Appium+window7+python3.5+HTMLTestRunner.

--Android 自动化

--作者：雷子

-- 时间：20170210

一：测试环境搭建：

1. appium 简介

官网： http://appium.io/

官网介绍： Appium is an open source test automation framework for use with native, hybrid and mobile web apps. It drives iOS, Android, and Windows apps using the WebDriver protocol.（注解：Appium是一个开源的自动化测试框架 使用本机, 混合动力 和移动web应用程序。 它使iOS、Android和Windows应用程序使用WebDriver协议。）

2. 下载Appium：

点击 官网的Download Appium自动下载（最新版本）

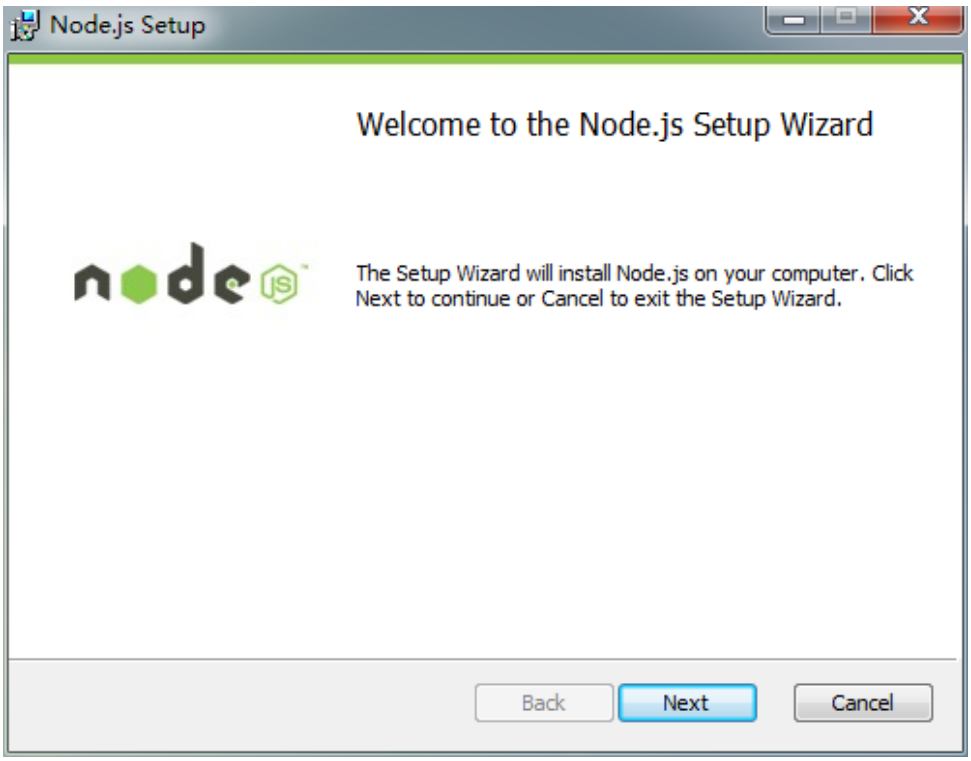
其他版本：https://bitbucket.org/appium/appium.app/downloads/

百度盘下载：http://pan.baidu.com/s/1jGvAlSu

3. 安装node.js,官网提示先装node.js，https://nodejs.org/en/下载node.js。选择的是Windows7（32位）。

下载后点击安装，默认安装 就可以。

安装：



安装完成后，打开 Windows 命令行（win+R 输入 cmd），输入 npm 验证环境

4. 命令行安装：

npm install -g appium

安装会比较慢，但是官方推荐使用这个。

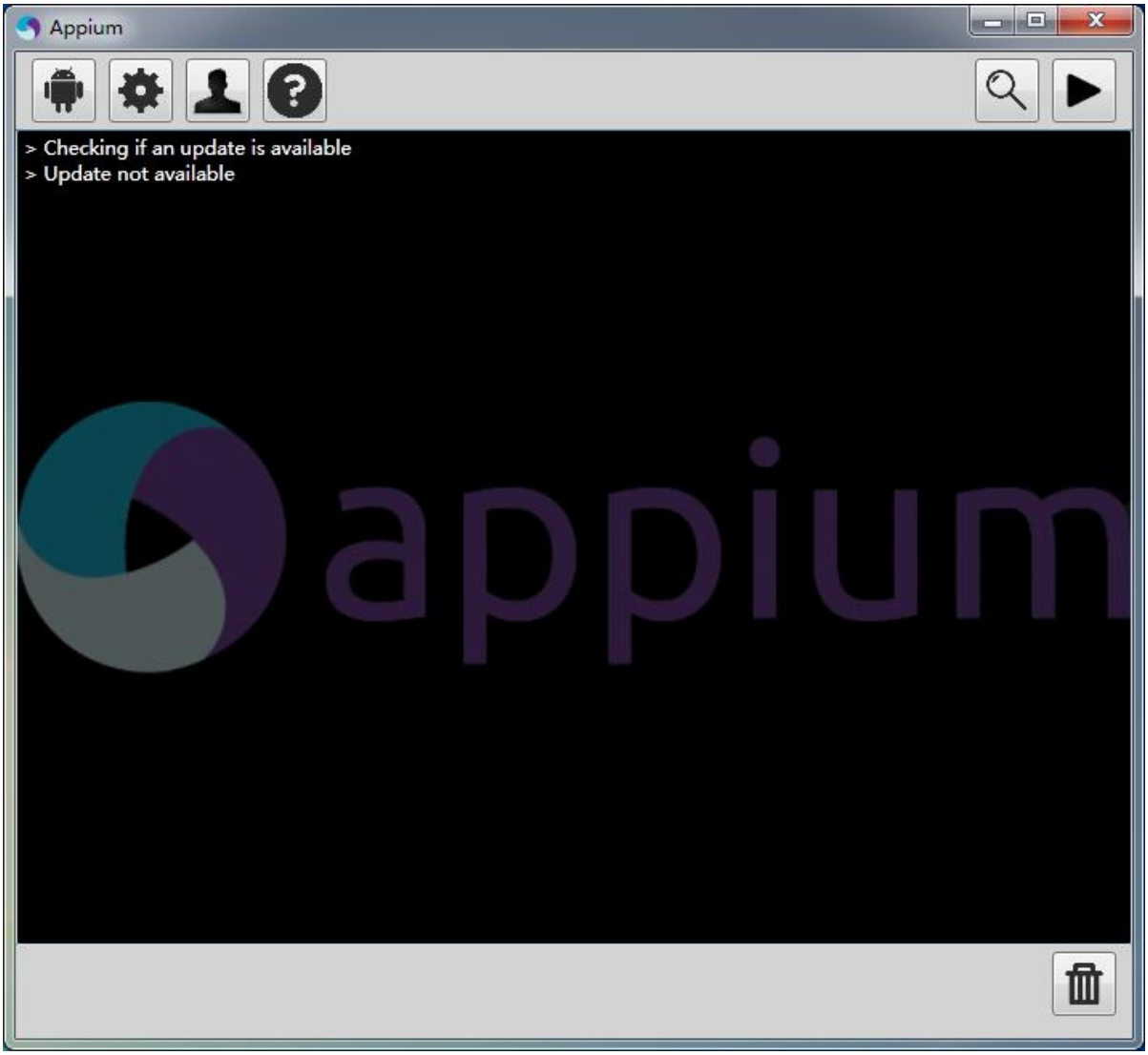
前面我们下载完 appium 的文件，可以直接安装，

会提示缺少.NET Framework ，需要下载这个组件就可以，因为 Appium 是由.NET 开发的，所以，它会依赖 .NET framework 相关组件，下载可以去百度搜索 .net framework ，

