

Appium 自动化(图文教程)

作者：上海-悠悠



更多自动化（selenium、python 接口）扫码关注

个人微信公众号：yoyoketang (扫二维码关注)

个人博客地址： <http://www.cnblogs.com/yoyoketang/>

QQ 交流群：512200893

目录

第 1 章 环境搭建.....	5
1.1 android-sdk 环境	5
1.1.1 环境准备.....	5
1.1.2 jdk 安装.....	6
1.1.3 android-sdk 下载安装	8
1.1.4 android-sdk 环境变量	9
1.1.5 adb 环境.....	10
1.1.6 连接手机.....	11
1.2 appium 环境	12
1.2.1 必备软件安装.....	12
1.2.1 Python 安装	13
1.2.3 安装 node.js.....	13
1.2.4 安装 appium	15
1.2.5 安装.net framework.....	16
1.2.6 appium-doctor	17
第 2 章 API 详解	18
2.1 启动 app	18
2.1.1 下载 aapt	19
2.1.2 获取 apk 包名	21
2.1.3 获取 launcherActivity	21
2.1.4 写脚本.....	22
2.1.5 运行 appium	23
2.1.6 最终代码如下.....	25
2.2 元素定位 uiautomatorviewer	26
2.2.1 启动 uiautomatorviewer.bat.....	26
2.2.2 连接手机.....	28
2.2.3 定位元素.....	29
2.2.4 点搜索框.....	30
2.2.5 参考代码.....	30
2.2.6 元素定位.....	31
2.3 Appium Inspector	32
2.3.1 设置 appium	33
2.3.2 开启 appium	34
2.3.3 Inspector Window	34
2.3.4 查看属性.....	35
2.5.5 常见异常.....	35
2.4 Remote 远程控制	36
2.4.1 设置 IP	36
2.4.2 访问地址.....	37
2.4.3 配置测试机.....	38
2.4.4 远程操作.....	39
2.5 输入中文.....	40
2.5.1 定位搜索.....	40

2.5.2 运行脚本.....	40
2.5.3 屏蔽软键盘.....	41
2.5.4 输入中文字符.....	42
2.5.5 还原设置.....	44
2.5.6 最终脚本如下.....	44
2.6 Appium API.....	45
1.contexts.....	46
2. current_context.....	46
3. context	46
4. find_element_by_ios_automation.....	47
5. find_element_by_accessibility_id.....	47
6.scroll.....	47
7. drag_and_drop.....	48
8.tap	48
9. swipe	49
10.flick.....	49
11.pinch.....	50
12.zoom.....	50
13.reset	51
15. keyevent	52
16. press_keycode.....	52
17. long_press_keycode.....	52
19. wait_activity.....	53
20. background_app	54
21.is_app_installed	54
22.install_app.....	54
23.remove_app.....	55
24.launch_app.....	55
25.close_app	55
26. start_activity	56
27.lock.....	57
29.open_notifications	57
30.network_connection.....	57
31. set_network_connection	58
32. available_ime_engines.....	59
33.is_ime_active	59
34.activate_ime_engine.....	59
35.deactivate_ime_engine.....	60
36.active_ime_engine.....	60
37. toggle_location_services.....	60
38.set_location.....	61
39.tag_name	61
40.text	61
41.click	62

42.submit	62
43.clear.....	62
44.get_attribute	62
45.is_selected.....	64
46.is_enabled.....	64
47.find_element_by_id.....	64
48. find_elements_by_id	65
49. find_element_by_name.....	65
50. find_elements_by_name	65
51. find_element_by_link_text	66
52. find_elements_by_link_text	66
53. find_element_by_partial_link_text.....	66
54. find_elements_by_partial_link_text.....	67
55. find_element_by_tag_name.....	67
56. find_elements_by_tag_name	67
57. find_element_by_xpath.....	68
58. find_elements_by_xpath	68
59. find_element_by_class_name	69
60. find_elements_by_class_name.....	69
61. find_element_by_css_selector	70
62.send_keys.....	70
63. is_displayed.....	71
64. location_once_scrolled_into_view	71
65.size.....	72
66. value_of_css_property.....	72
67.location	72
68.rect	73
69. screenshot_as_base64.....	73
70.execute_script.....	73
71.execute_async_script.....	74
72.current_url	74
73. page_source.....	74
74.close	75
75.quit.....	75

