**Android Studio单元测试**

1. **Code Coverage Tool（代码覆盖率工具）**

众所周知，在编写单元测试案例的过程中，有一个重要的指标就是代码覆盖率。编写单元测试并不是为了追求100%覆盖率，但覆盖率在单元测试中仍占据着不可或缺的地位。

经典的Java Coverage Tool : Emma、Eclemma（Eclipse推荐使用）、Cobertura

接下来，我们来看Android Studio支持的Coverage Tool ： jacoco、IntelliJ IDEA。

在Android Stuido新建过工程的同学，应该有注意到，该工程默认会新建androidTest及test的测试包。

在Android Stuido中，在androidTest编写的单元测试，默认使用jacoco插件生成包含代码覆盖率的测试报告；而test包下的单元测试代码，则直接使用Android Studio已有工具IntelliJ IDEA生成覆盖率，也可以通过自定义gradle task使用jacoco插件生成与androidTest相同格式的测试报告。

1. **androidTest与test**

**区别：**androidTest是存放一些与View(UI界面层)相关的单元测试案例的测试代码集，需要在真机或虚拟机上运行。一般使用的框架有：Instrumentation、Espresso，也可以编写自动化测试案例（框架：Uiautomator2.0）；而test包则一般只存放与Model(数据层)相关的单元测试案例，但Android几乎无法实现MV完全解耦，所以目前在test包下可能也会涉及到View的测试。直接在JVM虚拟机上运行即可，速度快。框架：Robolectric+Mockito+其他。

**运行方式：**Android Studio 2.0版本开始，已经能够智能检测当前的测试是androidTest还是test了，低版本的可以在Build Variants中设置Test Artifact或者Add New Configuration 时选择Unit Tests(对应test),Android Instrumentation Tests(对应androidTest)。

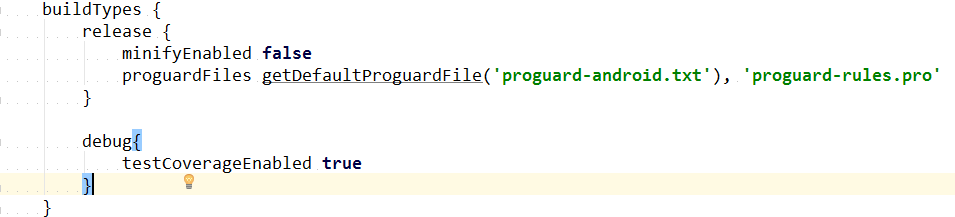
1. **jacoco在androidTest中的使用**

1、指令与配置

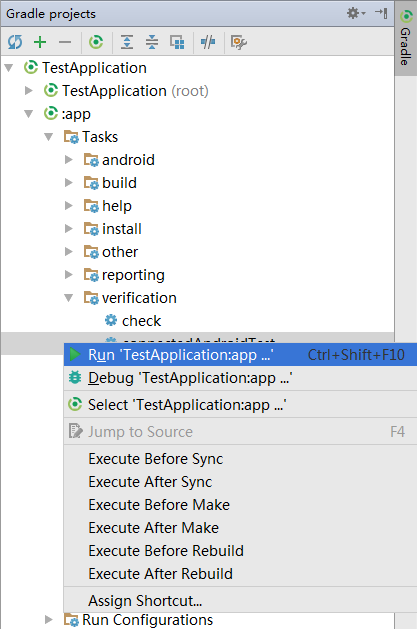
gradlew clean connectedDebugAndroidTest

在module的gradle文件中添加以下代码：





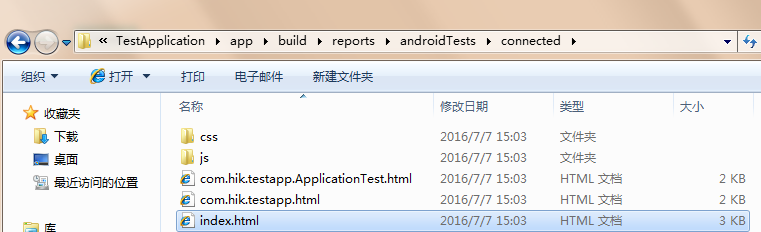
添加完代码之后，打开右侧的Gradle面板运行connectedAndroidTest任务，或在Terminal控制台输入 gradlew clean connectedAndroidTest，回车