点击下载安装就可以。

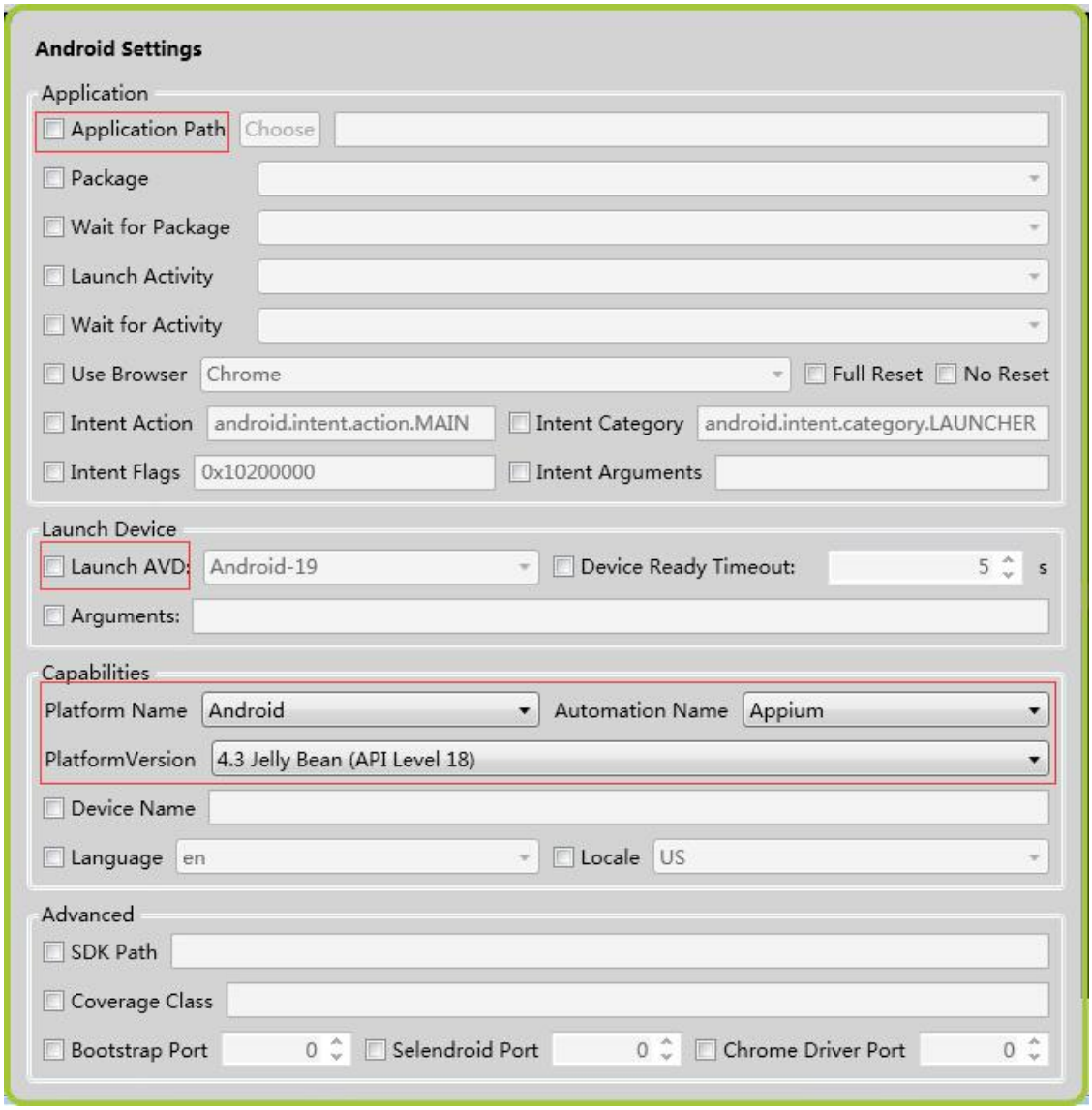
5. 启动 appium

appium 客户端安装成功，



1.1 使用 appium：Android Settings。

点击左上角的第一个机器人图标、



设置：

1.Application Path：点击 Choose 按钮后会弹出选择 Android 应用的路径选择框，选择需要测试的应用即可。

2.Launch AVD: 如果有多个模拟器，这里选择一个作为测试用的模拟器。

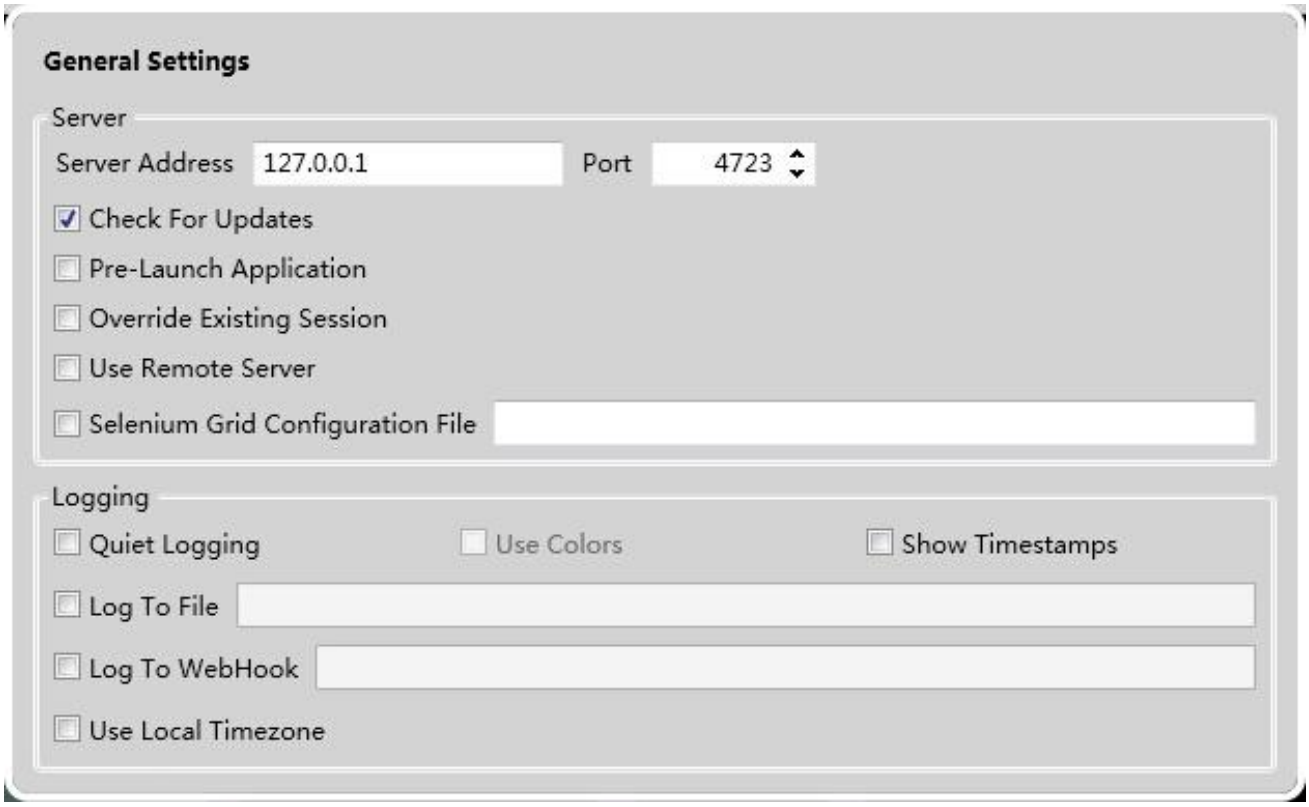
3.Platform Name:这个当然选择 Android，因为现在是在说 Android 的自动化测试。

