Android 自动化测试初探-1:捕获 Activity 上的 Element

第一部分：前言

Android 系统下应用程序的测试现在应该还算是个新的领域，网上关于这方面的资料很多都是基于白盒测试的，一般都是基于 JUnit 框架和 Android SDK 中 android.test 等命名空间下的内容进行，但是有一个前提，那就是必须要有应用程序的源代码以提供测试接入点，但是这在很多软件公司中是 不现实的。很多测试工程师做的工作是完全黑盒，基本接触不到源代码，白盒测试大部分也是由开发自己完成。

回顾一下 Windows 下的黑盒测试自动化，先前使用的是微软提供的基于.net framework 的 UI Automation 自动化测试框架（要求版本在.net framework 3.0 以上，即 VS.NET 2008 开发环境），对与擅长 C#语言的人来说，使用起来确认比较好用。本人也写了基于 UI Automation 的轻量级的自动化框架，将在以后的博文中引出。

那在 Android 操作系统中能不能做类似于 UI Automation 的事情呢？不幸的是，Android 的权限控制分的非常清楚，不同程序之间的数据访问只能通过 Intent，content provider 类似的功能实现。也就是说你开发的运行在 Android 中的自动化程序想要捕获当前运行的 AUT (Application under Test) 界面上的控件等 Element（该术语引自 UI Automation，觉得翻译成元素有点生硬，故不作翻译）基本不可能，你也拿不到当前 active activity 的引用（截止本文发帖为止，个人暂时没有找到办法获得此引用）。

无路可走了？模拟器里面不能走，外面能不能走？或许可以。

第二部分：捕获 Activity 上的 Element

在 Android 的 SDK 中自带了一个对自动化测试比较有用的工具：hierarchyviewer（位于 SDK 的 tools 目录下）。在模拟器运行的情 况下，使用该工具可以将当前的 Activity 上的 Element 以对象树的形式展现出来，每个 Element 所含的属性也能一一尽显。这有点像 Windows 上运行的 UI SPY，唯一遗憾的是不支持事件的触发。不过没有关系，可以想办法绕，当务之急是能在自行编写的自动化测试代码里找到 Activity 上的 Element。

第一个想到的办法就是看 hierarchyviewer 源码，不巧，网上搜了一下，没有资源。或许 Google 的官网上有，但是上不去。看来只能反编译 了，找来 XJad，暴力之。虽然反编译出来的代码很多地方提示缺少 import，但代码基本上是正确的。看了一下，确实也知道了许多。后来在编写代码的过 程中，确实也证明了如果想引用 hierarchyviewer.jar 这个包并调试，还是需要知道里面的一些设置的。

创建基于 hierarchyviewer.jar 这个包的调用，需要将它和另外两个包，ddmlib.jar（在 hierarchyviewer.jar 同级目录中有）和 org-netbeans-api-visual.jar（需要下载并安装 netbeans，在其安装目录中有）一并导入到工程项目中，因为 hierarchyviewer 的实现依附于这两个包。

想在代码中获取 Activity 上的 Element 需要进行如下几个步骤（如果使用过 hierarchyviewer 这个工具后会发现，自动化代码所要写的就是该工具上的使用步骤）：

1. Ensure adb running

2. Set adb location（因为 hierarchyviewer 和模拟器的沟通完全是依靠 adb 做的，所以设定正确的 adb 程序的位置至关重要，本人就曾在这个问题上栽了半天多）

3. Get Active Device (这个等同动作发生在启动 hierarchyviewer 工具时)

4. Start View Server (等同于工具上 Start Server 菜单触发事件)

5. Load Scene (等同于工具上 Load View Hierarchy 菜单触发事件)

6. Get Root View Node (获得对象树的根节点，这个动作在工具上点击 Load View Hierarchy 菜单后会自动加载)

7. Get Sub View Node (获得想要查找的 View Node，这个动作在工具上点击 Load View Hierarchy 菜单后会自动加载)

说明：上述步骤中一些名称实际上就是 hierarchyviewer 中所提供的可访问方法名称，如 startViewServer、 loadScene、rootNode 等。另外 View Node 实际上 hierarchyviewer 中的一个类，表示的对象树上的一个 Element。

现将部分核心代码粘贴如下：

1. Set adb location

```
System.setProperty("hierarchyviewer.adb","E:\\ \\Android\\android-sdk-windows\\tools");
```

其中”hierarchyviewer.adb” 这个 key 是 hierarchyviewer.jar 中指定的，后面的 value 是存放 Android SDK 的路径。这个目录必须是当前运行的模拟器所对应的 adb 的目录，不能自行使用其他目录下 adb，否则会发生 adb 进程异常退出的错误。

2. Get Active Device

```
IDevice[] devices = null;

DeviceBridge.terminate();

while(null==devices || 0==devices.length){

DeviceBridge.initDebugBridge() ;

//it must wait for some time, otherwise will throw exception

try {

    Thread.sleep(1000);
```

```
} catch (InterruptedException e) {

e.printStackTrace();

}

devices = DeviceBridge.getDevices() ;

}

return devices;
```

以上方法返回的是所有当前运行的 Device 列表

3. Start View Server

```
DeviceBridge.startViewServer(device);
```

4. Load Scene

```
ViewHierarchyLoader.loadScene(device,Window.FOCUSED_WINDOW) ;
```

5. Get Root View Node

```
vhs.getRoot() ;
```

其中 vhs 是 ViewHierarchyScene 的实例对象

6. Get Sub View Node

```
public ViewNode findFirstChildrenElement(IDevice device, ViewNode entryViewNode, String
elementID){

        ViewNode node=null;

        if(0!=entryViewNode.children.size()){

            for(int i=0;i<entryViewNode.children.size();i++){

                node=entryViewNode.children.get(i);

                if(node.id==elementID)

                    return node;

                else
```

```
                continue;
            }
        }
        return node;}
```

虽然上述步骤所涉及的代码量不多，但是花了一天多的时间才终究研究出来，写下此文希望对正在研究 Android 自动化测试的同学们有些帮助。

到目前为止，Element 已经得到了，接下去就是实现怎么去触发这些 Element,如 click button，enter text 等等，尚需再慢慢研究，等有结果再贴出来分享！

