# Android单元测试

## java单元测试框架

目前主要是用junit + Mockito

## Android单元测试框架

### 2.1 [AndroidJUnitRunner](https://link.jianshu.com/?t=http://developer.android.com/tools/testing-support-library/index.html" \l "AndroidJUnitRunner" \t "http://www.jianshu.com/p/_blank)介绍

一个兼容Junit4的Andriod单元测试框架, 使用跟Junit是一样的，只不过需要运行在android真机或模拟器环境

### 2.2 [Espresso](https://link.jianshu.com/?t=https://google.github.io/android-testing-support-library/docs/espresso/index.html" \t "http://www.jianshu.com/p/_blank)介绍

支持Android UI测试的单元测试框架, 运行在真机或模拟器，android任何代码都能运行。

支持等待异步任务，如：等待网络接口请求回调成功后，再执行测试代码，基于此不但可以做单个界面的单元测试，也可以稍微改良下后，做整个apk的业务流程测试。

### 2.3 Robolectric介绍

运行在jvm上，但是框架本身引入了android依赖库，所以可以做android单元测试，运行速度比运行在真机or模拟器快。但Robolectric也有局限性，例如不支持加载so，但是可以配合junit和mockito使用

### 2.4 UIAutomator介绍

谷歌推出用于UI自动化测试的工具，也就是普通的手工测试，点击每个控件元素看看输出的结果是否符合预期。比如登陆界面分别输入正确和错误的用户名密码然后点击登陆按钮看看是否能否登陆以及是否有错误提示等。

UI Automator测试框架是基于instrumentation的API，运行在Android JunitRunner 之上，同时UI Automator Test只运行在Android 4.3(API level 18)以上版本。

特点：不依赖anadroid源码，

缺点：

1.如果想要使用resource-id定位控件，则需要level 18及以上才可以

2.对于子线程异步任务没办法把控，只能等待指定的一段时间，若在等待时间内，异步任务未返回，导致未更新指定界面，则导致测试用例失败

### 2.5 Monkey介绍

android自带的测试框架，通过向手机发送随机事件（手势，点击，输入）进行应用程序压力测试（可指定应用程序包名，也可以说是稳定性测试），测试应用程序的稳定性和健壮性

### 2.6 MonkeyRunner介绍

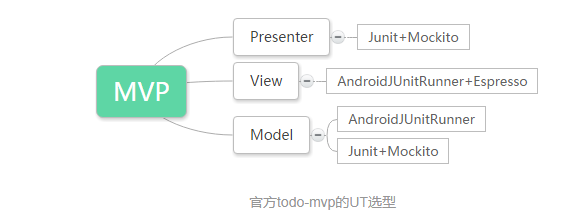
Android自动化测试工具。它提供一个API，运用该API编写的程序可以不用通过Android代码来直接控制Android设备和模拟器。既能按照用户的需求来产生特定序列的特定事件，还能截屏保存和比较图片，编写简单。借助于python，也能实现强大的功能

## Android官方MVP项目单元测试

### 3.1 项目Github地址：

[https://github.com/googlesamples/android-architecture](https://link.jianshu.com/?t=https://github.com/googlesamples/android-architecture" \t "http://www.jianshu.com/p/_blank)

