ANDROID APP测试工具调研

工具一

android SDK里自带的测试工具之一monkey。

1.什么是monkey

Monkey工具直接运行在设备或模拟器的adb shell中，生成用户或系统的伪随机事件流，主要应用在压力和可靠性测试上，运行该命令可以随机地向目标程序发送各种模拟键盘事件流，并且可以自己定义发送的次数，以此观察被测应用程序的稳定性和可靠性。

2．Monkey的特征

（1）、 测试的对象仅为应用程序包，有一定的局限性。

（2）、 Monkey测试使用的事件流数据流是随机的，不能进行自定义。

（3）、 可对Monkey 测试的对象，事件数量，类型，频率等进行设置。

3. Monkey的基本用法

启动指定的应用程序，并向其发送500个伪随机事件

$ adb shell monkey -p your.package.name -v 500

命令选项参考：

下表中列出了Monkey命令行可用的全部选项。

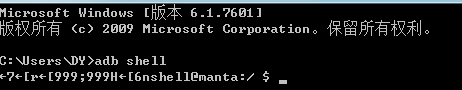
|  |  |  |
| --- | --- | --- |
| **类别** | **选项** | **说明** |
| 常规 | --help | 列出简单的用法。 |
| -v | 命令行的每一个 -v 将增加反馈信息的级别。 Level 0( 缺省值 ) 除启动提示、测试完成和最终结果之外，提供较少信息。 Level 1 提供较为详细的测试信息，如逐个发送到 Activity 的事件。 Level 2 提供更加详细的设置信息，如测试中被选中的或未被选中的 Activity 。 |
| 事件 | -s <seed> | 伪随机数生成器的 seed 值。如果用相同的 seed 值再次运行 Monkey ，它将生成相同的事件序列。 |
| --throttle <milliseconds> | 在事件之间插入固定延迟。通过这个选项可以减缓 Monkey 的执行速度。如果不指定该选项， Monkey 将不会被延迟，事件将尽可能快地被产成。 |
| --pct-touch <percent> | 调整触摸事件的百分比(触摸事件是一个down-up事件，它发生在屏幕上的某单一位置)。 |
| --pct-motion <percent> | 调整动作事件的百分比(动作事件由屏幕上某处的一个down事件、一系列的伪随机事件和一个up事件组成)。 |
| --pct-trackball <percent> | 调整轨迹事件的百分比(轨迹事件由一个或几个随机的移动组成，有时还伴随有点击)。 |
| --pct-nav <percent> | 调整“基本”导航事件的百分比(导航事件由来自方向输入设备的up/down/left/right组成)。 |
| --pct-majornav <percent> | 调整“主要”导航事件的百分比(这些导航事件通常引发图形界面中的动作，如：5-way键盘的中间按键、回退按键、菜单按键) |
| --pct-syskeys <percent> | 调整“系统”按键事件的百分比(这些按键通常被保留，由系统使用，如Home、Back、Start Call、End Call及音量控制键)。 |
| --pct-appswitch <percent> | 调整启动Activity的百分比。在随机间隔里，Monkey将执行一个startActivity()调用，作为最大程度覆盖包中全部Activity的一种方法。 |
| --pct-anyevent <percent> | 调整其它类型事件的百分比。它包罗了所有其它类型的事件，如：按键、其它不常用的设备按钮、等等。 |
| 约束限制 | -p <allowed-package-name> | 如果用此参数指定了一个或几个包，Monkey将只允许系统启动这些包里的Activity。如果你的应用程序还需要访问其它包里的Activity(如选择取一个联系人)，那些包也需要在此同时指定。如果不指定任何包，Monkey将允许系统启动全部包里的Activity。要指定多个包，需要使用多个 -p选项，每个-p选项只能用于一个包。 |
| -c <main-category> | 如果用此参数指定了一个或几个类别，Monkey将只允许系统启动被这些类别中的某个类别列出的Activity。如果不指定任何类别，Monkey将选择下列类别中列出的Activity： Intent.CATEGORY\_LAUNCHER或Intent.CATEGORY\_MONKEY。要指定多个类别，需要使用多个-c选项，每个-c选 项只能用于一个类别。 |
| 调试 | --dbg-no-events | 设置此选项，Monkey将执行初始启动，进入到一个测试Activity，然后不会再进一步生成事件。为了得到最佳结果，把它与-v、一个或几个包约束、以及一个保持Monkey运行30秒或更长时间的非零值联合起来，从而提供一个环境，可以监视应用程序所调用的包之间的转换。 |
| --hprof | 设置此选项，将在Monkey事件序列之前和之后立即生成profiling报告。这将会在data/misc中生成大文件(~5Mb)，所以要小心使用它。 |
| --ignore-crashes | 通常，当应用程序崩溃或发生任何失控异常时，Monkey将停止运行。如果设置此选项，Monkey将继续向系统发送事件，直到计数完成。 |
| --ignore-timeouts | 通常，当应用程序发生任何超时错误(如“Application Not Responding”对话框)时，Monkey将停止运行。如果设置此选项，Monkey将继续向系统发送事件，直到计数完成。 |
| --ignore-security-exceptions | 通常，当应用程序发生许可错误(如启动一个需要某些许可的Activity)时，Monkey将停止运行。如果设置了此选项，Monkey将继续向系统发送事件，直到计数完成。 |
| --kill-process-after-error | 通常，当Monkey由于一个错误而停止时，出错的应用程序将继续处于运行状态。当设置了此选项时，将会通知系统停止发生错误的进程。注意，正常的(成功的)结束，并没有停止启动的进程，设备只是在结束事件之后，简单地保持在最后的状态。 |
| --monitor-native-crashes | 监视并报告Android系统中本地代码的崩溃事件。如果设置了--kill-process-after-error，系统将停止运行。 |
| --wait-dbg | 停止执行中的Monkey，直到有调试器和它相连接。 |