第 1 章 环境搭建

1.1 android-sdk 环境

前言

appium 可以说是做 app 最火的一个自动化框架，它的主要优势是支持 android 和 ios，另外脚本语言也是支持 java 和 Python。小编擅长 Python，所以接下来的教程是 appium+python 的实例。

学习 appium 最大的难处在于环境的安装，80%的人死于环境安装，然后就没然后了，10%的人被环境折腾一周以上，只有剩下的 10%人品好，可以顺利安装。

1.1.1 环境准备

小编的环境是 Windows 7 版本 64 位系统（32 位的同学自己想办法哦）

1. jdk1.6.0 （64 位）
2. android-sdk_r24.3.4-windows
3. python:2.7 （3.6 也可以）
4. appium: 1.4.13.1
5. Node.js: node-v4.4.7-x64
6. Appium-Python-Client

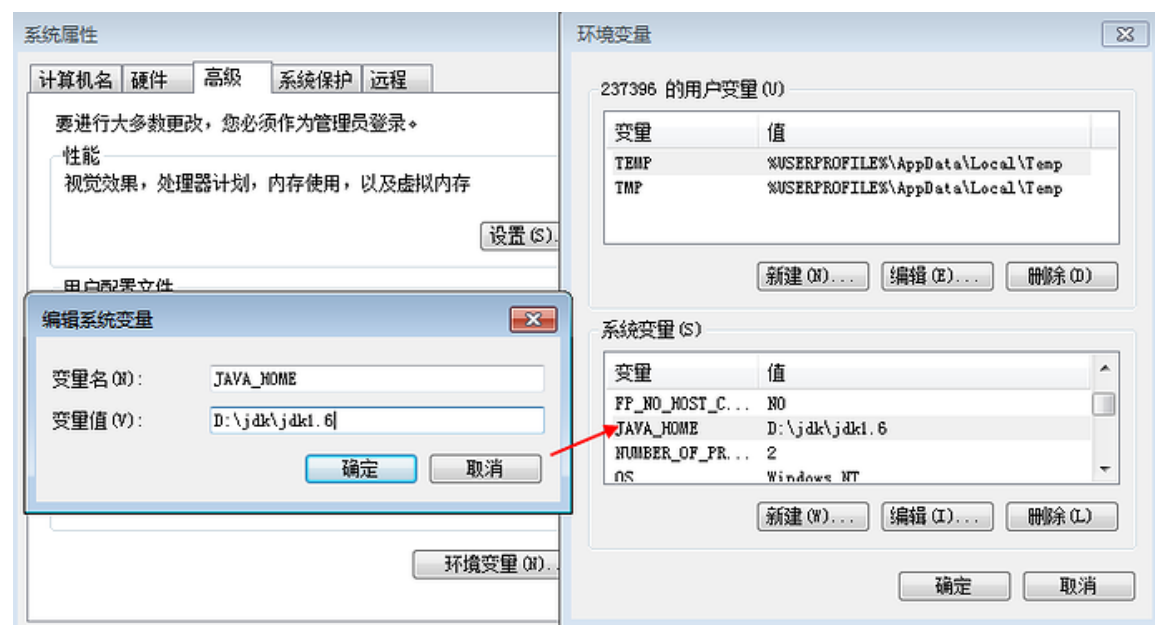
1.1.2 jdk 安装

1. 下载 jdk 包, 小编的是 64 位 1.6 版本, 其它高级版本应该也是可以的。
根据自己的系统选择对应版本

2. 一路傻瓜式安装, 注意安装路径不要有空格, 不要有中文。jdk 和 jre
不要放在一个文件夹下

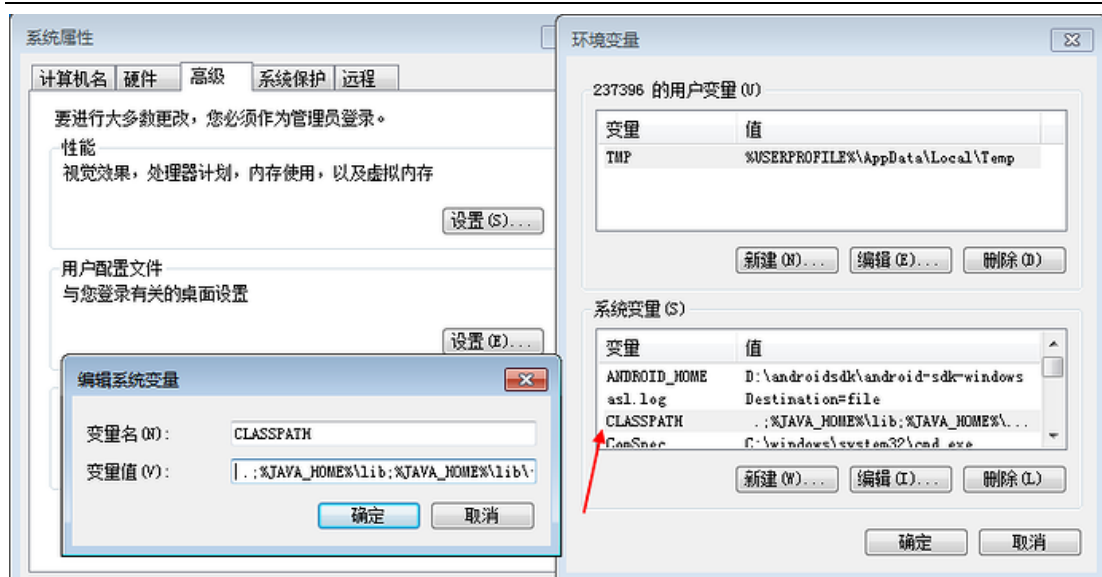
3. 设置三个环境变量, 我的电脑>选择“属性”->“高级”->“环境
变量”->“系统变量”->“新建”

JAVA_HOME----D:\Java\jdk1.6.0” （根据自己安装路径填写）



1471426392934219.png746x404 18.4 KB

CLASSPATH--- .;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar;