### 3.2 各层的单元测试选型



P层：不需要任何Android环境，因此使用Junit测试即可

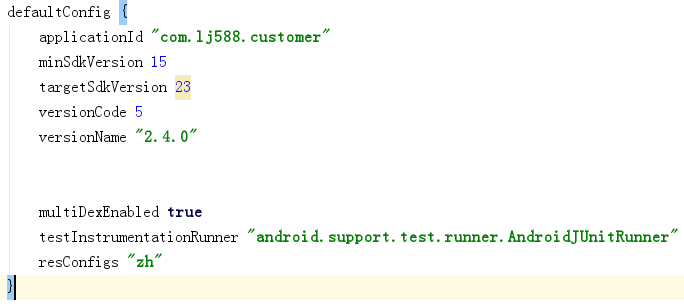
V层：使用Google强大的Espresso进行UI的测试

M层：涉及到数据库相关操作，因此需要依赖Android环境，使用AndroidJUnitRunner进行测试

## Espresso配置

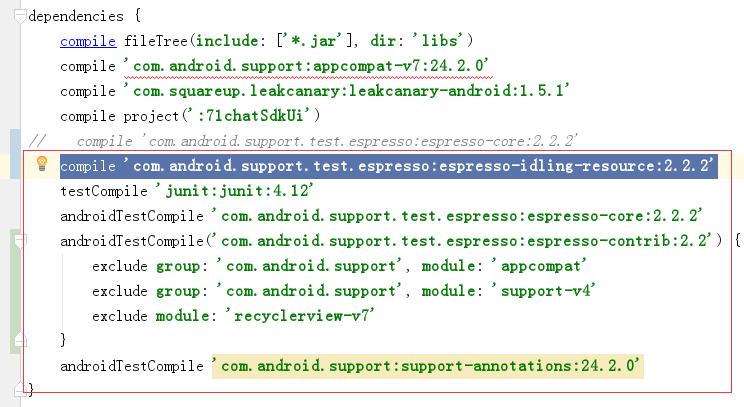
由于当前项目采用MVC模式，Java层和Android库依赖耦合度较高，故采用Espresso单元测试框架。

### 4.1 首先在主工程的build.gradle中加入如下代码



testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

### 4.2 在dependencies中加入测试依赖包:



1.compile **'com.android.support.test.espresso:espresso-idling-resource:2.2.2'**

testCompile **'junit:junit:4.12'**

androidTestCompile **'com.android.support.test.espresso:espresso-core:2.2.2'**

androidTestCompile(**'com.android.support.test.espresso:espresso-contrib:2.2'**) {  
 exclude **group**: **'com.android.support'**, **module**: **'appcompat'** exclude **group**: **'com.android.support'**, **module**: **'support-v4'** exclude **module**: **'recyclerview-v7'**}

androidTestCompile **'com.android.support:support-annotations:24.2.0'**

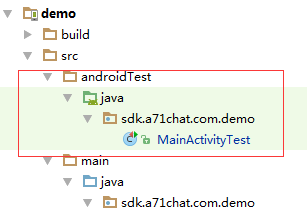
### 4.3注意事项

这里我们引用了一个espresso-idling-resource:2.2.2依赖包，这个包里面就一个接口：IdlingResource， 这个接口是用来实现espresso 测试线程等待我们apk中调用的网络请求的，等网络请求请求到数据后，则 espresso测试线程继续工作，详情见5.12