4. Monkey测试的一个实例

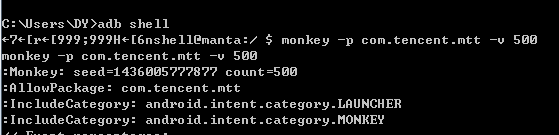
在命令行中输入：adb devices查看设备连接情况



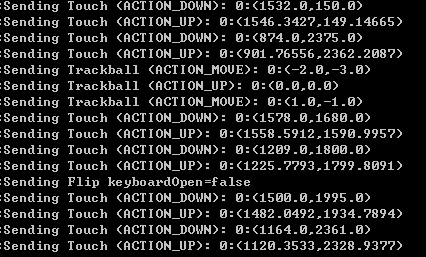
在有设备连接的前提下，在命令行中输入：adb shell 进入shell界面



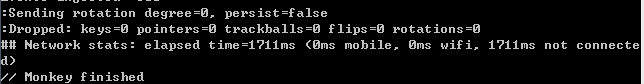
以com.tencent.mtt作为对象进行MonkeyTest（其中-p表示对象包 –v 表示反馈信息级别）



下面开始发送随机事件



应该是日志以及测试完成



结果分析

Monkey主要是随机地向目标程序发送各种模拟键盘事件流，因为主要是一系列的屏幕点击时间，可以呈现目标程序在经过随机点击事件目标程序的稳定性，在经过反复的monkey测试下的，目标程序没有发现异常情况，我们应该可以认为目标测试程序是相对稳定的。个人认为monkey主要是向目标程序发送各种模拟键盘事件流，并不能实现认为想要实现的一些事件，所以可以说monkey应该达不到自动化测试工具的标准。