1471426406862959.png783x404 19.6 KB

PATH-----;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;

在 path 路径下加上面那两个，这里就不多说了

4. 打开 cmd 验证是否安装成功，输入 `java -version`，然后输入 `javac` 能显示版本号和下面的帮助信息说明安装成功

```

C:\Users\...>java -version
java version "1.6.0_39"
Java(TM) SE Runtime Environment (build 1.6.0_39-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.14-b01, mixed mode)

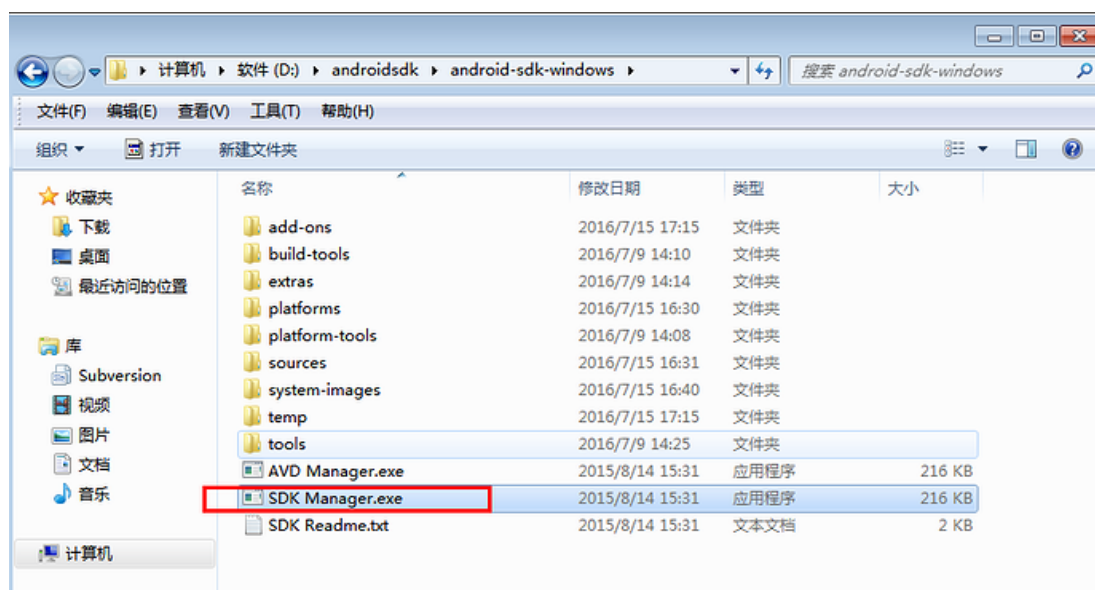
C:\Users\...>javac
用法: javac <选项> <源文件>
其中, 可能的选项包括:
-g 生成所有调试信息
-g:none 不生成任何调试信息
-g:<lines,vars,source> 只生成某些调试信息
-nowarn 不生成任何警告
-verbose 输出有关编译器正在执行的操作的消息
-deprecation 输出使用已过时的 API 的源位置
-classpath <路径> 指定查找用户类文件和注释处理程序的位置
-cp <路径> 指定查找用户类文件和注释处理程序的位置
-sourcepath <路径> 指定查找输入源文件的位置
-bootclasspath <路径> 覆盖引导类文件的位置
-extdirs <目录> 覆盖安装的扩展目录的位置
-endorseddirs <目录> 覆盖签名的标准路径的位置
半:

```

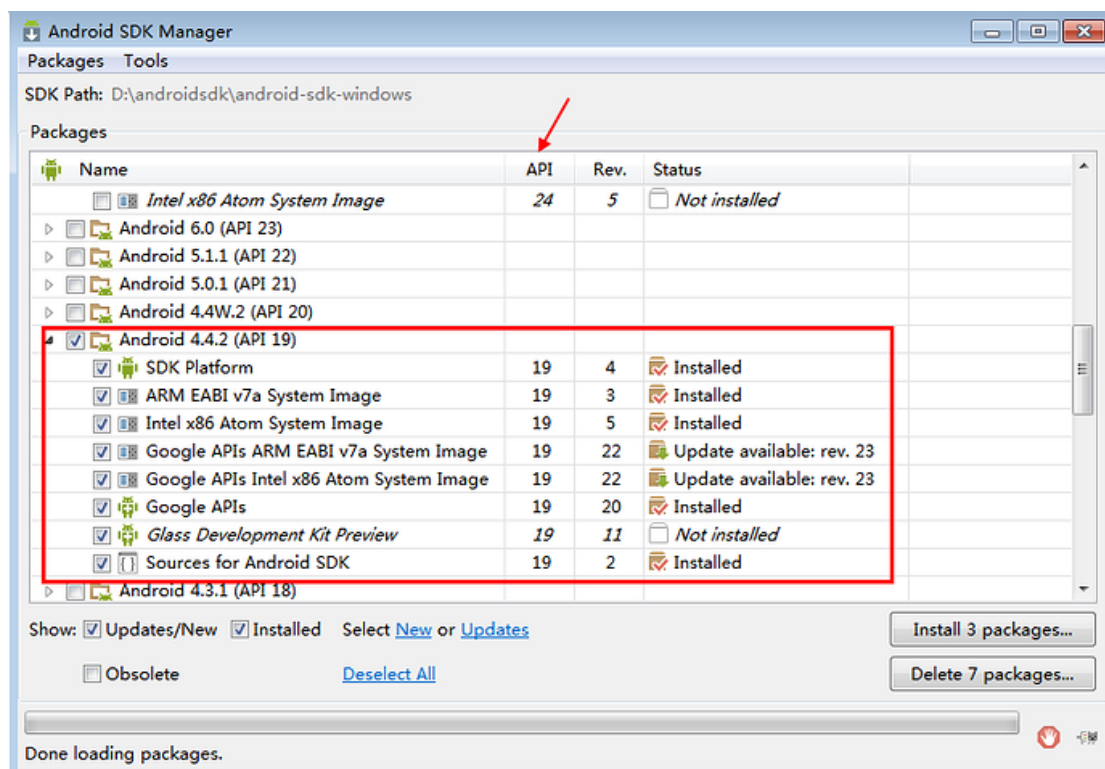
1.1.3 android-sdk 下载安装

1. 下载 android-sdk, 这个是做 android 测试和开发的必备环境, 如果不会下载的话, 在 QQ 群: 512200893 群文件下载

2. 解压后, 里面有个 SDK manager.exe 双击打开

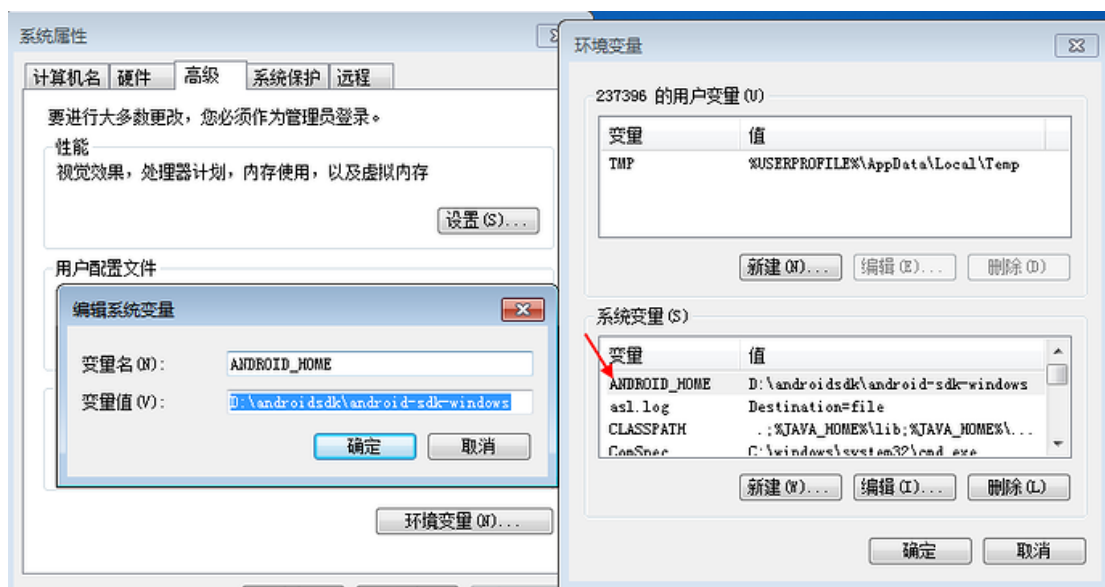