4.Automation Name:当然选择 Appium，我们不是在玩 Appium 吗？

5.PlatformVersion:这个当然选择和模拟器中一样的版本

1.2 General Settings

点击 Appium 左上角的第二个齿轮图标则弹出 General Settings 窗口，如下图所示。



启动 appium 服务

6. 安装 Android 环境

Android 是由 Java 语言开发的，所以想开发 Android 应用首先需要 Java 环境，官网：http://www.java.com/zh_CN/download/manual.jsp（ava 环境分 JDK 和 JRE，JDK 就是 Java Development Kit.简单的说 JDK 是面向开发人员使用的 SDK，它提供了 Java 的开发环境和运行环境。JRE 是 Java Runtime Enviroment 是指 Java 的运行环境，是面向 Java 程序的使用者，而不是开发者。），选择 Windows 的 jak 安装。默认安装路径就可以。

设置环境变量：

“我的电脑”右键菜单--->属性--->高级--->环境变量--->系统变量

找到 path，加入 java 安装路径，点击保存，然后命令行输入 java 判断 java 环境配置是否成功

7. 安装android adt & SDK

官方下载地址：http://developer.android.com/sdk/index.html，官网不一定可以访问到，提供一下链接:

这里提供 adt-bundle 下载链接：

http://dl.google.com/android/adt/adt-bundle-windows-x86-20140702.zip

https://dl.google.com/android/adt/adt-bundle-windows-x86_64-20140702.zip

http://dl.google.com/android/adt/adt-bundle-mac-x86_64-20140702.zip

http://dl.google.com/android/adt/adt-bundle-linux-x86-20140702.zip

http://dl.google.com/android/adt/adt-bundle-linux-x86_64-20140702.zip

Android-sdk 下载地址：

http://dl.google.com/android/android-sdk_r23.0.2-windows.zip

http://dl.google.com/android/installer_r23.0.2-windows.exe

http://dl.google.com/android/android-sdk_r23.0.2-macosx.zip

http://dl.google.com/android/android-sdk_r23.0.2-linux.tgz

adt和sdk 都可以用来运行android 模拟器，所以下载哪个都可以。解压后，放到自己目录就可以，配置与java的类似，我的目录是：
D:\appium\android-sdk-windows

“我的电脑”右键菜单--->属性--->高级--->环境变量--->系统变量-->新建..

变量名：ANDROID_HOME

变量值：D:\android\android-sdk-windows

找到 path 变量名—>“编辑”添加：

变量名：PATH

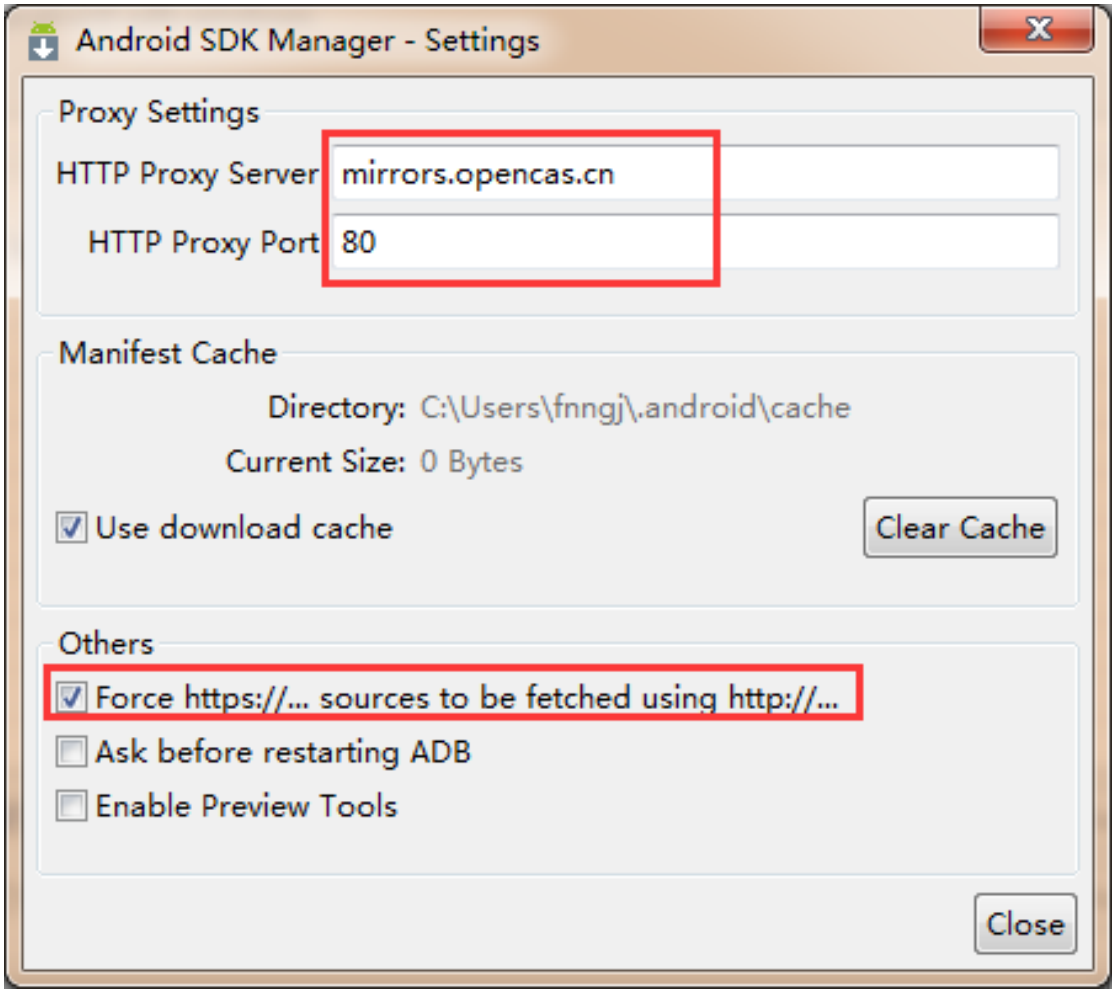
变量值：;%ANDROID_HOME%\tools;