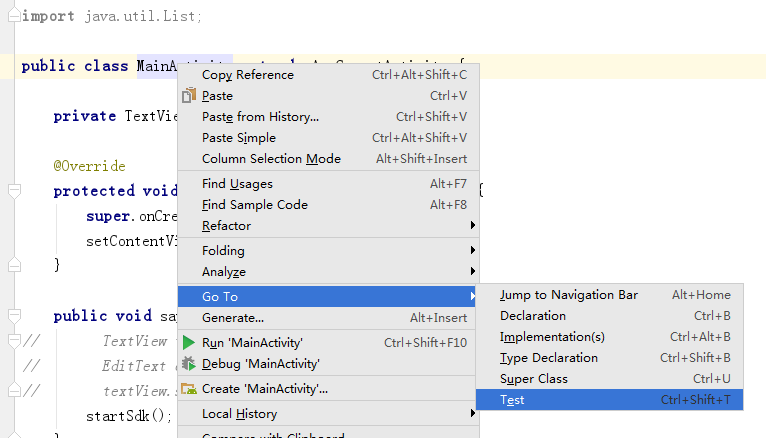
## 5 Espresso 使用说明

### 5.1 首先在主工程下新建androidTest目录

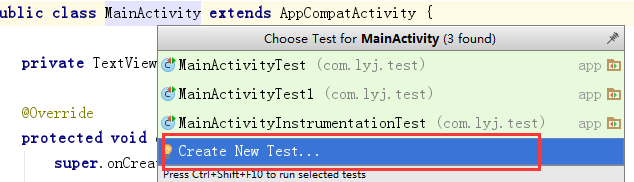
存在此目录则不必创建



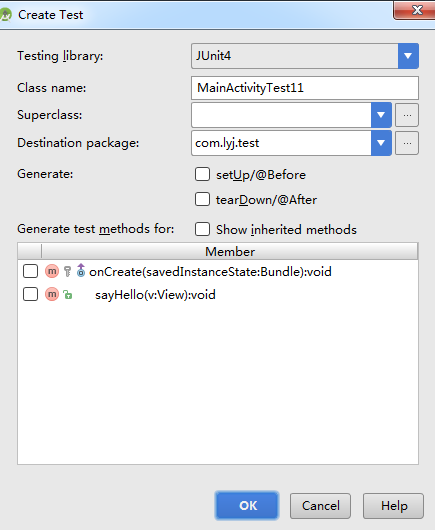
### 5.2 鼠标放在MainActivity上点击右键新建一个测试类，如下：



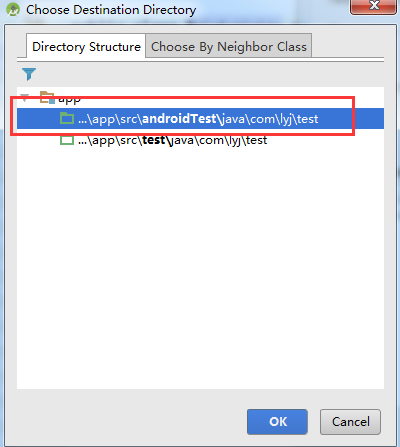
点击Test进入如下界面:



点击Create New Test...



点击OK



选择androidTest目录，在生成的测试类中，我们加入如下代码，如下图所示：



### 5.3 @Rule

代表测试规则

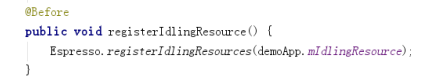


这行代码，可以理解为 需要测试的类 为 MainActivity, 运行这个测试类，将直接打开MainActivity界面。

### 5.4 @Befor

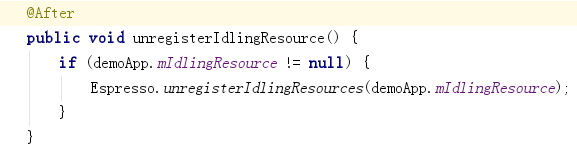
代表正式执行UI测试之前，需要进行的相关初始化工作, 这里我们在

Espresso里面注册IdlingResource，为以后的UI线程同步作准备。



### 5.5 @After

代表UI测试执行完毕之后，需要进行的操作。



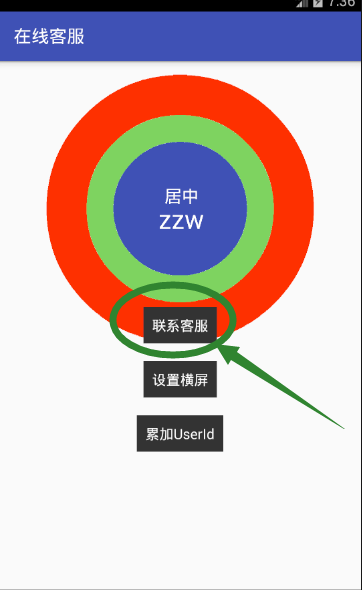
### 5.6 @Test

代表开始测试UI



代码说明:





这行代码表示找到 界面上显示内容为 “联系客服” 的控件，如上图， 并执行click操作，如果界面上没有符合条件的控件，或者符合条件的控件处于隐藏状态，则本行代码将执行错误，测试用例失败。

### 5.7 onView()说明

onView(withClassName()); 根据视图的类名称查找控件

onView(withId()); 根据视图的在xml中的id查找控件

onView(withText()); 根据在视图中显示的文本内容查找控件

### 5.8 Perform(ViewAction) 说明

一些常用的方法如下：

click()：返回一个点击动作，和我们自己手动点击按钮一样。

clearText()：返回一个清除指定view中的文本action，在测试EditText时用的比较多。

swipeLeft()：返回一个从右往左滑动的action,常用于ViewPager

swipeRight()：返回一个从左往右滑动的action,常用于ViewPager

swipeDown()：返回一个从上往下滑动的action。

swipeUp()：返回一个从下往上滑动的action。

closeSoftKeyboard()：返回一个关闭输入键盘的action。

pressBack()：返回一个点击手机上返回键的action。

doubleClick()：返回一个双击action

longClick()：返回一个长按action

### 5.9 校验check说明

doesNotExist：断言某一个view不存在。

matches：断言某个view存在，且符合一列的匹配。

selectedDescendentsMatch：断言指定的子元素存在，且他们的状态符合一些列的匹配。



本行代码就是用来检测控件是否在界面上显示，若控件处于隐藏状态，直接退出测试，且报错

### 5.10 ListView GridView子布局点击

这个需要用到适配器匹配，调用Espresso.onData()方法

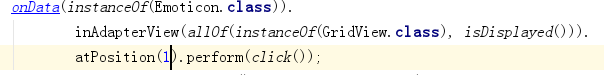


如上图，需要点击第2个GirdView里面的emoji表情：

第一步点击 上图中1操作

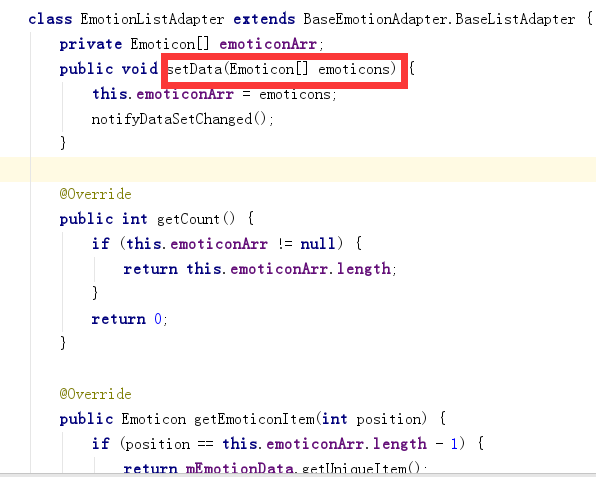


第二步点击 上图中2操作



本行代码解读：

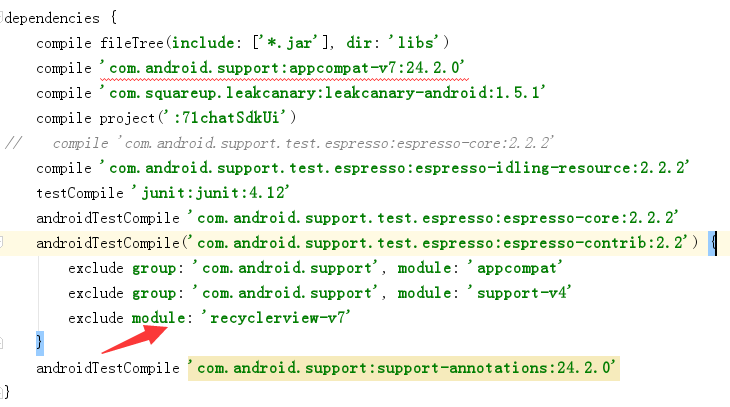
找到界面中类型为GridView的所有显示在界面上的布局，并且匹配 adapter内部数据类型为Emoticon.class的条目，定位到第2个位置，执行点击操作。