3. 勾选你要现在的 API 版本和对应的 android 版本, 后面模拟器会用到, 然后坐等下载

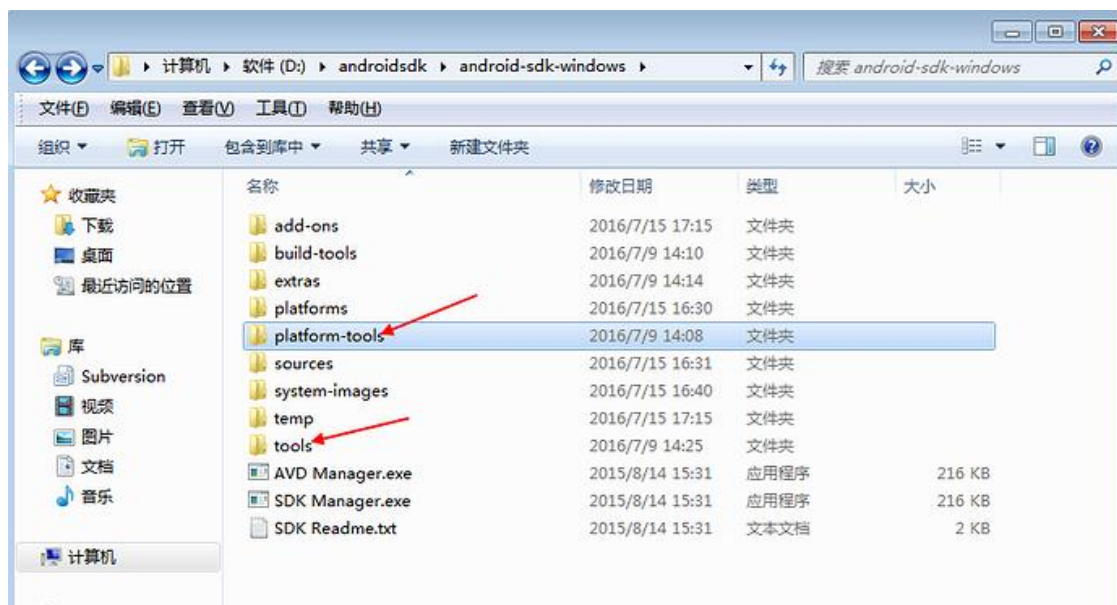


1.1.4 android-sdk 环境变量

1. 在系统变量新建：ANDROID_HOME，对应变量为：
D:\androidsdk\android-sdk-windows（sdk 安装路径）



2. path 添加两个变量，将以下箭头所指的两个文件路径添加到 path 里