8.SDK Manager 安装模拟器

双击启动 SDK Manager.exe 程序。更新下载最新的 sdk 就可以，不一定可以下载这里推荐一个网站：http://www.androiddevtools.cn/

它提供了国内的代理，以及各种开发工具的安装。

在 Android SDK Manager 的菜单栏上点击“Tools”---->“Options...”设置相关代码，如下图：这样设置后就可以更新



安装 SDK Platform-Tools

Android SDK Platform-tools 是版本有区别的工具文件夹，里面有 adb 、aapt、fastboot 等工具包。在 AndroidDevTools.cn 网站上找到 Android SDK Platform-tools 下载链接。

把解压出来的 platform-tools 文件夹放在 android sdk 根目录下，并把 adb 所在的目录添加到系统 PATH 路径里。

Path 中增加下面就可以

%ANDROID_HOME%\platform-tools;

点击保存后，使用 Windows 命令行 输入 adb 查看是否配置成功，成功后输入 appium-doctor，检查 Appium 环境

如果出现“All Checks were successful”的提示，证明 appium 环境配置成功。

9. 安装 python3

官网：https://www.python.org/downloads/ 下载自己想要的版本，

笔者的是 python3.5 的版本，默认安装就行，安装成功后，将 python

加入环境变量，在命令行输入 python 验证 python 环境是否成功。

安装成功后，使用命令行安装：

```
pip install Appium-Python-Client
```

安装 python 的 Appium 库。在文件使用 from appium import webdriver

成功即安装成功。

10. 下 载 HTMLTestRunner.py （ 生 成 测 试 报 告 ）。

http://download.csdn.net/detail/qq_26664581/9439036 python3 的版本。或者 http://www.cnblogs.com/sunshishi/p/4569159.html 复制下来保存为 HTMLTestRunner.py，放在安装目录的 lib 下，在文件使用 import HTMLTestRunner 成功即配置成功。

到此，环境配置成功，接下来需要解惑的是 api。（python）了解后为接下来的编写脚本做铺垫。

二：python 语言客户端库以及 api 详解（Android）

github：https://github.com/appium/python-client

github 已经把使用方法告诉大家，但是是英文的，笔者试图自己整理一下。

（1）Appium 服务关键字

1、通用：

1，automationName　自动化测试引擎 设置为：Appium（默认）或 Selendroid

2，platformName　手机操作系统 设置为：iOS, Android 或 FirefoxOS

3.platformVersion　手机操作系统版本　设置为： 7.1， 4.4

4，browserName 需要进行自动化测试的手机 web 浏览器名称。如果是对应用进行自动化测试,这个关键字的值应为空。iOS 系统上可以用 Safari ,Android 系统上可以用 Chrome、 Chromium 或 Browser

5.newCommandTimeout 设置命令超时时间，单位：秒。达到超时时间仍未接收到新的命令时 Appium 会假设客户端退出然后自动结束会话。比如 60

6.autoLaunch Appium 是否需要自动安装和启动应用。默认值 true true, false

7.language (Sim/Emu-only) 设定模拟器（ simulator / emulator ）的语言。

|如： fr

8.locale(Sim/Emu-only) 设定模拟器（ simulator / emulator ）的区域设置。|如： fr_CA

9.udid 连接的物理设备的唯一设备标识|如： 1ae203187fc012g

10.orientation (Sim/Emu-only) 在一个设定的方向模式中开始测试 LANDSCAPE (横向) 或 PORTRAIT (纵向)

11.autoWebview 直接转换到 WebView 上下文。 默认值 false、true, false

12.noReset 不要在会话前重置应用状态。默认值 false。true, false

13.fullReset(iOS) 删除整个模拟器目录。(Android) 通过卸载——而不是清空数据——来重置应用状态。在 Android 上，这也会在会话结束后自动清除被测应用。默认值 false|true, false

2、**Android特有**

1. appActivity 你要从你的应用包中启动的 Android Activity 名称。它通常需要在前面添加 （如使用.MainActivity 而不是 MainActivity) MainActivity, .Settings|

2.appPackage 你想运行的 Android 应用的包名 | 比如 com.example.android.myApp, com.android.settings

3.appWaitActivi 你想要等待启动的 Android Activity 名称 SplashActivity

4.deviceReadyTimeout 设置等待一个模拟器或真机准备就绪的超时时间 5

5，androidCoverage 用于执行测试的 instrumentation 类。作为命令 adb shell am instrument -e coverage true -w 的 -w 参数。com.my.Pkg/com.my.Pkg.instrumentation.MyInstrumentation

enablePerformanceLogging (仅适用于 Chrome 和 webview) 开启 6.Chromedriver 的性能日志。 (默认 false) true, false

7.androidDeviceReadyTimeout 等待设备在启动应用后准备就绪的超时时间。以秒为单位。如 30

8.androidDeviceSocket 开发工具的 socket 名称。只有在被测应用是一个使用 Chromium 内核的浏览器时需要。socket 会被浏览器打开，然后 Chromedriver 把它作为开发者工具来进行连接。如 chrome_devtools_remote

9.avd 需要启动的 AVD（安卓虚拟设备）名称。如 api19
10，avdLaunchTimeout 以毫秒为单位，等待 AVD 启动并连接到 ADB 的超时时间。(默认值 120000)300000
11、avdReadyTimeout 以毫秒为单位，等待 AVD 完成启动动画的超时时间。(默认值 120000) 300000
12.avdArgs 启动 AVD 时需要加入的额外的参数。如 -netfast
13.useKeystore 使用一个自定义的 keystore 来对 apk 进行重签名。默认值 false ，true or false
14.keystorePath 自定义 keystore 的路径。默认：~/.android/debug.keystore|如 /path/to.keystore
15.keystorePassword 自定义 keystore 的密码。如 foo
16.keyAliaskey 的别名 |如 androiddebugkey
17.keyPassword key 的密码 如 foo
18.chromedriverExecutable webdriver 可执行文件的绝对路径(如果 Chromium 核心提供了对应的 webdriver， 应该用它代替 Appium 自带的 webdriver) /abs/path/to/webdriver
19.autoWebviewTimeout 以毫秒为单位，等待 Webview 上下文激活的时间。默认值 2000 如 4
20.intentAction 用于启动 activity 的 intent action。(默认值 android.intent.action.MAIN) 如 android.intent.action.MAIN, android.intent.action.VIEW
21.intentCategory 用于启动 activity 的 intent category。(默认值 android.intent.category.LAUNCHER | 如 android.intent.category.LAUNCHER,
android.intent.category.APP_CONTACTS
22.intentFlags 用于启动 activity 的标识（flags）(默认值 0x10200000) | 如 0x10200000
23.optionalIntentArguments 用于启动 activity 的额外 intent 参数。请查看 [Intent 参数](http://developer.android.com/tools/help/adb.html#IntentSpec) | 如 --esn <EXTRA_KEY>, --ez <EXTRA_KEY> <EXTRA_BOOLEAN_VALUE>

24.stopAppOnReset 在使用 adb 启动应用前停止被测应用的进程（process）。如果被测应用是被另一个应用创建的，当这个参数被设定为 false 时，允许另一个应用的进程在使用 adb 启动被测应用时继续存活。默认值 true|，true 或 false