Android 自动化测试初探-2: Hierarchyviewer 捕获 Element 的实现原理

Android SDK tools 下的工具 hierarchyviewer 可以展现 Device 上的 Element 的层次分布和自身属性，其核心函数之一就是 LoadScene，研究后发现其实现方法是向 Device 的 4939 端口通过 socket 的方式发送了一个 DUMP 的命令，Device 会自动处理该命 令并将所有 Screen 上的 Element 层次结构和属性一并发回，实现代码如下：

```
public static void listElement(IDevice device){
Socket socket;
    BufferedReader in;
    BufferedWriter out;
    socket = null;
    in = null;
    out = null;

    do{
     DeviceBridge.setupDeviceForward(device);
    }while(4939!=DeviceBridge.getDeviceLocalPort(device));
```

```
        socket = new Socket();
        try {
socket.connect(new InetSocketAddress("127.0.0.1", DeviceBridge.getDeviceLocalPort(device)));
out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

System.out.println("==> DUMP");
out.write((new StringBuilder()).append("DUMP -1").toString());

out.newLine();
            out.flush();
            do
            {
                String line;
if ((line = in.readLine()) == null || "DONE.".equalsIgnoreCase(line))
                    break;
                line = line.trim();
                System.out.println(line);
            } while (true);
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

运行后的结果摘录其中一部分（button5），列举如下。注：当前 device 中运行的是 2.1SDK 中自带的 Calculator 程序：

com.android.calculator2.ColorButton@43b8bee8 mText=1,5 getEllipsize()=4,null mMinWidth=1,0 mMinHeight=1,0 mMeasuredWidth=2,79 mPaddingBottom=1,0 mPaddingLeft=1,0 mPaddingRight=1,0 mPaddingTop=1,0 mMeasuredHeight=2,78 mLeft=2,81 mPrivateFlags_DRAWING_CACHE_INVALID=3,0x0 mPrivateFlags_DRAWN=4,0x20 mPrivateFlags=8,16779312 mID=9,id/digit5 mRight=3,160 mScrollX=1,0 mScrollY=1,0 mTop=1,0 mBottom=2,78 mUserPaddingBottom=1,0 mUserPaddingRight=1,0 mViewFlags=9,402669569 getBaseline()=2,54 getHeight()=2,78 layout_gravity=4,NONE layout_weight=3,1.0 layout_bottomMargin=1,0 layout_leftMargin=1,1 layout_rightMargin=1,0 layout_topMargin=1,0 layout_height=11,FILL_PARENT layout_width=11,FILL_PARENT getTag()=4,null getVisibility()=7,VISIBLE getWidth()=2,79 hasFocus()=5,false isClickable()=4,true isDrawingCacheEnabled()=5,false isEnabled()=4,true isFocusable()=4,true isFocusableInTouchMode()=5,false isFocused()=5,false isHapticFeedbackEnabled()=4,true isInTouchMode()=4,true isOpaque()=5,false isSelected()=5,false isSoundEffectsEnabled()=4,true willNotCacheDrawing()=5,false willNotDraw()=5,false

另外还支持如下命令：

- LIST will show the list of windows:

LIST

43514758 com.android.launcher/com.android.launcher.Launcher

4359e4d0 TrackingView

435b00a0 StatusBarExpanded