5.关于Monkey测试的停止条件

Monkey Test执行过程中在下列三种情况下会自动停止：

（1）、如果限定了Monkey运行在一个或几个特定的包上，那么它会监测试图转到其它包的操作，并对其进行阻止。

（2）、如果应用程序崩溃或接收到任何失控异常，Monkey将停止并报错。

（3）、如果应用程序产生了应用程序不响应(application not responding)的错误，Monkey将会停止并报错。

通过多次并且不同设定下的Monkey测试才算它是一个稳定性足够的程序。

工具二

android SDK里自带的测试工具之二monkeyruuner

1.是monkeyrunner

monkeyrunner工具提供了一个API，使用此API写出的程序可以在Android代码之外控制Android设备和模拟器。通过monkeyrunner，您可以写出一个Python程序去安装一个Android应用程序或测试包，运行它，向它发送模拟击键，截取它的用户界面图片，并将截图存储于工作站上。monkeyrunner工具的主要设计目的是用于测试功能/框架水平上的应用程序和设备，或用于运行单元测试套件，当然也可以将其用于其它目的。

2. monkeyrunner工具同Monkey工具的差别

Monkey：

Monkey工具直接运行在设备或模拟器的adb shell中，生成用户或系统的伪随机事件流。

monkeyrunner：

monkeyrunner工具则是在工作站上通过API定义的特定命令和事件控制设备或模拟器。

3. monkeyrunner的测试类型

（1）、多设备控制：monkeyrunner API可以跨多个设备或模拟器实施测试套件。您可以在同一时间接上所有的设备或一次启动全部模拟器（或统统一起），依据程序依次连接到每一个，然后运行一个或多个测试。您也可以用程序启动一个配置好的模拟器，运行一个或多个测试，然后关闭模拟器。

（2）、 功能测试： monkeyrunner可以为一个应用自动贯彻一次功能测试。您提供按键或触摸事件的输入数值，然后观察输出结果的截屏。

（2）、 回归测试：monkeyrunner可以运行某个应用，并将其结果截屏与既定已知正确的结果截屏相比较，以此测试应用的稳定性。

（4）、 可扩展的自动化：由于monkeyrunner是一个API工具包，您可以基于Python模块和程序开发一整套系统，以此来控制Android设备。除了使用monkeyrunner API之外，您还可以使用标准的Python os和subprocess模块来调用Android Debug Bridge这样的Android工具。

device.shell("input text goup01")  
按下HOME键     device.press('KEYCODE\_HOME','DOWN\_AND\_UP')  
按下BACK键     device.press('KEYCODE\_BACK','DOWN\_AND\_UP')  
按下下导航键     device.press('KEYCODE\_DPAD\_DOWN','DOWN\_AND\_UP')  
按下上导航键     device.press('KEYCODE\_DPAD\_UP','DOWN\_AND\_UP')  
按下OK键       
device.press('KEYCODE\_DPAD\_CENTER','DOWN\_AND\_UP')  
          
相应的按键对应的名称如下：  
home键：KEYCODE\_HOME  
back键：KEYCODE\_BACK  
send键：KEYCODE\_CALL  
end键：KEYCODE\_ENDCALL  
上导航键：KEYCODE\_DPAD\_UP  
下导航键：KEYCODE\_DPAD\_DOWN  
左导航：KEYCODE\_DPAD\_LEFT  
右导航键：KEYCODE\_DPAD\_RIGHT  
ok键：KEYCODE\_DPAD\_CENTER  
上音量键：KEYCODE\_VOLUME\_UP  
下音量键：KEYCODE\_VOLUME\_DOWN  
power键：KEYCODE\_POWER  
camera键：KEYCODE\_CAMERA  
menu键：KEYCODE\_MENU

4. 运行monkeyrunner

进入shell命令交互模式后，首要一件事就是导入monkeyrunner所要使用的模块。直接在shell命令下输入：from com.android.monkeyrunner import MonkeyRunner,MonkeyDevice 回车。

截止到这步完成，我们就可以利用monkeyrunner进行测试工作了。这里有两种方案，一是直接在shell命令下输入以下命令，另一种是将以下命令写到python文件里(要懂点python脚本)。