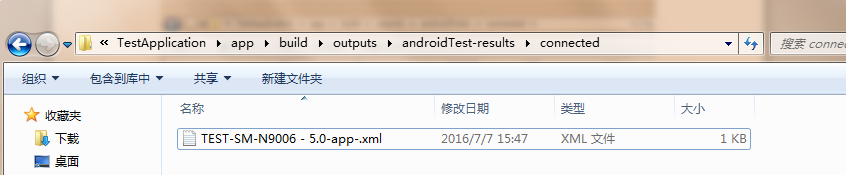


2、单元测试报告

在\app\build\reports\androidTests\connected

中可以查看HTML格式的单元测试报告；

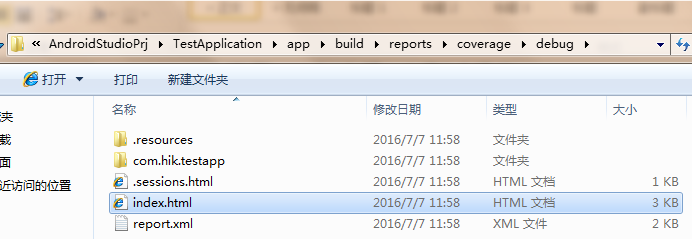
在\app\build\outputs\androidTest-results\connected