43463710 StatusBar

43484c58 Keyguard

DONE.

Android 自动化测试初探-3：架构实现

前两节讲了用 Android SDK 自带的 tool-hierarchyviewer 来捕获 Activity 上 Element，并分析了其中的原理。对于要实现 GUI 自动化，还有哪些工作没有完成呢？

n　Invoke 界面上的 Element，如点击按钮，在文本框中输入内容等

n Press 手机自身所有的按键，如 HOME 键，Menu 键，左右上下方向键，通话键，挂机键等

n 判断测试结果

前面说过，直接从 Emulator 内部获取当前 Activity 上的 Element 这条路已经断了，同理，探索像 UI Automation 上一样 Invoke Element 的操作估计是行不通了，因为你拿不到 Element 的对象实例，所以实例所支持的方法当然也没有办法拿到。

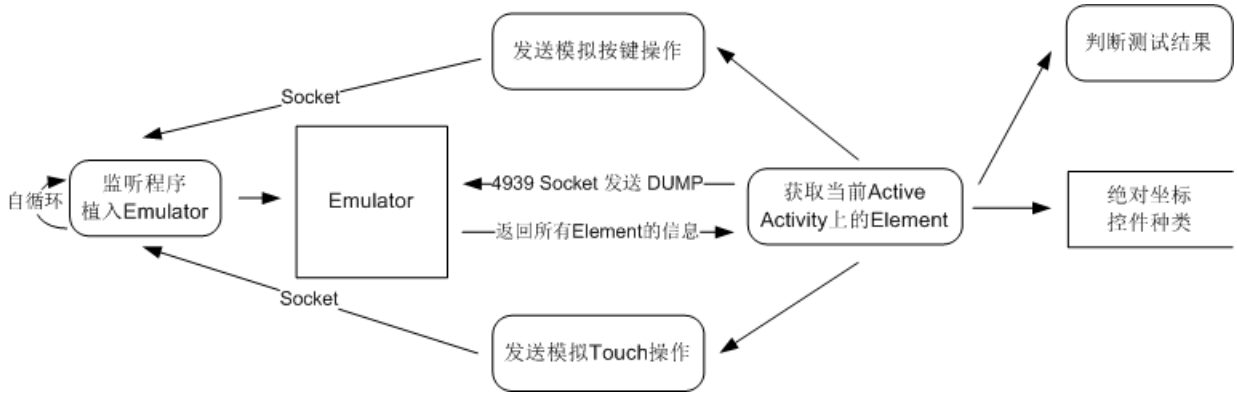
怎么办？实在不行，基于坐标来对 Element 进行触发总可以吧。在 Windows 中发送基于坐标发送键盘和鼠标事件一般是在无法识别 Element 的情况下，想的最后一招，这使我想起起了 Android 中的 monkey 测试，对着屏幕就是一通乱点，压根就不管点的是什么。所幸的是，当 前 Android 系统中我们得到了 Element 的属性信息，其中就包括坐标信息，而且这种信息是具有弹性的，也就是说即使 Element 的坐标随着开发 的改变而有所变化，也不用担心，因为当前的坐标是实时获得的。

那么怎样才能给 Element 发送模拟按键等操作呢？总不能用 Windows 当前的键盘和鼠标事件吧，那样一旦模拟器的位置改变或失去焦点，啥都白 搭，风险太大了。看来给 Emulator 内部发送模拟按键等操作比较靠谱。查了一下 SDK，其中确实有这样的方法存在，但是我们当前的测试基础架构程序位 于 Emulator 外部，怎么办？突然想起了 hierarchyviewer 的实现机制，通过 Socket 来发送信息。Hierarchyviewer 有 系统自带的进程给予答复响应（具体是哪个进程进行的响应不清楚，没有研究过）。那么我们也来模拟做一个 Listener 总可以吧。

其实对于模拟按键发送，网上的帖子很多，但大部分是基于一种方式实现的，IWindowManager 接口。不巧的是，SDK 并没有将该接口提供为 public，所以需要用 android 源码替代 android.jar 包进行编译的方式进行绕行，感觉方法有点复杂。在后面另一篇系列文章中我会列出我 在网上看到的另一种基于 Instrumentation 和 MessageQueue 的机制实现方法。

最后就剩下判断测试结果了。判断测试结果一般分为如下两种：外部条件是否满足，如文件是否产生，数据是否生成等；内部条件是否满足，如对应的 Element 是否出现或消失，Element 上内容如字符串是否有变化。对于前一种本文不予讨论，后一种情况下，Element 出现或消失可以通过 hierarchyviewer 来获取。仔细研究过 hierarchyviewer 会发现，它并没有提供 Element 界面上内容（Text）的属性。这 可有点晕了，好像又要回到实现捕获 Activity 实例的老路上来了。考虑图像识别？这好像不靠谱。突然想到，4939 端口上发送 DUMP 命令后的返回结 果中会不会有此类 hierarchyviewer 没有显示出来的信息呢，万幸，还真有。在我上一篇博文（Hierarchyviewer 捕获 Element 的实现原理）中查询 mText 字段，会发现 mText=1,5 这样的信息，其实就是代表了计算器 Button5 上显示的内容 5，逗号前 的 1 表示后跟一位信息。

至此，问题似乎都解决掉了。画个基础架构图做个总结：



Android 自动化测试初探-4:模拟键盘鼠标事件

通过 Socket + Instrumentation 实现模拟键盘鼠标事件主要通过以下三个部分组成：

l Socket 编程：实现 PC 和 Emulator 通讯，并进行循环监听

l Service 服务：将 Socket 的监听程序放在 Service 中，从而达到后台运行的目的。这里要说明的是启动服务有两种方 式，bindService 和 startService，两者的区别是，前者会使启动的 Service 随着启动 Service 的 Activity 的消亡而 消亡，而 startService 则不会这样，除非显式调用 stopService，否则一直会在后台运行因为 Service 需要通过一个 Activity 来进行启动，所以采用 startService 更适合当前的情形

l Instrumentation 发送键盘鼠标事件：Instrumentation 提供了丰富的以 send 开头的函数接口来实现模拟键盘鼠标，如下所述：

sendCharacterSync(int keyCode) //用于发送指定 KeyCode 的按键

sendKeyDownUpSync(int key) //用于发送指定 KeyCode 的按键

sendPointerSync(MotionEvent event) //用于模拟 Touch

sendStringSync(String text) //用于发送字符串

注意：以上函数必须通过 Message 的形式抛到 Message 队列中。如果直接进行调用加会导致程序崩溃。

对于 Socket 编程和 Service 网上有很多成功的范例，此文不再累述，下面着重介绍一下发送键盘鼠标模拟事件的代码：

1. 发送键盘 KeyCode：

步骤 1. 声明类 handler 变量