命令说明：

这里摘录部分关键命令 详情请参考上述网站或下面的官方网站

http://developer.android.com/reference/android/view/KeyEvent.html

device=MonkeyRunner.waitForConnection() #连接手机设备



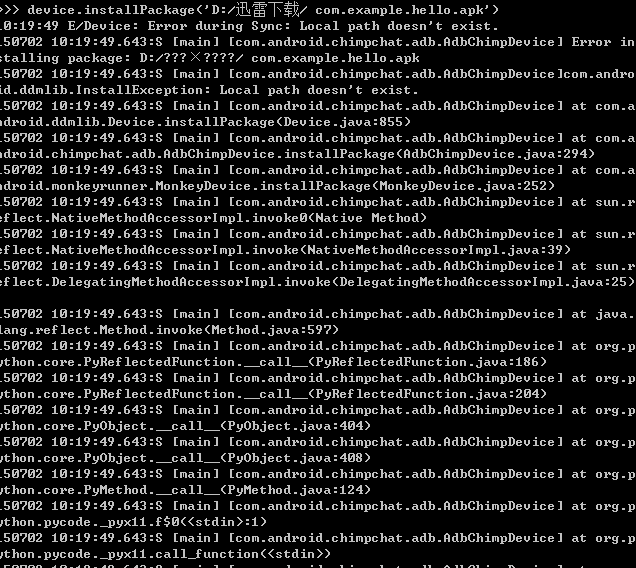
#安装apk包到手机设备(apk文件我放在tools文件夹下)。

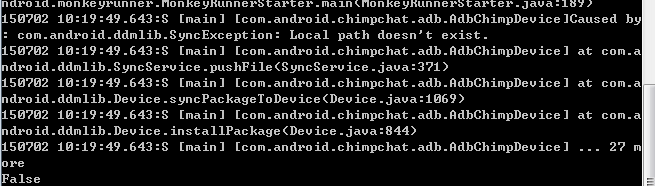
device.installPackage("./dome.apk")

安装成功返回true



安装失败返回false(大多数是安装路径有问题)





#启动其中的任意activity了，只要传入package和activity名称即可。

device.startActivity(component="com.dome/com.dome.MainActivity") #打开这个应用程序的某个Activity。

成功（没有报错）



device.reboot() #手机设备重启



# 截取屏幕截图 (尝试没有成功)

result = device.takeSnapShot

# 将截图保存至文件(尝试没有成功)

result.writeToFile('./shot1.png','png')

更多命令详见

http://developer.android.com/guide/developing/tools/monkeyrunner\_concepts.html

python脚本

就是把系列指令写在x.py 文件里，运行cmd 执行如： monkeyrunner D:\test.py

指令，D:\ 是存放 x.py文件的路径。

测试结果分析

MonkeyRunner提供的操作有限，适合重复性、短路径的测试，不适合连续性的操作，由于type和drag方法的不稳定，很容易偏离我们的预期。MonkeyRunner操作依赖于控件坐标（也可以通过id，比较耗时），一旦UI发生改变，脚本无法使用。

Monkeyrunner的等待机制MonkeyRunner.sleep,无法更加友好地等待；MonkeyRunner不提供结果，我们只能通过截图比较，无法摆脱人工干预。

5.monkeyrunner的缺陷：

•不支持中文字符的输入（只能输入ASC码中存在的字符）

•基于UI操作，一旦遇到复杂的应用，测试代码的复杂度指数增长；测试设备性能不同，需要根据设备的快慢对操作加时延；脚本的运行速度慢

•无法将测试结果抓取出来，只能采用截图比较。无法准确统计测试数据。

•drag方法只能实现简单的滑动屏幕的操作，不能进行复杂的拖拽。微博客户端的下拉刷新操作无法模拟出来。

工具三  Robotium

Robotium是一个测试框架，能够方便你为Android应用程序编写强大、健壮的自动化黑盒测试用例。利用Robotium的支持，用例开发人员能够编写功能、系统和验收测试方案，跨越多个 Android activities。Robotium支持Activities、Dialogs、Toasts、Menus和Context Menus。