可以查看xml格式的单元测试报告。

3、覆盖率报告

在\app\build\reports\coverage\debug



中可以查看html格式的覆盖率报告index.html和xml格式的覆盖率报告report.xml。

1. **jacoco在test中的使用**

1、指令与配置

gradlew clean testDebugUnitTest jacocoTestReport

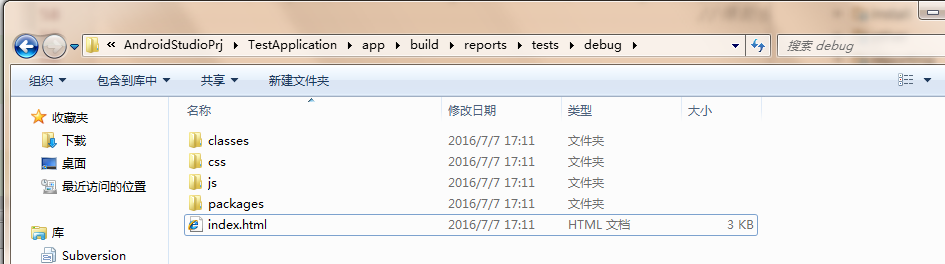
在test包下，如果使用jacoco生成覆盖率将会麻烦一点点

需要在gradle中自定义task，代码如下：



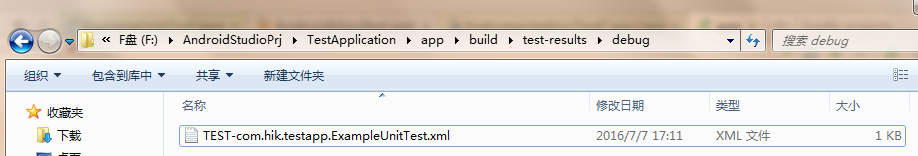
2、单元测试报告

在 \app\build\reports\tests\debug



中可以查看html格式的单元测试报告；

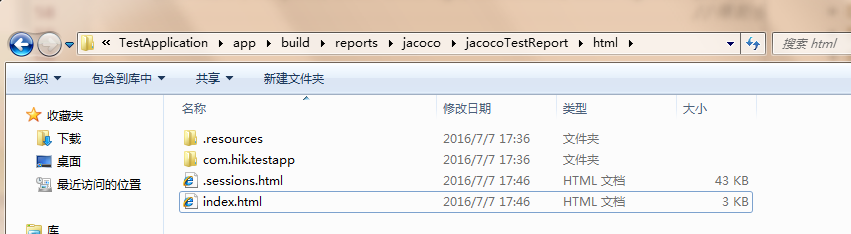
在\app\build\test-results\debug



中可以查看xml格式的单元测试报告。

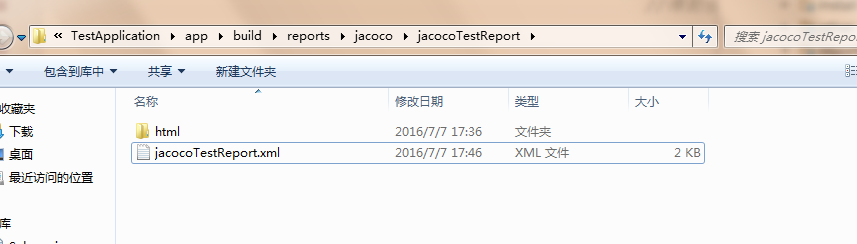
3、覆盖率报告

在\app\build\reports\jacoco\jacocoTestReport\html



中可以查看html格式的覆盖率报告；

在\app\build\reports\jacoco\jacocoTestReport\



中可以查看xml格式的覆盖率报告。

**五、同时运行androidTest和test**

1、指令与配置

同时运行这两个测试只能使用命令行执行，Android Studio中同时只能运行其中一种测试，但是命令行则可以同时运行这两种测试模式。

gradlew clean connectedDebugAndroidTest testDebugUnitTest jacocoTestReport



executionData改变一下。

2、单元测试报告

单元测试结果文件路径不会改变，依然分开存放在build文件夹中

\app\build\outputs\androidTest-results\connected

// instrument test result folder xml格式

\app\build\reports\androidTests\connected

// instrument test result folder html格式

\app\build\test-results\ debug

//unit test result folder xml格式

\app\build\reports\tests\debug

//unit test result folder html格式

3、覆盖率报告

我们使用UnitTest的jacoco插件来合并androidTest的覆盖率结果，然后得到合并后的测试覆盖率。

修改

"jacocoTestReport" gradle task中executionData这个属性

executionData = files('../app/build/jacoco/testDebugUnitTest.exec’, '../app/build/outputs/code-coverage/connected/coverage.ec')

测试结果的存放路径为：

\app\build\reports\jacoco\jacocoTestReport

**六、问题**

1、在使用jacoco进行test的时候出现如下问题：

java.lang.RuntimeException: Method runTest in android.test.InstrumentationTestCase not mocked. See http://g.co/androidstudio/not-mocked for details.

解决方法：在gradle中增加：

android {

// ...

testOptions {

unitTests.returnDefaultValues = true

}

}

2、在AS中的terminal中运行指令出现下载2.10.zip文件失败的问题

解决方法：自己另外下载一个2.10.zip文件，放置在

C:\Users\xieyajuan\.gradle\wrapper\dists\gradle-2.10-all\a4w5fzrkeut1ox71xslb49gst目录下,即可解决问题