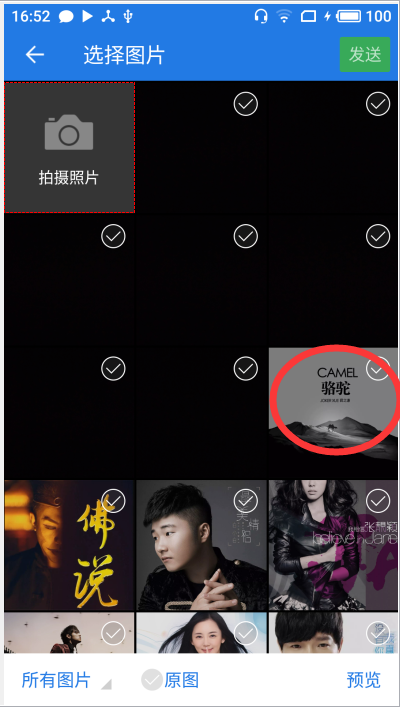
从实际项目中的代码中可以看到 adapter中的数据类型为Emoticon.class

### 5.11 RecyclerView子布局点击

由于RecyclerView不适用adapter类型，故不能用4.3.10的方式进行子布局操作，需要在dependencies中加入 exlude module:’recylerview-v7’



例如 -- 访客端的图库采用了recylerView，如下，



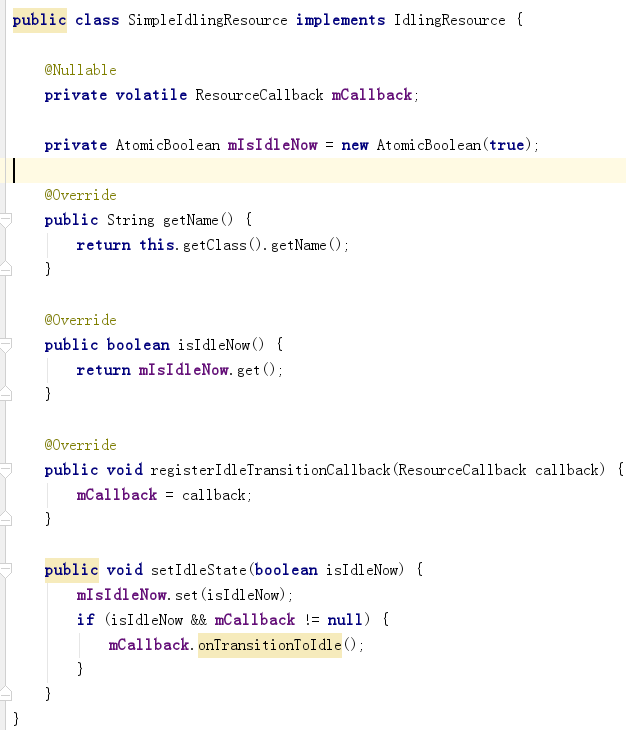


R.id.folder\_list 为recyclerView在xml中的控件id， 这里实现的是点击第9个控件。

### 5.12 IdlingResource UI同步

实际界面中，经常要等待网络回调完成后，才能根据数据显示对应的UI，这时候就需要用到IdlingResource

5.12.1 新建SimpleIdlingResource 继承 IdlingResource 接口

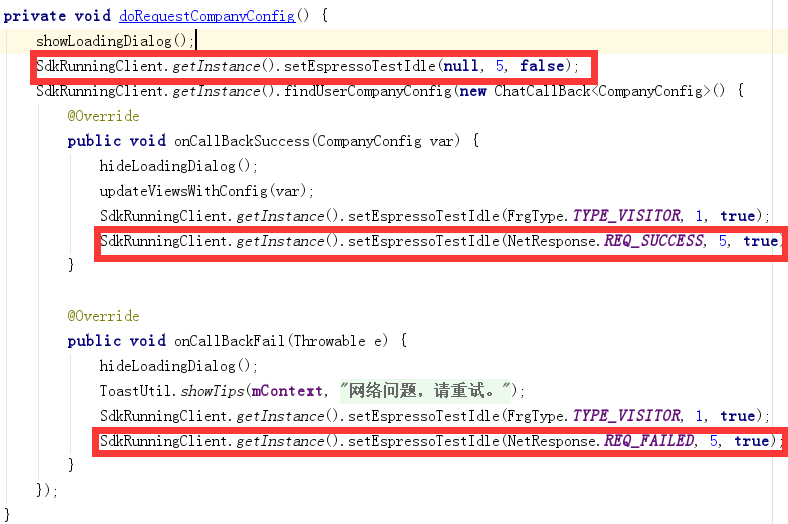


实现一些继承的方法，通过调用ResourceCallback.onTransitionToIdle()通知Espresso。

例如：访客端sdk的选择主题界面：

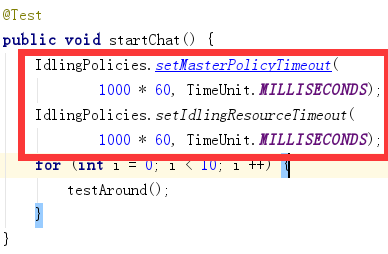


本界面需要等待网络回调数据，来进行UI刷新，如果进入本界面，未等待网络回调而操作UI，此时所有控件处于隐藏状态，espresso进行测试操作，就会失败，由此，我们在调用接口前将 IdlingResource 置为false，网络回调完成，界面渲染完毕后，将IdlingResource 置为true。



SdkRunningClient.getInstance().setEspressoTestIdle 相关调用见5.16

### 5.13 Espresso设置超时时长：



### 5.14 Espresso 判断控件是否显示

由于Espresso只支持检测控件是否存在，不能判断控件是否显示，在我们访客端就有这种情况，控件的显示隐藏可以按照后台设置而改变，做集成业务测试的话，那我们就需要兼容，判断控件显示，才能做相应的操作。