25.unicodeKeyboard 使用 Unicode 输入法。默认值 false，可设置为 true 或 false

26.resetKeyboard 在设定了 unicodeKeyboard 关键字的 Unicode 测试结束后，重置输入法到原有状态。如果单独使用，将会被忽略。默认值 false| true 或 false

27.noSign 跳过检查和对应用进行 debug 签名的步骤。只能在使用 UiAutomator 时使用，使用 selendroid 是不行。默认值 false 可设置为 true 或 false

28.ignoreUnimportantViews 调用 uiautomator 的函数 setCompressedLayoutHierarchy()。由于 Accessibility 命令在忽略部分元素的情况下执行速度会加快，这个关键字能加快测试执行的速度。被忽略的元素将不能够被找到，因此这个关键字同时也被实现成可以随时改变的 *设置（settings）* 。默认值 false 可设置为 true 或 false

例子（安卓）：

```
from appium import webdriver
desired_caps = {}
desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '4.2'
desired_caps['deviceName'] = 'Android Emulator'
desired_caps['app'] =
PATH('../../../apps/selendroid-test-app.apk')
self.driver =
webdriver.Remote('http://localhost:4723/wd/hub',
desired_caps)
```

文中提到的比较多，大家可以记住一些常用的就可以，其他的作为了解。

（2）api（python）

1.定位：

A.利用 Android UIAutomator定位：

例： driver.find_element_by_android_uiautomator('new UiSelector().description("Animation")')

B.利用Accessibility ID定位

例：

```
driver.find_element_by_accessibility_id('Animation')
```

C. 利用 id定位：

例： driver.find_element_by_id(‘login’)

D 利用 Xpath定位

例： driver.find_element_by_xpath(xpath)

E:利用name定位

例： driver.find_element_by_name('Touch Paint')

F：利用 classname 定位

例： driver.find_elements_by_class_name(’listView’)

2.锁定屏幕 driver.lock(5)

3.把当前应用放到后台去 driver.background_app(5)

4. 长按住键盘 driver.long_press_keycode(keycode)

5.检查应用是否已经安装 driver.is_app_installed('com.example.android.apis')

6.安装应用到设备中去 driver.install_app('path/to/my.apk')

7.从设备中删除一个应用 driver.remove_app('com.example.android.apis')

8.模拟设备摇晃 driver.shake()

9.关闭应用 driver.close_app()

10.启动应用 driver.launch_app()

11.应用重置 driver.reset()

12.列出所有的可用上下文 driver.contexts

13.列出当前上下文 driver.current_context

14.将上下文切换到默认上下文 driver.switch_to.context(None)

15. 截图 driver.get_screenshot_as_file(filename)

16.发送一个按键事件给设备 driver.keyevent(176)

17.Android only 得到当前 activity driver.current_activity

18.生成触摸动作的接口。这部分文档很快将会补充更多的内容进来

```
action = TouchAction(driver)
action.press(element=el, x=10, y=10).release().perform()
```

19.模拟用户滑动 driver.swipe(75, 500, 75, 0, 1000)

20.在 0% 到 100% 内双指缩放屏幕 driver.pinch(element=el)

21.放大屏幕 在 100% 以上放大屏幕 driver.zoom(element=el)

22.从设备中拉出文件 driver.pull_file('Library/AddressBook/AddressBook.sqlitedb')

23.推送文件到设备中去

```
data = "some data for the file"
path = "/data/local/tmp/file.txt"
```

driver.push_file(path, data.encode('base64'))
24.隐藏键盘 driver.hide_keyboard()
25.安装 appdriver.install_app(path)
26.卸载 appdriver.remove_app(app_id)
27.打印当前 activitydriver.current_activity
28.振动 driver.shake()
29.打开通知栏(api 18 以上)driver.open_notifications()
30.获取网络 driver.network_connection
31.获取手机屏幕分辨率 driver.get_window_size()
32.设置屏幕分辨率 driver.set_window_size(width,height)
33.获取当前坐标位置 driver.get_window_position()
34.开关定位服务 driver.toggle_location_services()
35.开关定位服务 driver.toggle_location_services()
36. 关闭 close
37.退出关闭所有连接 quit
38. 获取当前页面源 page_source
39.获取当前页面网页 current_url
4.获取元素左上角坐标 location
41 获取元素大小 size
42. 元素是否可用 is_enabled（）返回 True of False
43.元素是否可选择 is_selected（）
44.清除 clear
45.点击 click
46.设置经纬度 用法 driver.set_location(纬度，经度，高度)
47.返回当前输入法包名 active_ime_engine
48.关闭当前输入法 deactivate_ime_engine
49.激活输入法 activate_ime_engine
用法
driver.activate_ime_engine(“com.android.inputmethod.latin/.LatinIME”)
50. s_ime_active

检查设备是否有输入法服务活动。返回真/假。 安卓 用法
print(driver.is_ime_active())
51.返回可用输入法 available_ime_engines
52.设置网络 set_network_connection
先加载 from appium.webdriver.connectiontype import ConnectionType
dr.set_network_connection(ConnectionType.WIFI_ONLY)
ConnectionType 的类型有 NO_CONNECTION = 0 AIRPLANE_MODE = 1 WIFI_ONLY = 2 DATA_ONLY = 4 ALL_NETWORK_ON = 6
53.返回网络类型 network_connection
54.值后台 background_app
用法 driver.background_app(5) 置后台 5 秒后再运行
55. 每隔几秒获取一次当前的 activity 返回的 True 或 False
driver.wait_activity('.activity.xxx',5,2)
56. 取消执行该动作 cancel()