3、自动生成的test中ExampleUnitTest类继承InstrumentationTestCase

会出现所有用例即使是错误的也会测试通过问题，修改为继承TestCase即可解决该问题。

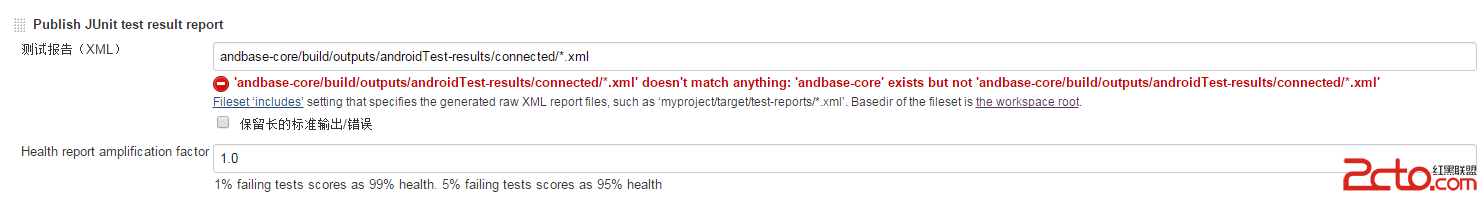
七、Jenkins服务器配置自动化单元测试

1、调用Gradle脚本



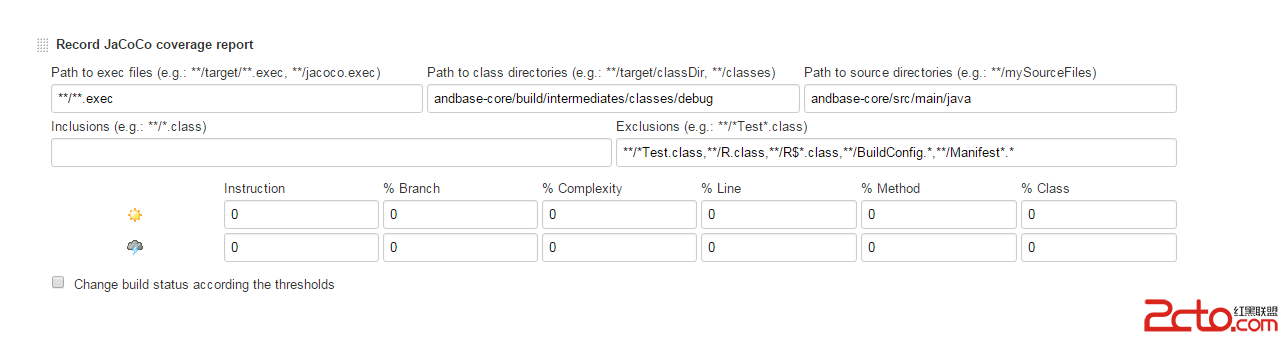
去掉connectedAndroidTest的关联，因为我们已经独立使用命令执行connectedAndroidTest了，所以jenkins服务器在跑job的时候，**请确保已经打开了Android模拟器**，否则会出错。

2、配置Junit单元测试报告



这时候，报了个错误，说当前路径没有匹配到文件，没关系，因为我们还没有执行命令之前，一些文件夹还没有生成，所以可以先忽略。

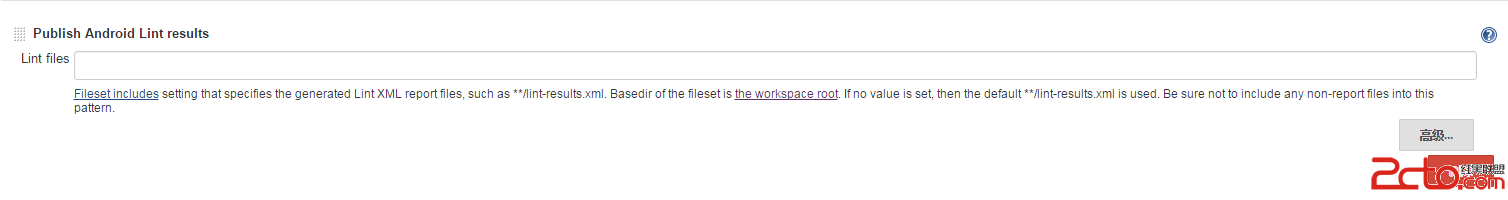
3、配置Jacoco代码覆盖率报告



主要的几个参数配置，“Path to class directories”配置的是编译后.class文件的路径地址，Android都是放在build路径下build\intermediates\classes；“Path to source directories”配置的是java代码路径。

4、配置Android Link报告

Android Lint是Android自带的一个功能，它可以检测一些不规范的写法，并提示。该功能gradle不用配置任何东西，只要执行build之后就会自动生成报告。

上图中不用填写入任何路径，默认的即可。

5、邮件通知