Robotium自动化测试方法能够模仿普通用户行为，可以试着把一些原来由测试工程师做的测试变成Robotium自动化实现。

 参考文档：http://www.robotium.cn/

Robotium 提供下列好处

1.以最小的应用程序知识，开发功能强大的测试案例。

2.框架支持多个activities 自动活动。

3.最短的时间需求写出测试用例。

4.测试案例的可读性比标准的仪器测试大大提高。

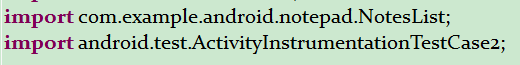
5.通过运行时绑定GUI组件使测试用例更强大。

6.执行测试用例速度快。

7.顺利整合了Maven或Ant来运行测试，持续集成的一部分。

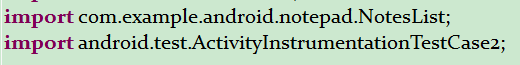
情况一，有源代码

编写Robotium测试程序  
1）导包

//导入需要测试的工程  


//robotium提供的测试用类  


//测试工程要继承用来测试activity的父类

2）泛型写需要测试的工程的入口activity名NotesList。

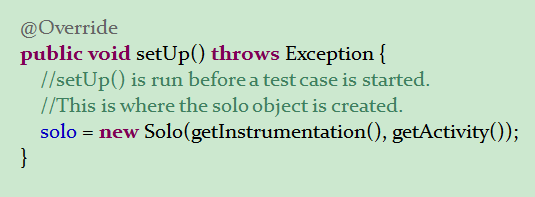
|  |
| --- |
|  |

3）修改构造方法

|  |
| --- |
|  |

4) 在测试方法前覆写父类的setUp()方法：

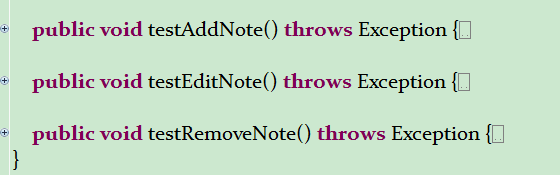
该方法用来初始化solo，绑定对应的Activity。



5）在测试方法后覆写父类的tearDown()方法：

该方法用来清理资源垃圾，关闭activity。

|  |
| --- |
|  |

6）具体操作事件  


7）Solo类运用

Solo类中提供了自动点击、取得、拖拽、搜索等各种方法。

声明Solo类型的成员变量private Solo solo;

**典型方法：**

①  点击：

[clickOnButton(int)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnButton(java.lang.String))—Clicks on a Button with a given index.

[clickOnButton(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnButton(java.lang.String))—Clicks on a Button with a given text.

[clickOnCheckBox(int)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnCheckBox(int))—Clicks on a CheckBox with a given index.

[clickOnView(View)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnView(android.view.View))—Clicks on a given View.

[clickOnText(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnText(java.lang.String))—Clicks on a View displaying a given text.

[clickLongOnText(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickLongOnText(java.lang.String))—Long clicks on a given View.

[clickOnRadioButton(int)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnRadioButton(int))—Clicks on a RadioButton with a given index.

[clickOnScreen(float, float)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#clickOnScreen(float, float))—Clicks on a given coordinate on the screen.

②  取得：

[getCurrentActivity()](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#getCurrentActivity())—Returns the current Activity.

[getText(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#getText(java.lang.String))—Returns a TextView which shows a given text.

[getView(int)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#getView(int))—Returns a View with a given id.

[getEditText(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#getEditText(java.lang.String))—Returns an EditText which shows a given text.

[getImage(int)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#getImage(int))—Returns an ImageView with a given index.

③  拖拽：

[drag(float, float, float, float, int)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#drag(float, float, float, float, int))—Simulate touching a given location and dragging it to a new location.

④  搜索：

[searchText(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#searchText(java.lang.String))—Searches for a text string and returns true if at least one item is found with the expected text.

[searchEditText(String)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#searchEditText(java.lang.String))—Searches for a text string in the EditText objects located in the current Activity.