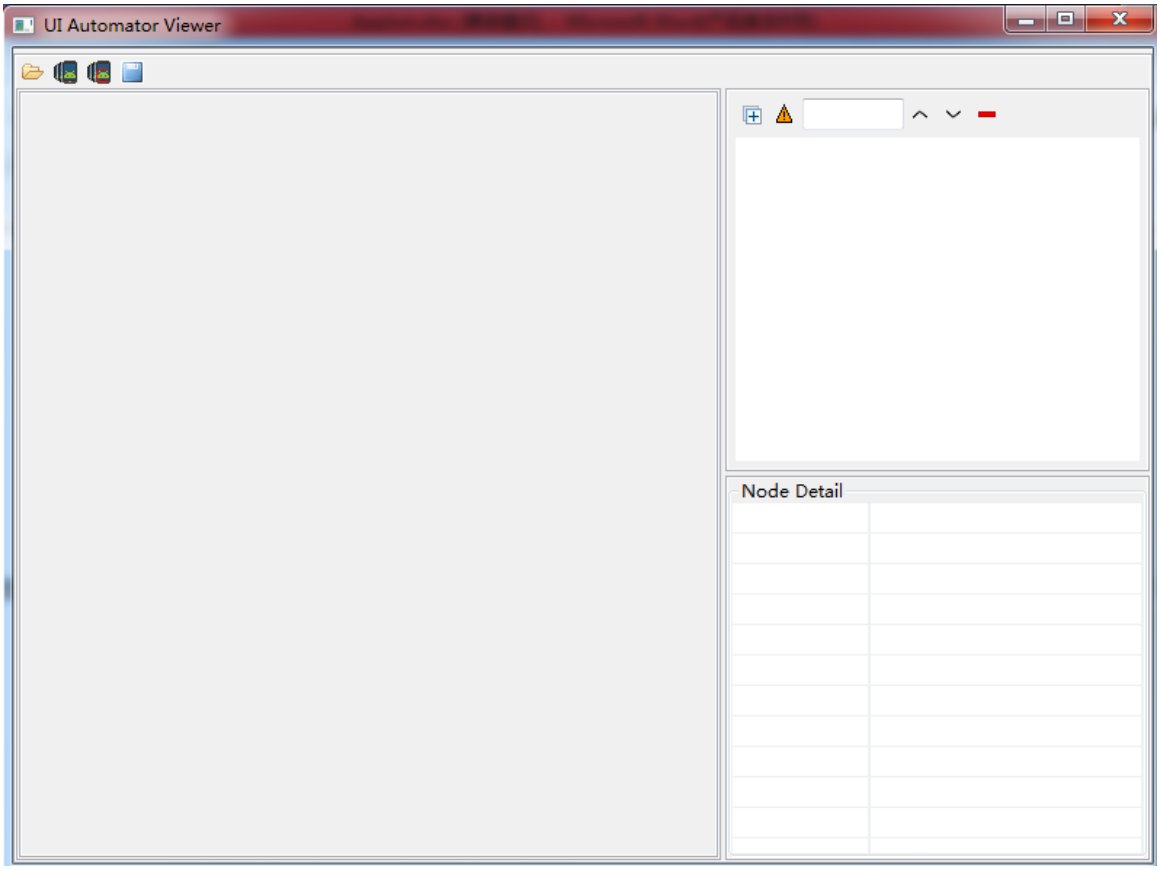
（3）Appium 服务器参数

1.进入 REPL 模式 --shell
2.不输出具体日志 -q, --quiet
3.连接的物理实体机的 udid -U, --udid
4.监听的 ip 地址 -a, --address 默认 0.0.0.0
5.监听端口 -p ，-port 默认 4723
6.(Android-only) 连接设备的端口号 -dp,--devices-p 默认 4724
7.允许 session 覆盖 (冲突的话) --session-override
8.将日志输出到指定文件 -g, --log
9.在终端输出里显示时间戳 --log-timestamp
10.不在终端输出中显示颜色 --log-no-colors
11.同时发送日志到 HTTP 监听器 -G, --webhook
12.(Android-only) 你要运行的 apk 的 java 包 --app-pkg
13.(Android-only) 打开应用时，启动的 Activity 的名字 --app-activity
14.(Android-only) 你想等待的 Activity 的 包名

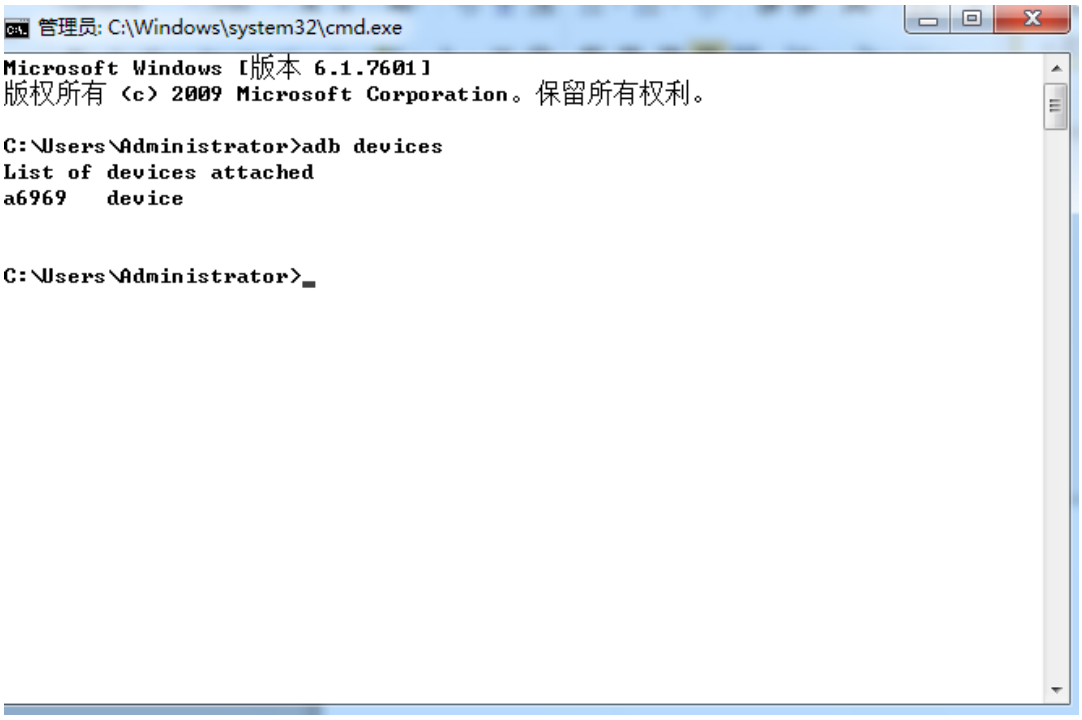
--app-wait-package
15.(Android-only) 你想等待的 Activity 名字 --app-wait-activity
16.要启动的 avd 的名字—avd
17.(Android-only) 等待设备准备好的时间，以秒为单位 --device-ready-timeout 默认 5
18.指定 JSON 格式的配置文件 ，用来在 selenium grid 里注册 appium －nodeconfig
19.robot 的 ip 地址-ra, --robot-address 默认 0.0.0.0
20.robot 端口-rp, --robot-port 默认-1
21.用来和 Selendroid 交互的本地端口 --selendroid-port 默认 8080
22.ChromeDriver 运行的端口 --chromedriver-port 默认 9515
23.Android-only) 设置签名 apk 的 keystore --use-keystore
24.(Android-only) keystore 的路径 --keystore-path 默认 /Users/user/.android/debug.keystore
25.(Android-only) keystore 的密码 --keystore-password 默认：android
26.(Android-only) Key 的别名
--key-alias 默认：androiddebugkey
27.(Android-only) Key 的密码--key-password 默认：android
28.打印 Appium 服务器的配置信息，然后退出
--show-config

三、**元素定位**（Android）。

Uiautomatorview、hierarchyviewr 和 Inspector，Uiautomatorviewer 是安卓 sdk 自带，根据个人习惯，本人以 Uiautomatorviewer 为例，使用方式差不多。在 tool 文件下。双击 uiautomatorviewer.bat 启动。

