android自动化测试框架：CTS、monkey、monkeyrunner、benchmark

monkeyrunner

monkeyrunner工具提供了一个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，我们可以写一个python程序对android应用程序或测试包进行安装、运行、发送模拟击键，对用户界面进行截图并将截图存储在workstation上等操作。monkeyrunner工具的主要设计目的是用于测试application/framework层上的应用程序和设备、或用于运行单元测试套件，也可以用于其它目的。

monkey工具，是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为android测试提供了以下独特的功能：

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。可以在同一时间接上所有设备或一次启动全部模拟器，依据程序依次连接到每一个，然后运行一个或多个测试。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。

4、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，我们可以开发基于python模块和程式的一整套系统，以此来控制android设备。除了使用monkeyrunner API，我们还可以使用标准的python os和ubprocess模块来调用android debug bridge这样的android工具。如ADB这样的android工具，也可以将自己写的类添加到monkeyrunner API中。

运行monkeyrunner

可以直接使用一个代码文件运行monkeyrunner，抑或在交互式对话中输入monkeyrunner语句。不论使用哪种方式，你都需要调用SDK目录的tools子目录下的monkeyrunner命令。如果提供一个文件名作为运行参数，则monkeyrunner将视文件内容为python程序，并加以运行；否则，它将提供一个交互对话环境。

monkeyrunner命令语法

monkeyrunner -plugin <plugin\_jar> <programe\_filename> <programe\_option>

monkeyrunner API

主要包括三个模块

1、MonkeyRunner:这个类提供了用于连接monkeyrunner和设备或模拟器的方法，它还提供了用于创建用户界面显示提供了方法。

2、MonkeyDevice:代表一个设备或模拟器。这个类为安装和卸载包、开启Activity、发送按键和触摸事件、运行测试包等提供了方法。

3、MonkeyImage:这个类提供了捕捉屏幕的方法。这个类为截图、将位图转换成各种格式、对比两个MonkeyImage对象、将image保存到文件等提供了方法。

注意：在运行monkeyrunner之前必须先运行相应的模拟器，否则monkeyrunner无法连接到设备

运行模拟器有两种方法：1、通过eclipse中执行模拟器 2、在CMD中通过命令调用模拟器

这里介绍通过命令，在CMD中执行模拟器的方法

命令：emulator -avd test

上面命令中test是指模拟器的名称。

导入需要的模块

import sys

from com.android.monkeyrunner import MonkeyRunner as mr

from com.android.monkeyrunner import MonkeyDevice as md

from com.android.monkeyrunner import MonkeyImage as mi

如果给导入的模块起了别名，就应该使用别名，而不能使用原名，否则会出现错误。

比如连接设备或模拟器，起了以上别名后，命令应该如下：

device=mr.waitForConnection()

也可以采用以下方式

from com.android.monkeyrunner import MonkeyRunner,MonkeyDevice,MonkeyImage

也可以采用这种方式

import com.android.monkeyrunner

但是在使用时，就显得特别麻烦

device=com.android.monkeyrunner.MonkeyRunner.waitForConnection()

我们也可以给它一个别名

import com.android.monkeyrunner as cam

但是在使用时，就显得特别麻烦

device=cam.MonkeyRunner.waitForConnection()

#等待连接到设备，与模拟器连接，返回monkeydevice对象,代表连接的设备。没有报错的话说明连接成功。

参数1：超时时间，单位秒，浮点数。默认是无限期地等待。

参数2：串deviceid，指定的设备名称。默认为当前设备（手机优先，比如手机通过USB线连接到PC、其次为模拟器）。

默认连接：device=MonkeyRunner.waitForConnection()

参数连接：device = mr.waitForConnection(1.0,'emulator-5554')

向设备或模拟器安装要测试的APK

device.installPackage('myproject/bin/MyApplication.apk') #参数是相对或绝对APK路径

路径级别用“/”，不能用“\”，比如d:\www\a.apk，而应该写成d:/www/a.apk

安装成功返回true,此时查看模拟器我们可以在IDLE界面上看到安装的APK的图标了。

从设备中删除指定的软件包，包括其相关的数据和调整缓存

device.removePackage('myproject/bin/MyApplication.apk')

删除成功返回true。

#启动任意的Activity

device.startActivity(component="your.www.com/your.www.com.TestActivity")

或者

device.startActivity(component="your.www.com/.TestActivity")

此时可以向模拟器发送如按键、滚动、截图、存储等操作了。

执行一个adb shell命令，并返回结果，如果有的话

device.shell("...")

暂停目前正在运行的程序指定的秒数

MonkeyRunner.sleep(秒数，浮点数)

获取设备的屏蔽缓冲区，产生了整个显示器的屏蔽捕获。（截图）

result=device.takeSnapshot()

返回一个MonkeyImage对象（点阵图包装），我们可以用以下命令将图保存到文件

result.writeToFile('takeSnapshot\\result1.png','png')

写文件MonkeyImage

MonkeyImage.writeToFile(参数1:输出文件名，也可以包括路径,参数2:目标格式)

写成功返回true，否则返回false

键盘上的类型指定的字符串，这相当于要求每个字符串中的字符按（键码，DOWN\_AND\_UP）.