```
private static Handler handler;
```

步骤 2. 循环处理 Message

```
//在 Activity 的 onCreate 方法中对下列函数进行调用
private void createMessageHandleThread(){
    //need start a thread to raise looper, otherwise it will be blocked
        Thread t = new Thread() {
            public void run() {
                Log.i( TAG,"Creating handler ..." );
                Looper.prepare();
                handler = new Handler(){
                    public void handleMessage(Message msg) {
                        //process incoming messages here
                    }
                };
                Looper.loop();
                Log.i( TAG, "Looper thread ends" );
            }
        };
        t.start();
}
```

步骤 3. 在接收到 Socket 中的传递信息后抛出 Message

```
handler.post( new Runnable() {
            public void run() {
Instrumentation inst=new Instrumentation();
```

inst.sendKeyDownUpSync(keyCode);

}

});

2. Touch 指定坐标，如下例子即 touch point（240,400）

Instrumentation inst=new Instrumentation();

inst.sendPointerSync(MotionEvent.obtain(SystemClock.uptimeMillis(),SystemClock.uptimeMillis(), MotionEvent.ACTION_DOWN, 240, 400, 0));

inst.sendPointerSync(MotionEvent.obtain(SystemClock.uptimeMillis(),SystemClock.uptimeMillis(), MotionEvent.ACTION_UP, 240, 400, 0));

3. 模拟滑动轨迹

将上述方法中间添加 MotionEvent.ACTION_MOVE

Android 自动化测试初探（五）：再述模拟键盘鼠标事件（adb shell 实现）

上一篇博文中讲述了通过 Socket 编程从外部向 Emulator 发送键盘鼠标模拟事件，貌似实现细节有点复杂。其实 Android 还有一种更简单的模拟键盘鼠标事件的方法，那就是通过使用 adb shell 命令。

1. 发送键盘事件：

命令格式 1：adb shell input keyevent “value”

其中 value 以及对应的 key code 如下表所列：

KeyEvent Value

KEYCODE
Comment
0
KEYCODE_UNKNOWN
1
KEYCODE_MENU
在 SDK2.1 的模拟器中命令失效，sendevent 命令可行
2
KEYCODE_SOFT_RIGHT

3
KEYCODE_HOME
4
KEYCODE_BACK
5
KEYCODE_CALL
6
KEYCODE_ENDCALL
7

KEYCODE_0
8
KEYCODE_1
9
KEYCODE_2
10
KEYCODE_3
11
KEYCODE_4
12
KEYCODE_5
13
KEYCODE_6
14
KEYCODE_7
15
KEYCODE_8
16
KEYCODE_9
17
KEYCODE_STAR
18
KEYCODE_POUND
19
KEYCODE_DPAD_UP
20
KEYCODE_DPAD_DOWN
21
KEYCODE_DPAD_LEFT
22
KEYCODE_DPAD_RIGHT
23
KEYCODE_DPAD_CENTER
24

KEYCODE_VOLUME_UP
25
KEYCODE_VOLUME_DOWN
26
KEYCODE_POWER
27
KEYCODE_CAMERA
28
KEYCODE_CLEAR
29
KEYCODE_A
30
KEYCODE_B
31
KEYCODE_C
32
KEYCODE_D
33
KEYCODE_E
34
KEYCODE_F
35
KEYCODE_G
36
KEYCODE_H
37
KEYCODE_I
38
KEYCODE_J
39
KEYCODE_K
40
KEYCODE_L
41
KEYCODE_M
42
KEYCODE_N
43
KEYCODE_O
44
KEYCODE_P
45
KEYCODE_Q
46

KEYCODE_R
47
KEYCODE_S
48
KEYCODE_T
49
KEYCODE_U
50
KEYCODE_V
51
KEYCODE_W
52
KEYCODE_X
53
KEYCODE_Y
54
KEYCODE_Z
55
KEYCODE_COMMA
56
KEYCODE_PERIOD
57
KEYCODE_ALT_LEFT
58
KEYCODE_ALT_RIGHT
59
KEYCODE_SHIFT_LEFT
60
KEYCODE_SHIFT_RIGHT
61
KEYCODE_TAB
62
KEYCODE_SPACE
63
KEYCODE_SYM
64
KEYCODE_EXPLORER
65
KEYCODE_ENVELOPE
66
KEYCODE_ENTER
67
KEYCODE_DEL
68

KEYCODE_GRAVE
69
KEYCODE_MINUS
70
KEYCODE_EQUALS
71
KEYCODE_LEFT_BRACKET
72
KEYCODE_RIGHT_BRACKET
73
KEYCODE_BACKSLASH
74
KEYCODE_SEMICOLON
75
KEYCODE_APOSTROPHE
76
KEYCODE_SLASH
77
KEYCODE_AT
78
KEYCODE_NUM
79
KEYCODE_HEADSETHOOK
80
KEYCODE_FOCUS
81
KEYCODE_PLUS
82
KEYCODE_MENU
83
KEYCODE_NOTIFICATION
84
KEYCODE_SEARCH
85
TAG_LAST_KEYCODE
命令格式 2：adb shell sendevent [device] [type] [code] [value]

如： adb shell sendevent /dev/input/event0 1 229 1 代表按下按下 menu 键

adb shell sendevent /dev/input/event0 1 229 0 代表按下松开 menu 键

说明：上述的命令需组合使用

另外所知道的命令如下：

Key Name CODE

MENU 229

HOME 102

BACK (back button) 158

CALL (call button) 231

END (end call button) 107

2. 发送鼠标事件(Touch)：

命令格式：adb shell sendevent [device] [type] [code] [value]

情况 1：在某坐标点上 touch

如在屏幕的 x 坐标为 40，y 坐标为 210 的点上 touch 一下，命令如下

```
adb shell sendevent /dev/input/event0 3 0 40
adb shell sendevent /dev/input/event0 3 1 210

adb shell sendevent /dev/input/event0 1 330 1 //touch
adb shell sendevent /dev/input/event0 0 0 0 //it must have

adb shell sendevent /dev/input/event0 1 330 0 //untouch
adb shell sendevent /dev/input/event0 0 0 0 //it must have
```

注：以上六组命令必须配合使用，缺一不可

情况 2：模拟滑动轨迹（可下载并采用 aPaint 软件进行试验）

如下例是在 aPaint 软件上画出一条开始于（100,200），止于（108,200）的水平直线

```
adb shell sendevent /dev/input/event0 3 0 100 //start from point (100,200)
adb shell sendevent /dev/input/event0 3 1 200

adb shell sendevent /dev/input/event0 1 330 1 //touch
adb shell sendevent /dev/input/event0 0 0 0

adb shell sendevent /dev/input/event0 3 0 101 //step to point (101,200)
adb shell sendevent /dev/input/event0 0 0 0
```

…… //must list each step, here just skip
adb shell sendevent /dev/input/event0 3 0 108 //end point(108,200)
adb shell sendevent /dev/input/event0 0 0 0

adb shell sendevent /dev/input/event0 1 330 0 //untouch
adb shell sendevent /dev/input/event0 0 0 0

尝试 Android Scripting Environment

前言
研究了一下 ASE (Android Scripting Environment: http://code.google.com/p/android-scripting/), 这个很火的项目（ http://www.javaeye.com /topic/407925 ）目前还处于 Alpha Release 状态。

它封装了一些 Java API, 从而可以通过脚本 (Python,LUA 等）去访问 某些 Java API, 从而大大提供开发效率。

目前封装的 API 包括 拨号 / 短信 /wifi/bt/barcode ， 等等，具体见：
http://code.google.com/p/android-scripting/wiki/ApiReference

首先从： http://code.google.com/p/android-scripting/downloads/list 下载到最新版本的 ASE, 比如 ASE_r25.apk

启动 Android 设备或者虚拟设备， 通过 adb install 将 ASE 安装到目标机上。

在目标机上运行 ASE, 首先通过 Menu->Add Interpreter 安装一种解释器，比如 Python.

安装完成后会自动下载相关脚本，并切换到脚本试图。

现在我们可以上下选择并运行某个脚本。

修改代码
下面我们尝试修改代码。由于直接在目标机上编辑比较麻烦，我们先在 PC 侧编辑好后再用 ADB shell 将其 push 到目标机上。

从 Logcat 信息我们可以看到 ASE 会下载脚本包：

http://android-scripting.googlecode.com/files/python_scripts_r7.zip

并解压缩到： /sdcard/ase/scripts

所以，我们：

1） 使用 wget 下载相关脚本：

wget http://android-scripting.googlecode.com/files/python_scripts_r7.zip

2） 解开这个包。修改代码：

unzip python_scripts_r7.zip

cp hello_world.py hello_world2.py

vi hello_world2.py

修改一下打印语句，比如 droid.makeToast('Goodbye, Android!')

3） 将这个修改的文件放到目标机上：

adb push hello_world2.py /sdcard/ase/scripts

4） 在目标机上运行 ASE ，可以看到我们修改的脚本，并可以运行！

通过 adb shell 执行脚本
通过 ASE 执行 脚本比较慢，需要人工操作！ 我们尝试在 PC 侧使用脚本通过 adb 来直接运行！

看 ASE 执行脚本的过程，似乎也只是调出了命令行，设置了一些环境变量， 最终运行脚本。

所以我们模仿它写了一个脚本， 如下脚本以待执行的 python 脚本为参数，将其通过 adb 放到 目标机上执行

```
#!/bin/sh

#================================================================

# By:    zjujoe@yahoo.com

# YOU SHOULD INSTALL ASE AND PYTHON INTERPRETER ON TARGET FIRSTLY

#

# I am using ase_r25.apk, for other version, blow path may change

# You should change AP_PORT with the one from ASE console

#

#start ASE sever first:

# adb -s emulator-5554 shell am start -a com.google.ase.action.LAUNCH_SERVER \
```

```
#       -n com.google.ase/.activity.AseServiceLauncher

#

#the get AP_PORT number for notification

#================================================================

#set -x

#change this which ASE server listion to

AP_PORT="48620"

DEVICE="-s emulator-5554"

#the script file to be run in ase

if [ $# -eq "1" ]; then

    FILE=$1

else

    echo $0 filename

    exit

fi

#shell script file to be run on target(which will run ase)

FILE2=t1.sh

TARGET_RUN_LOCATION=/data

TARGET_PYTHON_SCRIPTS_LOCATION=/sdcard/ase/scripts

#prepare the FILE2

echo export PYTHONPATH="/sdcard/ase/extras/python:/sdcard/ase/scripts/" > ${FILE2}

echo export AP_PORT=${AP_PORT} >> ${FILE2}

echo export TEMP="/sdcard/ase/extras/pythontmp" >> ${FILE2}

echo export PYTHONHOME="/data/data/com.google.ase/python" >> ${FILE2}
```

```
echo /data/data/com.google.ase/python/bin/python /sdcard/ase/scripts/${FILE} >> ${FILE2}

#push FILE to target

adb ${DEVICE} push ${FILE} ${TARGET_PYTHON_SCRIPTS_LOCATION}

#push FILE2 to target

adb ${DEVICE} push ${FILE2} ${TARGET_RUN_LOCATION}

adb ${DEVICE} shell chmod 777 ${TARGET_RUN_LOCATION}/${FILE2}

#run FILE2

adb ${DEVICE} shell ${TARGET_RUN_LOCATION}/${FILE2}

#remove FILE2 from target & local

adb ${DEVICE} shell rm ${TARGET_RUN_LOCATION}/${FILE2}

rm ${FILE2}

#remove FILE from target

#adb ${DEVICE} shell rm ${FILE}
${TARGET_PYTHON_SCRIPTS_LOCATION}
```

如上脚本可以很好的在 PC 侧直接执行，从而解决了自动化的问题。

尝试 Android Scripting Environment 之二

前言

前文 << 尝试 ASE>> 中我们写了一个脚本，可以通过 ADB 在 PC 端控制 ASE Python 脚本的执行。 这里，我们就通过该脚本，研究一下 ASE 自带的例子，从而知道 ASE 的能力，为我们或许编写自己的测试脚本做准备。

简介

ASE 通过 Android Façade 封装了很多方法，这些方法调用了 Android 的 API 来实现功能。我们正是通过这些方法来实现我们的功能。

从： http://code.google.com/p/android-scripting/wiki/AndroidFacadeAPI

可知，这些方法都有相同的返回值：

All ASE API calls return an object with three fields:

id: a strictly increasing, numeric id associated with the API call.
result: the return value of the API call, or null if there is no return value.
error: a description of any error that occurred or null if no error occurred.
从： http://code.google.com/p/android-scripting/wiki/ApiReference

可以查到所有的 API

当然，具体看示例代码时会发现，除了 Android FacadeAPI 对应的 android package 外，还用到很多其他 package，由于本人对 python 不熟（语法还是大体知道的，但是那么多 package 就基本不懂了）， 所以研究的过程中需要不时 Google 一下那些包的使用，一并记录在此。

Python 的参考可以用： http://docs.python.org/release/2.5/lib/lib.html

当然， ASE 带的 python 似乎还添加了一些非标准包，比如 Google doc service.

hello_world.py
1 import android

这里导入 android 包

2 droid = android.Android()

获得 android 对象

3 droid.makeToast('Hello, Android!')

调用 android 方法： makeToast

对 Android API 熟悉的话应该知道 Toast 就是一个提示对话框。

test.py 各种测试
比较直白的部分我们就不分析了。。。

import 部分
1 import sys

2 import types

3

4 # Test imports.

5 import android

6 import BeautifulSoup

From: http://hi.baidu.com/javalang/blog/item/84bac4bf731fb80f18d81fe1.html

Beautiful Soup 是用 Python 写的一个 HTML/XML 的解析器，它可以很好的处理不规范标记并生成剖析树 (parse tree) 。 它提供简单又常用的导航（ navigating ），搜索以及修改剖析树的操作。它可以大大节省你的编程时间。 对于 Ruby ， 使用 Rubyful Soup 。

7 import gdata.docs.service

From: http://code.google.com/p/gdata-python-client/

The Google Data APIs (Google Data) provide a simple protocol for reading and writing data on the web. Though it is possible to use these services with a simple HTTP client, this library provides helpful tools to streamline your code and keep up with server-side changes.

8 import sqlite3

From: http://docs.python.org/release/2.5/lib/module-sqlite3.html

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

9 import termios

From: http://docs.python.org/release/2.5/lib/module-termios.html

This module provides an interface to the POSIX calls for tty I/O control. For a complete description of these calls, see the POSIX or Unix manual pages. It is only available for those Unix versions that support POSIX termios style tty I/O control (and then only if configured at installation time).

10 import time

From: http://docs.python.org/release/2.5/lib/module-time.html

This module provides various time-related functions. It is always available, but not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

11 import xmpp

From: http://xmpppy.sourceforge.net/

xmpppy is a Python library that is targeted to provide easy scripting with Jabber. Similar projects are Twisted Words and jabber.py.

12

13 droid = android.Android()

14

15

event_loop
Helper 函数

16 def event_loop():

17 for i in range(10):

18 e = droid.receiveEvent()

Receives the most recent event (i.e. location or sensor update, etc.)

19 if e.result is not None:

20 return True

21 time.sleep(2)

22 return False

尝试接收一个 event, retry 10 次。

test_clipboard ：
尝试系统剪切板能否工作

25 def test_clipboard():

26 previous = droid.getClipboard().result

27 msg = 'Hello, world!'

28 droid.setClipboard(msg)

29 echo = droid.getClipboard().result

30 droid.setClipboard(previous)

```
31    return echo == msg
```

test_gdata
尝试连接 Google doc service

```
34 def test_gdata():
35    # Create a client class which will make HTTP requests with Google Docs server.
36    client = gdata.docs.service.DocsService()
37
38    # Authenticate using your Google Docs email address and password.
39    username = droid.getInput('Username').result
40    password = droid.getPassword('Password', 'For ' + username).result
41    try:
42     client.ClientLogin (username, password)
43    except:
44     return False
45
46    # Query the server for an Atom feed containing a list of your documents.
47    documents_feed = client.GetDocumentListFeed()
48    # Loop through the feed and extract each document entry.
49    return bool(list(documents_feed.entry))
```

test_gps

```
52 def test_gps():
53    droid.startLocating()
54    try:
55     return event_loop()
56    finally:
```

```
57      droid.stopLocating()
```

test_sensors(运行失败)

```
60 def test_sensors():
61    droid.startSensing()
62    try:
63      return event_loop()
64    finally:
65      droid.stopSensing()
```

test_speak (pass 但是听不到声音)

```
68 def test_speak():
69    result = droid.speak('Hello, world!')
```

Speaks the provided message via TTS.