这里，需要自定义ViewAction, 具体代码如下：

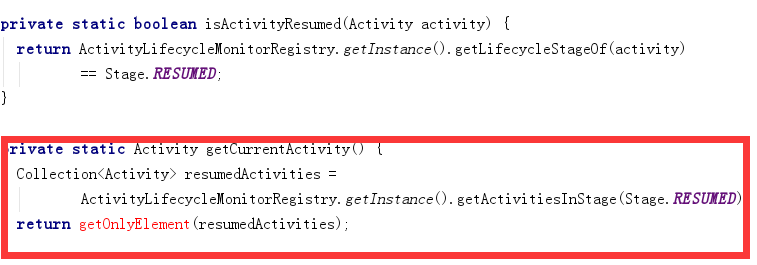




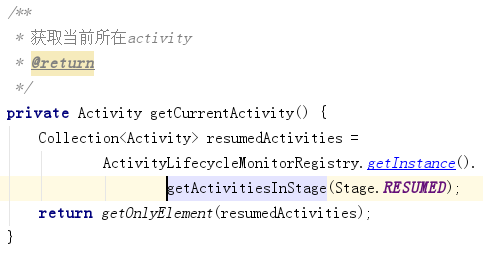


### 5.15 Espresso 检测当前的Activity类型

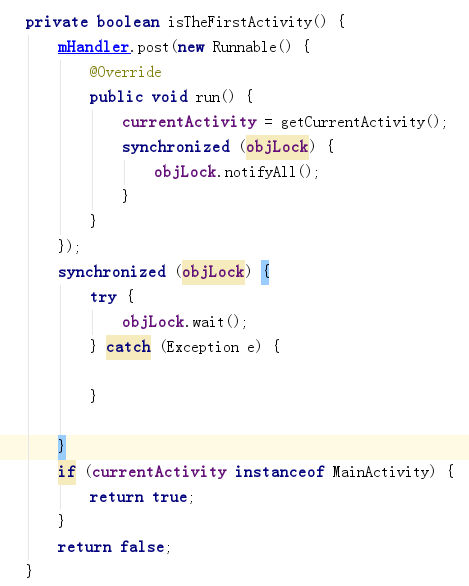
业务场景：有时候我们执行Espresso返回事件时，需要判断到底返回到哪个界面上去了，才能作相应操作，但是Espresso并无此方法开放，通过查看源码，我们可以看到在KeyEventAction类里面存在获取当前Activity的方法，如下



故把此方法提取到MainActivityTest类里面，



但是在执行getCurretActivity()方法时报错，原因Espresso提示为 - 未在主线程中进行操作，故对本方法进行改良，采取Espresso线程等待主线程执行完毕后，再往下执行代码。具体实现如下图：



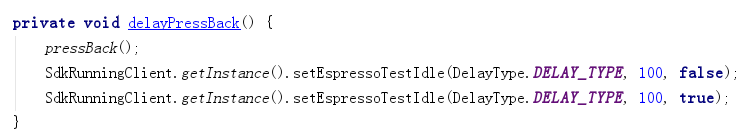
这里isTheFirstActivity()运行在Espresso子线程，mHandler中的Looper为主线程的looper，故getCurrentActivity()这句代码执行在主线程，执行完毕后，唤醒正在等待的Espresso工作线程，继续执行if (currentActivity instanceof MainActivity) 判断代码。

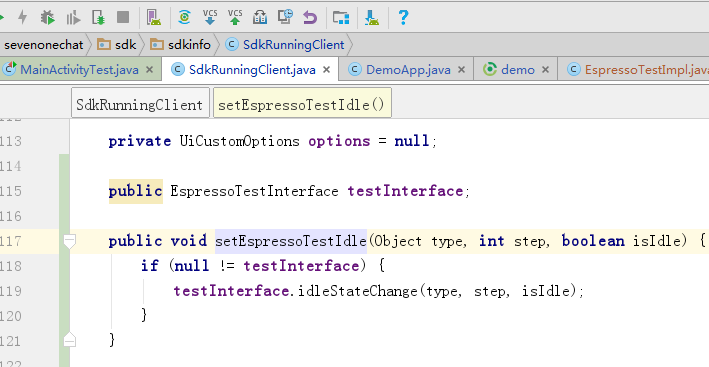
### 5.16 Activity跳转延迟

用Espresso做集成业务测试时，可能会存在，Espresso线程在执行A界面测试代码，但是界面还停留在B界面，这就是由于主线程跳转界面存在一定的延迟，而Espresso子线程的等待时间太短造成，所以我们需要借助IdlingResource 停顿Espresso子线程一段时间

具体代码 参照访客端sdk的实现如下：

改良Espresso的返回事件：



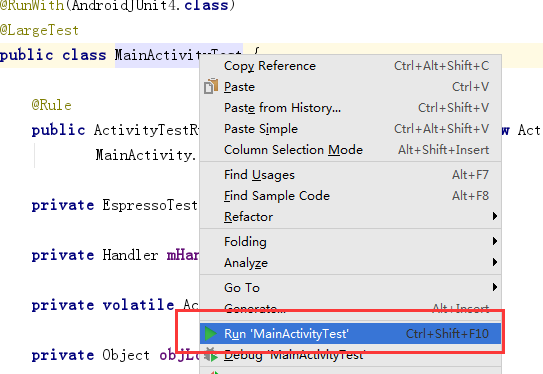




我这里等待了500毫秒，等待Android系统执行跳转完毕，目前来看，已经够了。

### 5.17 运行测试代码

右键点击MainActivityTest，如下图：



注：全文测试代码仅供参考