字符串发送到键盘

device.type('字符串')

唤醒设备屏幕（在设备屏幕上唤醒）

device.wake()

重新引导到指定的引导程序指定的设备

device.reboot()

=========================================================

在指定位置发送触摸事件（x,y的单位为像素）

device.touch(x,y,TouchPressType-触摸事件类型)

发送到指定键的一个关键事件

device.press(参数1:键码,参数2:触摸事件类型)

参数1：见android.view.KeyEvent

参数2，如有TouchPressType()返回的类型－触摸事件类型，有三种。

1、DOWN 发送一个DOWN事件。指定DOWN事件类型发送到设备，对应的按一个键或触摸屏幕上。

2、UP 发送一个UP事件。指定UP事件类型发送到设备，对应释放一个键或从屏幕上抬起。

3、DOWN\_AND\_UP 发送一个DOWN事件，然后一个UP事件。对应于输入键或点击屏幕。

以上三种事件做为press()或touch()参数。原英文如下：

use this with the type argument of press() or touch() to send a down event.

为了模拟输入键，发送DOWN\_AND\_UP。

参数1的部分具体内容逻辑：

按下HOME键 device.press('KEYCODE\_HOME',MonkeyDevice.DOWN\_AND\_UP)

按下BACK键 device.press('KEYCODE\_BACK',MonkeyDevice.DOWN\_AND\_UP)

按下下导航键 device.press('KEYCODE\_DPAD\_DOWN',MonkeyDevice.DOWN\_AND\_UP)

按下上导航键 device.press('KEYCODE\_DPAD\_UP',MonkeyDevice.DOWN\_AND\_UP)

按下OK键 device.press('KEYCODE\_DPAD\_CENTER',MonkeyDevice.DOWN\_AND\_UP)

device.press('KEYCODE\_ENTER',MonkeyDevice.DOWN\_AND\_UP)#输入回车

device.press('KEYCODE\_BACK',MonkeyDevice.DOWN\_AND\_UP)#点击返回

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

一、什么是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，但您当然也可以将其用于其它目的。

二、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。

三、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。

3、 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。

四、运行monkeyrunner

您可以直接使用一个代码文件运行monkeyrunner，抑或在交互式对话中输入monkeyrunner语句。不论使用哪种方式，您都需要调用SDK目录的tools子目录下的monkeyrunner命令。如果您提供一个文件名作为运行参数，则monkeyrunner将视文件内容为Python程序，并加以运行；否则，它将提供一个交互对话环境。

monkeyrunner的命令语法为：

monkeyrunner -plugin <plugin\_jar> <program\_filename> <program\_options>

五、实例

以sample中的ApiDemos为例，先将其生成ApiDemos.apk。

前提：已有device连接

1、 将ApiDemos.apk放在$Android\_Root\tools下。

2、 在$Android\_Root\tools下新建一个monkeyrunnerprogram.py文件，里面内容为：

# Imports the monkeyrunner modules used by this program

from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice, MonkeyImage

# Connects to the current device, returning a MonkeyDevice object

device = MonkeyRunner.waitForConnection()

# Installs the Android package. Notice that this method returns a boolean, so you can test

# to see if the installation worked.

device.installPackage('./ApiDemos.apk')

# Runs the component

device.startActivity(component='com.example.android.apis/.ApiDemos')

# Presses the Menu button

device.press('KEYCODE\_MENU','DOWN\_AND\_UP')

# Takes a screenshot

result = device.takeSnapshot()

# Writes the screenshot to a file

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

六、实例扩展

因为ApiDemos首页上按下MENU键没有菜单出现，为了更加形象化，在实例五的基础上继续试验：

1、 在$Android\_Root\tools下新建一个monkeyrunnerprogram1.py文件，里面内容为：

# Imports the monkeyrunner modules used by this program

from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice, MonkeyImage

# Connects to the current device, returning a MonkeyDevice object

device = MonkeyRunner.waitForConnection()

# Takes a screenshot

result = device.takeSnapshot()

# Writes the screenshot to a file

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

# Presses the Down button

device.press('KEYCODE\_DPAD\_DOWN','DOWN\_AND\_UP')

device.press('KEYCODE\_DPAD\_DOWN','DOWN\_AND\_UP')

device.press('KEYCODE\_DPAD\_DOWN','DOWN\_AND\_UP')

device.press('KEYCODE\_DPAD\_DOWN','DOWN\_AND\_UP')

device.press('KEYCODE\_DPAD\_DOWN','DOWN\_AND\_UP')

# Takes a screenshot

result = device.takeSnapshot()

# Writes the screenshot to a file

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

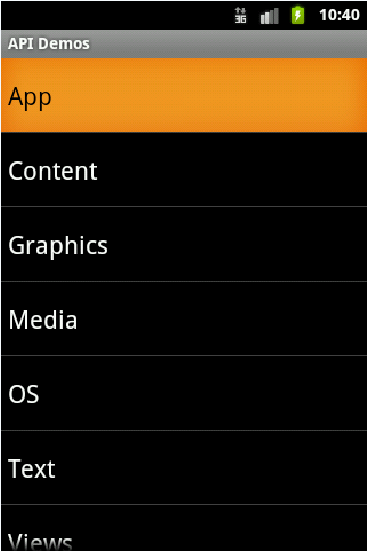
2、 将画面定位在Apidemos的首页，并将光标定位在第一项上。

3、 在$Android\_Root\tools目录下运行一下命令：

monkeyrunner monkeyrunnerprogram1.py

4、在运行过程中我们可以看见光标不断向下移动，并且可以在当前目录下我们自定义的截图：

运行前：shotbegin.png



运行后（做了五次下移操作）：shotend.png