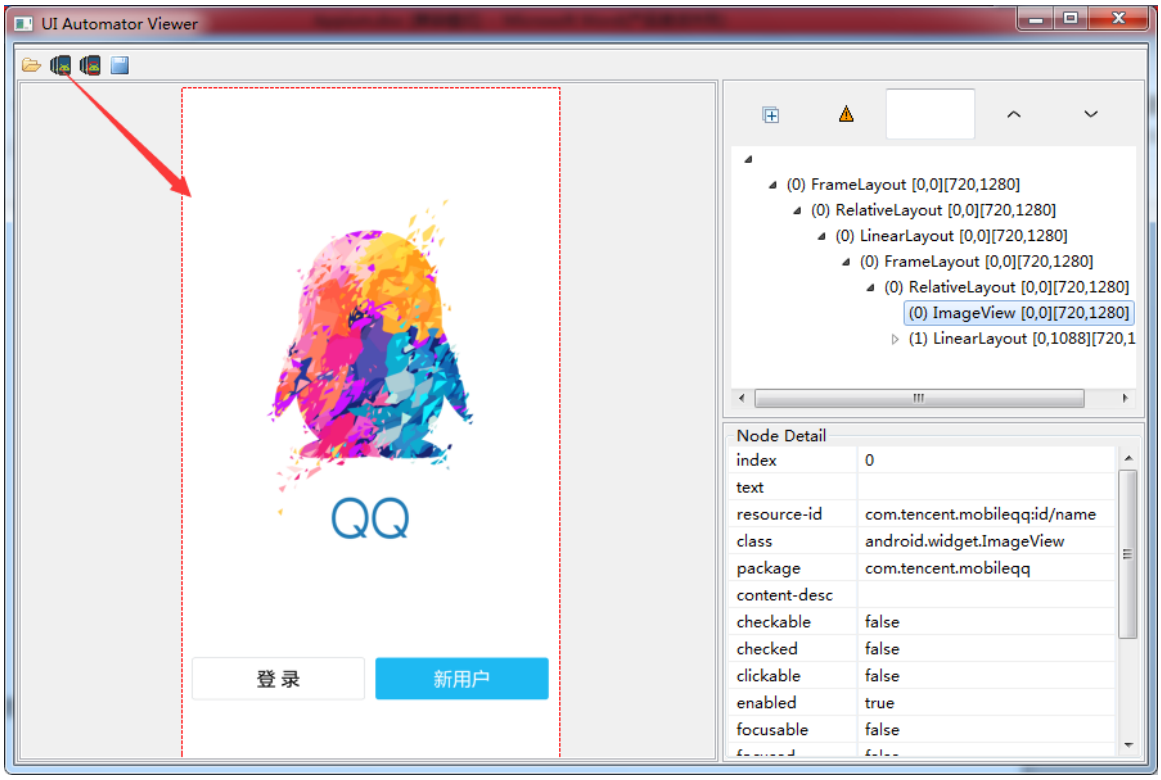


连接手机。cmd 输入 adb devices，如图，证明连接成功。

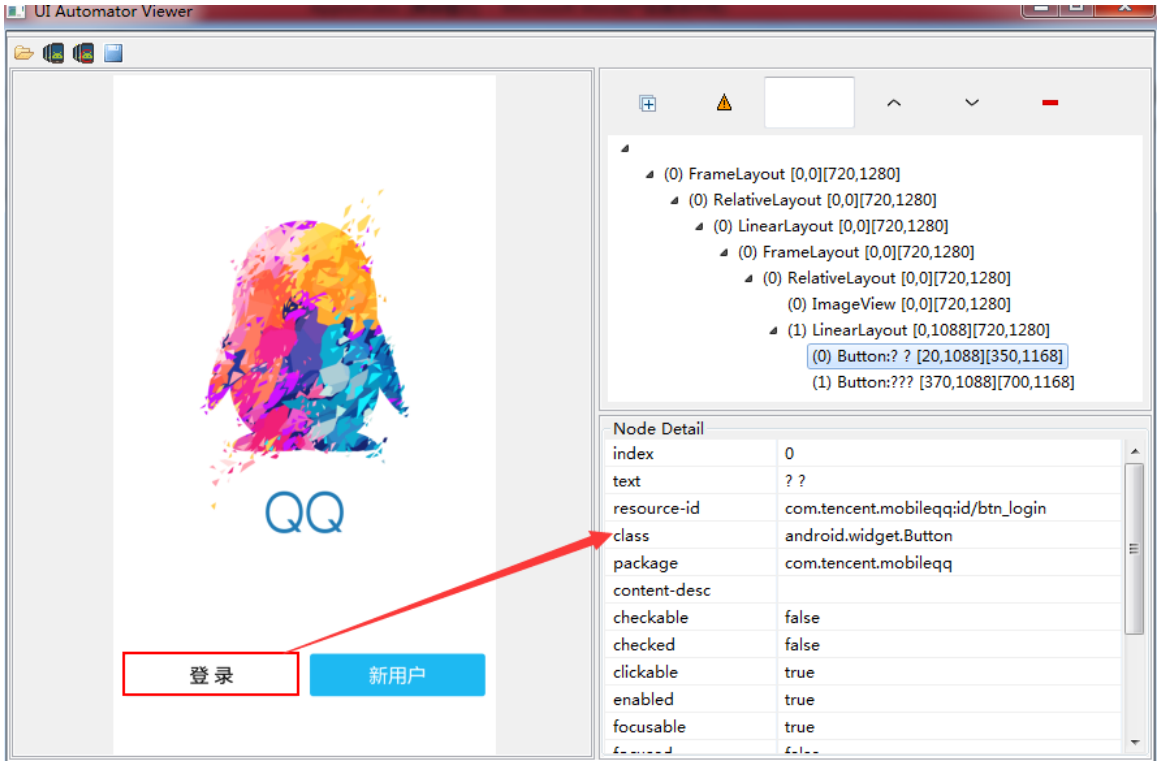


在手机上点开一个应用（qq 为例）：

点击 device screeshot 后。如图。



稍后点击界面上的登录后，就可以根据 Node Detail 内容来定位元素



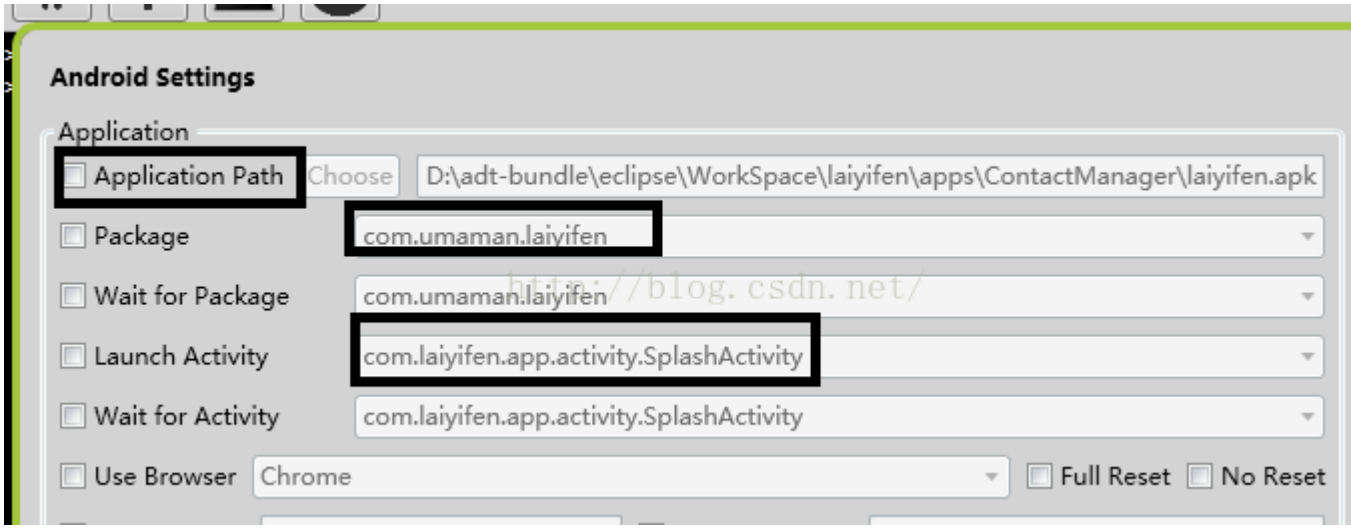
App 包名：使用 APK helper 查看

Activiy 名称：

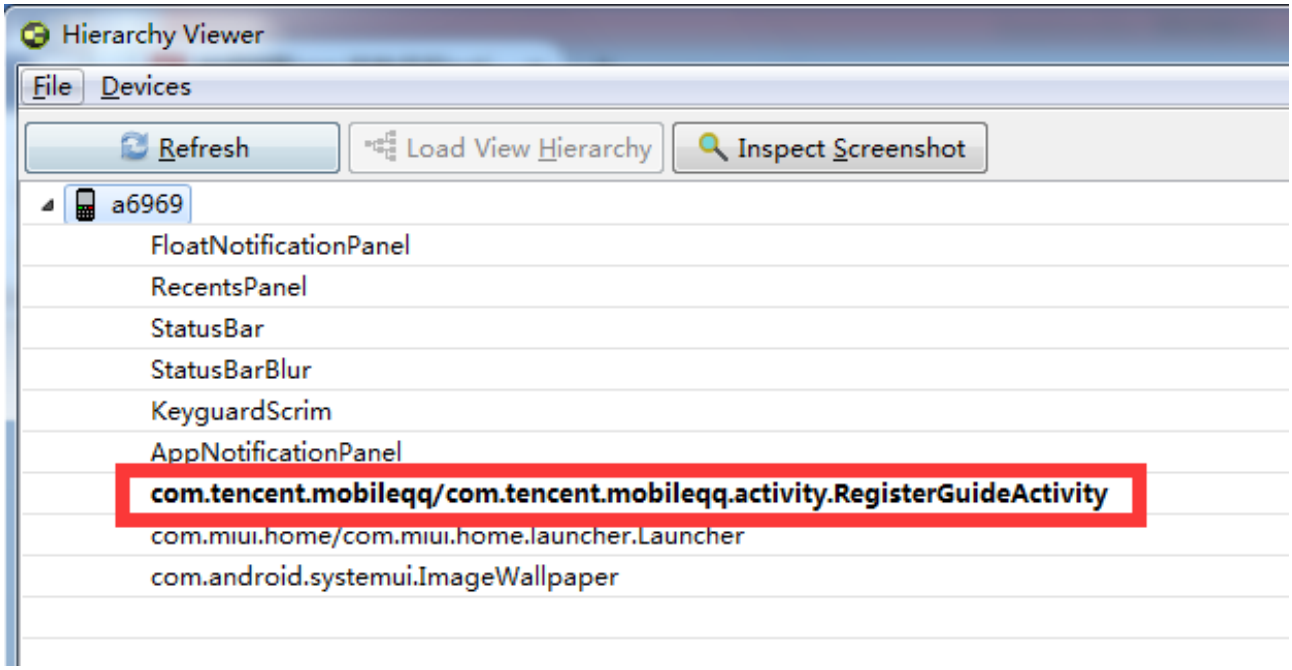
方法一：用重签名工具，

方法二：命令行：Aapt dumpbadging xxxx.apk

方法三：appium 设置页选择包后会显示。



方法四：hierarchyviewer 查看



方法五：

Adbshell dumpsys activity activities >d:\log.txt

之后去 log.txt 搜 package 和 Activity

方法六： adb logcat >d:\log.txt

之后去 log.txt 搜 package 和 Activity

那么到此，环境搭建，api，元素定位等都有所了解，那么我们接下来就是编写脚本来。下面给大家看一个脚本

```
from appium import webdriver#倒入 webdriver
import time,unittest,HTMLTestRunner#倒入库
class Testlogin(unittest.TestCase):
    def setUp(self):#初始化
        self.desired_caps={}
        self.desired_caps['platformName'] = 'Android'
        self.desired_caps['deviceName']='a6969'
        self.desired_caps['preformVersion']='5.0.2'
        self.desired_caps['appPackage']                    =
'com.tencent.mobileqq'
        self.desired_caps['appActivity']
='.activity.SplashActivity'

         self.driver=webdriver.Remote('http://localhost:472
         3/wd/hub', self.desired_caps)#启动 app
         time.sleep(2)
    def tearDown(self):#还原测试环境

        self.driver.find_element_by_id('com.tencent.mobile
        qq:id/conversation_head').click()

        self.driver.find_element_by_id('com.tencent.mobile
        qq:id/settings').click()

        self.driver.find_element_by_id('com.tencent.mobile
        qq:id/account_switch').click()

        self.driver.find_element_by_id('com.tencent.mobile