```
70    return result.error is None
```

test_phone_state 跟踪 phone 状态

```
73 def test_phone_state():
74    droid.startTrackingPhoneState()
75    try:
76      return event_loop()
77    finally:
78      droid.stopTrackingPhoneState()
```

test_ringer_silent 打开、关闭 静音模式

```
81 def test_ringer_silent():
82    result1 = droid.toggleRingerSilentMode()
83    result2 = droid.toggleRingerSilentMode()
84    return result1.error is None and result2.error is None
```

test_ringer_volume 设置音量

```
87 def test_ringer_volume():

88    get_result = droid.getRingerVolume()

89    if get_result.error is not None:

90      return False

91    droid.setRingerVolume(0)

92    set_result = droid.setRingerVolume(get_result.result)

93    if set_result.error is not None:

94      return False

95    return True
```

test_get_last_known_location

```
98 def test_get_last_known_location():

99    result = droid.getLastKnownLocation()

100    return result.error is None
```

test_geocode

```
103 def test_geocode():

104    result = droid.geocode(0.0, 0.0, 1)

105    return result.error is None
```

test_wifi 打开、关闭 Wifi

```
108 def test_wifi():

109    result1 = droid.toggleWifiState()

110    result2 = droid.toggleWifiState()

111    return result1.error is None and result2.error is None
```

test_make_toast

```
114 def test_make_toast():

115    result = droid.makeToast('Hello, world!')
```

```
116    return result.error is None

test_vibrate
119 def test_vibrate():

120    result = droid.vibrate()

121    return result.error is None

test_notify
124 def test_notify():

125    result = droid.notify('Hello, world!')

126    return result.error is None

test_get_running_packages
129 def test_get_running_packages():

130    result = droid.getRunningPackages()

131    return result.error is None

test_alert_dialog
134 def test_alert_dialog():

135    title = 'User Interface'

136    message = 'Welcome to the ASE integration test.'

137    droid.dialogCreateAlert(title, message)

138    droid.dialogSetPositiveButtonText('Continue')

139    droid.dialogShow()

140    response = droid.dialogGetResponse().result

141    return response['which'] == 'positive'

test_alert_dialog_with_buttons
144 def test_alert_dialog_with_buttons():

145    title = 'Alert'

146    message = ('This alert box has 3 buttons and '
```

```
147             'will wait for you to press one.')
148    droid.dialogCreateAlert(title, message)
149    droid.dialogSetPositiveButtonText('Yes')
150    droid.dialogSetNegativeButtonText('No')
151    droid.dialogSetNeutralButtonText('Cancel')
152    droid.dialogShow()
153    response = droid.dialogGetResponse().result
154    return response['which'] in ('positive', 'negative', 'neutral')
```

test_spinner_progress 旋转进度条

```
157 def test_spinner_progress():
158    title = 'Spinner'
159    message = 'This is simple spinner progress.'
160    droid.dialogCreateSpinnerProgress(title, message)
161    droid.dialogShow()
162    time.sleep(2)
163    droid.dialogDismiss()
164    return True
```

test_horizontal_progress 水平进度条

```
167 def test_horizontal_progress():
168    title = 'Horizontal'
169    message = 'This is simple horizontal progress.'
170    droid.dialogCreateHorizontalProgress(title, message, 50)
171    droid.dialogShow()
172    for x in range(0, 50):
173     time.sleep(0.1)
```

```
174     droid.dialogSetCurrentProgress(x)

175    droid.dialogDismiss()

176    return True
```

test_alert_dialog_with_list 列表对话框

```
179 def test_alert_dialog_with_list():

180    title = 'Alert'

181    droid.dialogCreateAlert(title)

182    droid.dialogSetItems(['foo', 'bar', 'baz'])

183    droid.dialogShow()

184    response = droid.dialogGetResponse().result

185    return True
```

test_alert_dialog_with_single_choice_list

```
188 def test_alert_dialog_with_single_choice_list():

189    title = 'Alert'

190    droid.dialogCreateAlert(title)

191    droid.dialogSetSingleChoiceItems(['foo', 'bar', 'baz'])

192    droid.dialogSetPositiveButtonText('Yay!')

193    droid.dialogShow()

194    response = droid.dialogGetResponse().result

195    return True
```

test_alert_dialog_with_multi_choice_lis

```
198 def test_alert_dialog_with_multi_choice_list():

199    title = 'Alert'

200    droid.dialogCreateAlert(title)

201    droid.dialogSetMultiChoiceItems(['foo', 'bar', 'baz'], [])
```

```
202    droid.dialogSetPositiveButtonText('Yay!')

203    droid.dialogShow()

204    response = droid.dialogGetResponse().result

205    return True
```

Test engine

```
208 if __name__ == '__main__':

209    for name, value in globals().items():

210     if name.startswith('test_') and isinstance(value, types.FunctionType):

211       print 'Running %s...' % name,

212       sys.stdout.flush()

213       if value():

214         print ' PASS'

215       else:

216         print ' FAIL'
```

bluetooth_chat.py 蓝牙聊天

```
1 import android

2 import time

3

4 droid = android.Android()

5 droid.toggleBluetoothState(True)

6 droid.dialogCreateAlert('Be a server?')