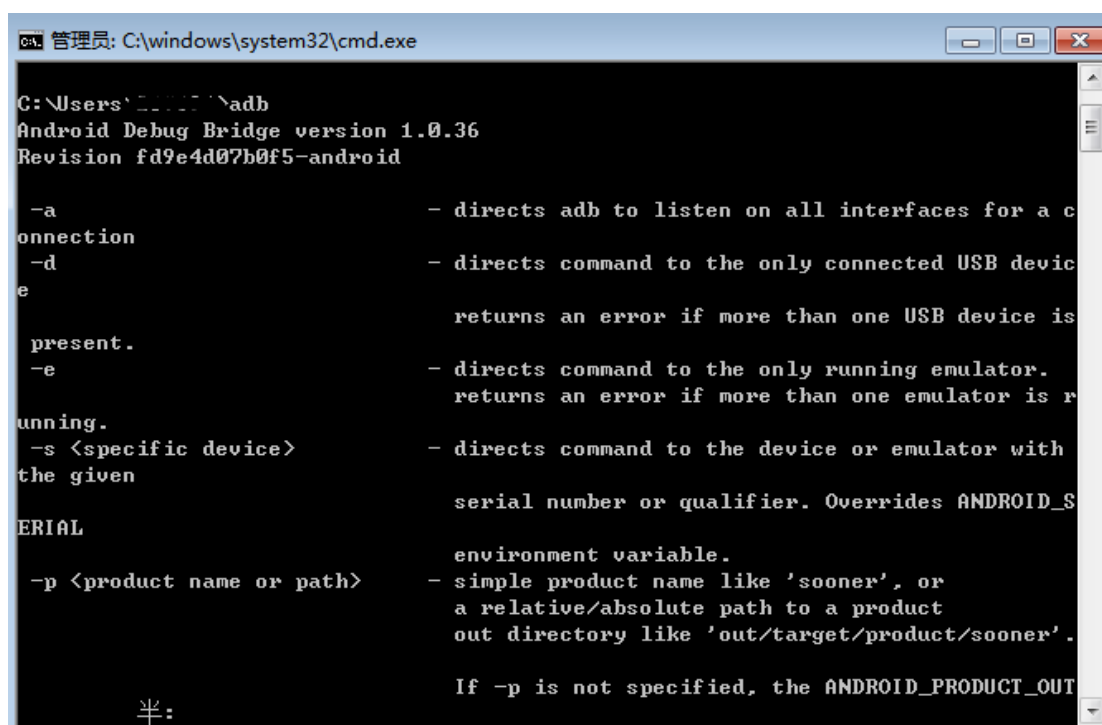


3. path 里面添加 D:\androidsdk\android-sdk-windows\tools 和 D:\androidsdk\android-sdk-windows\platform-tools

1.1.5 adb 环境

1. 因为 adb 是在 D:\androidsdk\android-sdk-windows\platform-tools 这个目录下的，所以上面添加了环境变量后，可以直接在 cmd 里面运行了。