        qq:id/logoutBtn').click()
```

```
        self.driver.find_element_by_id('com.tencent.mobile
        qq:id/dialogRightBtn').click()
       self.driver.quit()
def testLogin1(self):#测试用例

    self.driver.find_element_by_id('com.tencent.mobileqq:
    id/btn_login').click()#登录，定位方式 id

      time.sleep(2)

       me=self.driver.find_element_by_android_uiautomator
       ('new UiSelector().text("QQ 号/手机号/邮箱")')#定位
       输入 qq 号，使用 uiautomator 定位
      me.clear()#输入框输入前最好先清空下
      me.send_keys('319197149')

       password=self.driver.find_element_by_id('com.tence
       nt.mobileqq:id/password')
      password.clear()
      password.send_keys('lileilei.930423')

     self.driver.find_element_by_id('com.tencent.mobileqq
     :id/login').click()#点击登录

     m=self.driver.find_element_by_id('com.tencent.mobile
     qq:id/conversation_head')
      if m is not None:
          print('login is sucess')
      else:
          print('login is Flase')
```

```
            print(self.driver.find_element_by_id('com.tence
            nt.mobileqq:id/dialogText').text)
if __name__ == '__main__':
    suiteTest = unittest.TestSuite()
    suiteTest.addTest(Testlogin("testLogin1"))
   now=time.strftime('%Y-%m%d',time.localtime(time.time()))
        report_dir= r'%s.html'%now
        re_open= open(report_dir,'wb')

    runner=HTMLTestRunner.HTMLTestRunner(stream=re_open,tit
    le='QQ 测试',description='测试结果')
        runner.run(suiteTest)
```

这是一个完整简单的测试脚本，最后生成测试报告。到此一个完整的测试就完成来，

（请大家遵守劳动成果。可以微信打赏下在下，如果有疑问可以加我 qq：952943386，或者 email：leileili126@163.com，本人 qq 群：194704520，初次学习，菜鸟一枚。感谢大家翻阅）