7 droid.dialogSetPositiveButtonText('Yes')

8 droid.dialogSetNegativeButtonText('No')

9 droid.dialogShow()

10 result = droid.dialogGetResponse()
```

```
11 is_server = result.result['which'] == 'positive'
12 if is_server:
13    droid.bluetoothMakeDiscoverable()
14    droid.bluetoothAccept()
15 else:
16    droid.bluetoothConnect()
17
18 if is_server:
19    result = droid.getInput('Chat', 'Enter a message').result
20    if result is None:
21     droid.exit()
22    droid.bluetoothWrite(result + '\n')
23
24 while True:
25    message = droid.bluetoothReadLine().result
26    droid.dialogCreateAlert('Chat Received', message)
27    droid.dialogSetPositiveButtonText('Ok')
28     droid.dialogShow()
29    droid.dialogGetResponse()
30    result = droid.getInput('Chat', 'Enter a message').result
31    if result is None:
32     break
33    droid.bluetoothWrite(result + '\n')
34
```

```
35 droid.exit()
```

say_chat.py 聊天

```
1 """Say chat messages aloud as they are received."""

2

3 __author__ = 'Damon Kohler <damonkohler@gmail.com>'

4 __copyright__ = 'Copyright (c) 2009, Google Inc.'

5 __license__ = 'Apache License, Version 2.0'

6

7 import android

8 import xmpp

9

10 _SERVER = 'talk.google.com', 5223

11

12 def log(droid, message):

13   print message

14   self.droid.speak(message)

15

16

17 class SayChat(object):

18

19   def __init__(self):

20     self.droid = android.Android()

21     username = self.droid.getInput('Username').result

22     password = self.droid.getInput('Password').result
```

```
23    jid = xmpp.protocol.JID(username)
24    self.client = xmpp.Client(jid.getDomain(), debug=[])
25    self.client.connect(server=_SERVER)
26    self.client.RegisterHandler('message', self.message_cb)
27    if not self.client:
28      log('Connection failed!')
29      return
30    auth = self.client.auth(jid.getNode(), password, 'botty')
31    if not auth:
32      log('Authentication failed!')
33      return
34    self.client.sendInitPresence()
35
36  def message_cb(self, session, message):
37    jid = xmpp.protocol.JID(message.getFrom())
38    username = jid.getNode()
39    text = message.getBody()
40    self.droid.speak('%s says %s' % (username, text))
41
42  def run(self):
43    try:
44      while True:
45        self.client.Process(1)
46    except KeyboardInterrupt:
```

```
47       pass
```

take_picture.py 拍照

```
1 import android

2

3 droid = android.Android()

4 droid.cameraTakePicture('/sdcard/foo.jpg')
```

Weather.py

```
1 """Retrieve the weather report for the current location."""

2

3 __author__ = 'T.V. Raman <raman@google.com>'

4 __copyright__ = 'Copyright (c) 2009, Google Inc.'

5 __license__ = 'Apache License, Version 2.0'

6

7

8 import string

9 import urllib

10 import urllib2

11 from xml.dom import minidom

12

13 WEATHER_URL = 'http://www.google.com/ig/api?weather=%s&hl=%s'

14

15

16 def extract_value(dom, parent, child):

17    """Convenience function to dig out weather values."""
```

```
18    return
dom.getElementsByTagName(parent)[0].getElementsByTagName(child)[0].getAttribute('data')
19
20
21 def fetch_weather(location, hl=''):
22    """Fetches weather report from Google
23
24    Args:
25      location: a zip code (94041); city name, state (weather=Mountain View,CA);...
26      hl: the language parameter (language code)
27
28    Returns:
29      a dict of weather data.
30
31    """
32    url = WEATHER_URL % (urllib.quote(location), hl)
33    handler = urllib2.urlopen(url)
34    data = handler.read()
35    dom = minidom.parseString(data)
36    handler.close()
37
38    data = {}
39    weather_dom = dom.getElementsByTagName('weather')[0]
40    data['city'] = extract_value(weather_dom, 'forecast_information','city')
```

```
41    data['temperature'] = extract_value(weather_dom, 'current_conditions','temp_f')

42    data['conditions'] = extract_value(weather_dom, 'current_conditions', 'condition')

43    dom.unlink()

44    return data
```

notify_weather.py
getLastKnownLocation

```
1 """Display the weather report in a notification."""

2

3 __author__ = 'Damon Kohler <damonkohler@gmail.com>'

4 __copyright__ = 'Copyright (c) 2009, Google Inc.'

5 __license__ = 'Apache License, Version 2.0'

6

7 import android

8 import weather

9

10

11 def notify_weather(droid):

12    """Display the weather at the current location in a notification."""

13    print 'Finding ZIP code.'

14    location = droid.getLastKnownLocation().result

15    addresses = droid.geocode(location['latitude'], location['longitude'])

16    zip = addresses.result[0]['postal_code']

17    if zip is None:

18      msg = 'Failed to find location.'
```

```
19    else:
20      print 'Fetching weather report.'
21      result = weather.fetch_weather(zip)
22      msg = '%(temperature)s degrees and %(conditions)s, in %(city)s.' % result
23    droid.notify(msg, 'Weather Report', msg)
24    droid.exit()
25
26
27 if __name__ == '__main__':
28    droid = android.Android()
29    notify_weather(droid)
```

say_weather.py

```
1 """Speak the weather."""
2
3 __author__ = 'T.V. Raman <raman@google.com>'
4 __copyright__ = 'Copyright (c) 2009, Google Inc.'
5 __license__ = 'Apache License, Version 2.0'
6
7 import android
8 import weather
9
10
11 def say_weather(droid):
12    """Speak the weather at the current location."""
```

```
13  print 'Finding ZIP code.'
14  location = droid.getLastKnownLocation().result
15  addresses = droid.geocode(location['latitude'], location['longitude'])
16  zip = addresses.result[0]['postal_code']
17  if zip is None:
18   msg = 'Failed to find location.'
19  else:
20   print 'Fetching weather report.'
21   result = weather.fetch_weather(zip)
22   # Format the result for speech.
23   msg = '%(temperature)s degrees and %(conditions)s, in %(city)s.' % result
24  droid.speak(msg)
25
26
27 if __name__ == '__main__':
28  droid = android.Android()
29  say_weather(droid)
```

尝试 Android Scripting Environment 之三

前言

ASE 让人爱不释手，python 也是令人发狂的好东西，所以我们继续深入学习 ASE + Python!

远程运行

根据参考 1 的信息，我们写了一个脚本，和前面的脚本（run_ase_python_scritpt.sh）一样，它可以在 Android 设备上运行位于 PC 侧的 python 脚本。

和 run_ase_python_scritpt.sh 不一样，它并不是把脚本放到目标机上再执行，而是在 PC 本地运行！

通过将某个 TCP 端口转发到目标机，这样：

1. python 的解释执行在 PC 侧

2. 核心的功能调用在目标机上

从而效率更高了！也可见 ASE 支持的 Python 和 PC 侧没有两样！我们也从分体会到分布式计算的魅力！

~/android/testing/ase/python$ cat run_ase_python_scritpt2.sh