[searchButton(String, boolean)](http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/com/jayway/android/robotium/solo/Solo.html#searchButton(java.lang.String, boolean))—Searches for a Button with the given text string and returns true if at least one Button is found.

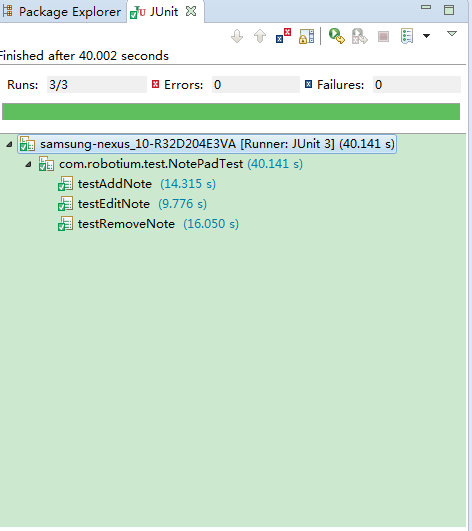
<http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar>

 官方down栏下下载所需要的jar包  
地址：<http://code.google.com/p/robotium/downloads/list>

更多方法请参见Solo的API文档（国内无法访问）：

<http://www.jarvana.com/jarvana/view/com/jayway/android/robotium/robotium-solo/2.0.1/robotium-solo-2.0.1-javadoc.jar!/index-all.html>

成功



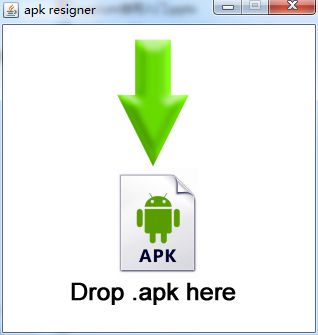
情况二 无源代码，有APK文件

 Android的apk发布，需要签名。具体信息，详见：http://developer.android.com/tools/publishing/app-signing.html

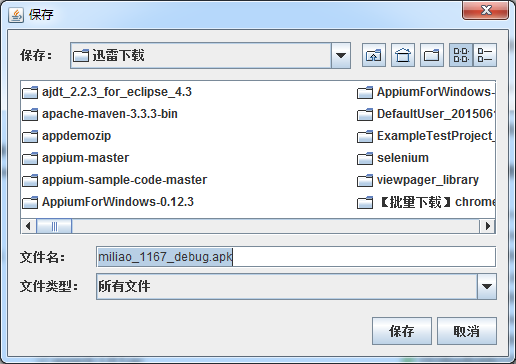
　　为了顺利在测试设备上安装apk，我们需要对拿到的apk进行重新签名（Re-sign）。

　　重新签名的工具，网上很多，我使用的是re-sign.jar。

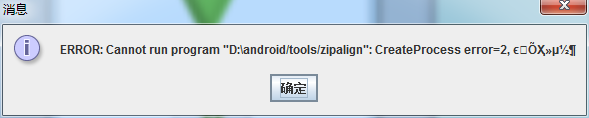
直接打开re-sign.jar，弹出窗体



将想要重新签名的APK文件拖入该窗体，弹出对话框，选择重签的APK保存的位置



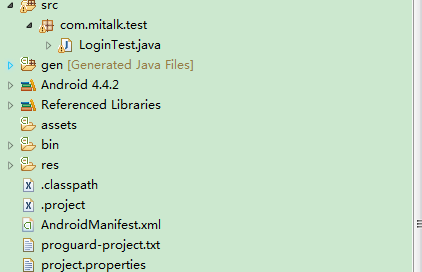
点击保存，结果失败



失败原因很明显，缺少一个zipalign.exe，到网上下载一个，重新运行，重签成功



和有源代码一样编写android测试项目



运行实现自动化的操作

结果分析

Robotium一种可以基于源代码的测试，也可以基于APK的测试，可以对app各种操作的方法进行测试，更加符合实际的测试过程，主要是对Solo类的各种方法熟练应用。

其他

网上还提供很多云测试，Testin云测试，百度云众测等

