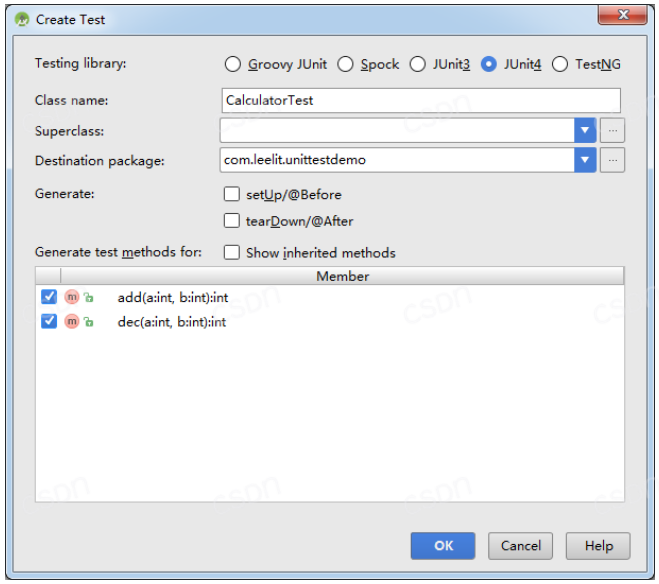
2、导入依赖

```
testCompile 'junit:junit:4.12'
```

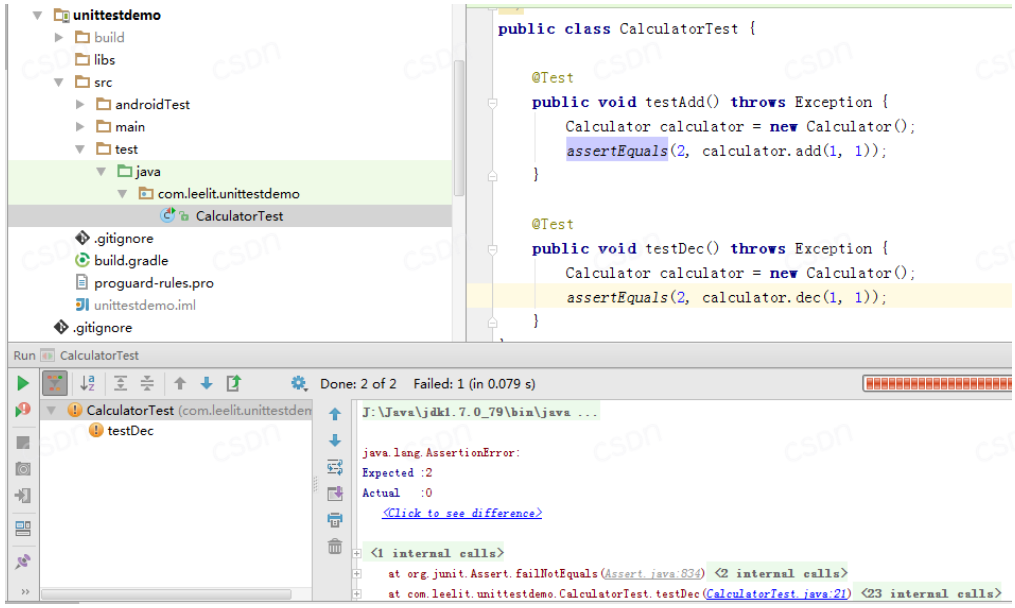
3、新建一个test/java包，必须。



4、和上面一样，自动生成测试类，生成的代码位置test目录下，编辑测试代码。



5、右击CalculatorTest类，Run。这一次无需启动真机或模拟器就可以运行了。



刚才提到本地测试很大原因是为了一个字快，理论上确实应该更快，然而在我的渣机器上还不如Instrumentation Test快…

我们一下子就实现了本地测试并且使用的是Junit4，Junit4是使用注解的测试框架，无需像Junit3一样测试方法必须使用test开头。上面这个例子我们自动生成的测试类同样可以使用Junit3。
如果我们把步骤2中的Junit4依赖去掉，Junit4不能使用这是不言而喻的，原本在Instrumentation Test可以使用的Junit3呢？答案是不能的，因为单元测试需要Runner，翻到最上面Eclipse那个工程可以看到Android默认的Runner是InstrumentationTestRunner，当我们使用Local Test时因为导入了Junit4依赖，AS能找到Junit4的Runner，怎么着也应该能兼容Junit3吧；而如果把Junit4依赖去掉，AS就找不到合适的Runner来Run这个Junit3的Local Test了。
最后Instrumentation Test能使用Junit4吗，答案是可以的，但是配置起来比较麻烦，这里就不多纠结了，有爱可以参考。
https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests.html#build

---

小结

1、Android内置基于Junit3的测试框架，默认Runner是InsrtumentationTestRunner。
2、Instrumentation Test需要启动真机或模拟器，默认可以使用Junit3，可以通过配置使用Junit4。
3、Local Test无需启动真机或模拟器，导入Junit4依赖后，支持Junit4兼容Junit3。Local Test不能依赖于Android框架，如果需要注入Android依赖，可以通过mock技术。

这篇文章未涉及的主题包括：
mock和UI测试。

讲了这么一大堆怎样实现单元测试，那究竟测试代码应该怎样写呢。可以参考一些牛库的测试，比如OkHttp：
https://github.com/square/okhttp/tree/master/okhttp-tests/src/test/java/okhttp3

> 参考：
>
> - https://github.com/googlesamples/android-testing/tree/master/unit
> - https://developer.android.com/training/testing/unit-testing/index.html
> - http://stackoverflow.com/questions/2047261/using-android-test-framework/2055455#2055455