```
#!/bin/sh

#====================================================================

# By:     zjujoe@yahoo.com

# YOU SHOULD INSTALL ASE AND PYTHON INTERPRETER ON TARGET FIRSTLY

#

# I am using ase_r25.apk, for other version, blow path may change

#

#start ASE sever first:

# adb -s emulator-5554 shell am start -a com.google.ase.action.LAUNCH_SERVER -n
com.google.ase/.activity.AseServiceLauncher

#

#then get AP_PORT number for notification

#====================================================================

#set -x

#change this which ASE server listion to

TARGET_AP_PORT=40729

DEVICE="-s emulator-5554"

export AP_PORT=9999

#the script file to be run in ase
```

```
if [ $# -eq "1" ]; then
    FILE=$1
else
    echo $0 filename
    exit
fi
if ! [ -e android.py ]; then
    echo "we need anroid.py from ase"
    exit
fi
adb ${DEVICE} forward tcp:${AP_PORT} tcp:${TARGET_AP_PORT}
python $1
```

注意：如果 ase 服务端口设置错误，将会出错：

```
songlixin@zjujoe-desktop:~/android/testing/ase/python$ ./run_ase_python_scritpt2.sh
hello_world.py
Traceback (most recent call last):
File "hello_world.py", line 3, in <module>
    droid.makeToast('Hello, Android!')
File "/home/songlixin/android/testing/ase/python/android.py", line 54, in rpc_call
    return self._rpc(name, *args)
File "/home/songlixin/android/testing/ase/python/android.py", line 43, in _rpc
    response = self.client.readline()
File "/usr/lib/python2.6/socket.py", line 406, in readline
    data = self._sock.recv(self._rbufsize)
```

socket.error: [Errno 104] Connection reset by peer

比较一下两个脚本的运行时间，相差很大！

$ time ./run_ase_python_scritpt.sh hello_world.py

0 KB/s (76 bytes in 0.087s)

5 KB/s (257 bytes in 0.044s)

real 0m1.839s

user 0m0.004s

sys 0m0.016s

$time ./run_ase_python_scritpt2.sh ./hello_world.py

real 0m0.217s

user 0m0.020s

sys 0m0.016s

将脚本放到目标机上
$ adb push my_script.py /sdcard/ase/scripts

目标机上后台执行脚本
adb shell am start -a com.google.ase.action.LAUNCH_SCRIPT -n
com.google.ase/.activity.AseServiceLauncher -e com.google.ase.extra.SCRIPT_NAME
hello_world.py

目标机上终端里执行脚本
adb shell am start -a com.google.ase.action.LAUNCH_TERMINAL -n
com.google.ase/.activity.AseServiceLauncher -e com.google.ase.extra.SCRIPT_NAME
hello_world.py

启动 ASE server
adb shell am start -a com.google.ase.action.LAUNCH_SERVER -n
com.google.ase/.activity.AseServiceLauncher

尝试 Android Scripting Environment 之四

前言

随着学习的深入，我们准备去修改一下源码，扩充其 API 以满足我们自己的个性化要求！比如，能够拨打一个电话，并返回该电话是否成功，是对方忙还是网络信号不好等等。

编译

按照 ASE 的 WiKi 上的文档，基本就可以了。几个注意点：

1） ASE 需要使用 Java1.6

2） AndroidScriptingEnvironmentTest 需要造一个 res 目录，才能编译通过。

添加代码

考虑到代码的干净性，我们决定添加一个项目，在其中添加我们自己开发的 API 函数.

TextToSpeechFacade 比较简单，我们就模仿它了。

cp –a TextToSpeechFacade DragonCustomFacade

然后搜索一下需要修改的地方：

$ find . | xargs grep TextToSpeechFacade

./AndroidScriptingEnvironment/src/com/google/ase/facade/FacadeConfiguration.java: list.addAll(MethodDescriptor.collectFrom(TextToSpeechFacade.class));

./AndroidScriptingEnvironment/src/com/google/ase/facade/FacadeConfiguration.java: receivers.add(new TextToSpeechFacade(service));

./AndroidScriptingEnvironment/.classpath: <classpathentry combineaccessrules="false" kind="src" path="/TextToSpeechFacade"/>

./DragonCustomFacade/src/com/google/ase/facade/TextToSpeechFacade.java:public class TextToSpeechFacade implements RpcReceiver {

./DragonCustomFacade/src/com/google/ase/facade/TextToSpeechFacade.java: public TextToSpeechFacade(Service service) {

./DragonCustomFacade/.project: <name>TextToSpeechFacade</name>

这样， 我们需要修改的地方有：

Ø AndroidScriptingEnvironment/src/com/google/ase/facade/FacadeConfiguration.java

中的两处，这里是把一个 Façade 类挂接到系统中以及初始化该类。所以我们需要把我们的 DragonCustomFacade 相应代码加入，注意初始化函数有几种形式，按需。

Ø ./AndroidScriptingEnvironment/.classpath

这里是定义依赖关系，简单模仿。

Ø ./DragonCustomFacade/src/com/google/ase/facade/TextToSpeechFacade.java

将该 Java 文件改为 DragonCustomFacade.java， 并修改其中的内容。 作为示例，我们只是现实一个 Toast 对话框。后续，我们可以按需扩充其中的功能。

```
package com.google.ase.facade;

import android.app.Service;

import android.widget.Toast;

import com.google.ase.jsonrpc.RpcReceiver;

import com.google.ase.rpc.Rpc;

import com.google.ase.rpc.RpcParameter;

import android.os.Handler;

public class DragonCustomFacade implements RpcReceiver {

private final Handler mHandler = new Handler();

private final Service mService;

public DragonCustomFacade(Service service) {

    mService = service;

}

@Override

public void shutdown() {

}

@Rpc(description = "Displays a short-duration Toast notification.")

public void DragonMakeToast(@RpcParameter(name = "message") final String message) {

    mHandler.post(new Runnable() {
```

```
      public void run() {
        Toast.makeText(mService, message, Toast.LENGTH_SHORT).show();
      }
    });
}
}
```

编译，运行，并用 python 脚本调用我们的函数：

```
$ cat t2.py
import android
droid = android.Android()
droid.DragonMakeToast('Hello, Dragon!')
```

运行之，发现可以打印出 Hello, Dragon!