2. 在 cmd 输入 adb 可以查看对应版本号



```
CA 管理员: C:\windows\system32\cmd.exe

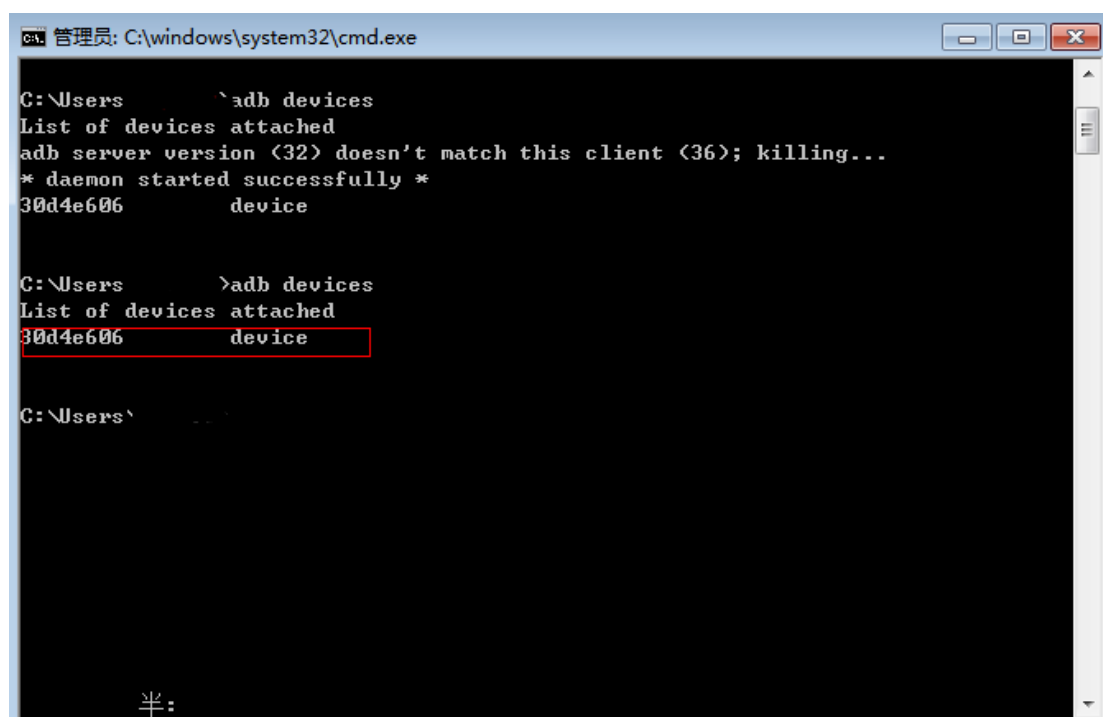
C:\Users\...>adb
Android Debug Bridge version 1.0.36
Revision fd9e4d07b0f5-android

-a                - directs adb to listen on all interfaces for a c
onnection
-d                - directs command to the only connected USB devic
e
                  returns an error if more than one USB device is
present.
-e                - directs command to the only running emulator.
                  returns an error if more than one emulator is r
unning.
-s <specific device> - directs command to the device or emulator with
the given
                  serial number or qualifier. Overrides ANDROID_S
ERIAL
                  environment variable.
-p <product name or path> - simple product name like 'sooner', or
                  a relative/absolute path to a product
out directory like 'out/target/product/sooner'.
                  If -p is not specified, the ANDROID_PRODUCT_OUT
```

1.1.6 连接手机

1. 手机用数据线连电脑，如果安装了 91 助手或者 360 什么的可以先下载手机驱动，确认手机能连上

2. 打开 cmd 输入:adb devices, 当屏幕上出现一串字符, 后面显示 devices 说明连接成功（出现其它的提示，得检查自己的环境了）



```
C:\Users\...>adb devices
List of devices attached
adb server version (32) doesn't match this client (36); killing...
* daemon started successfully *
30d4e606        device

C:\Users\...>adb devices
List of devices attached
30d4e606        device

C:\Users\...>
```

到这里 android 的测试开发环境已经装好了, 下一篇会教搭建搭建 appium 环境。安装过程中遇到各种奇葩问题, 请卸载完后, 仔细阅读, 从第一行开始, 一步一步走下来, 中间任何一个环境出问题, 都会导致最后 appium 运行失败。

1.2 appium 环境

前言 上一篇 android 测试开发环境已经准备好, 接下来就是 appium 的环境安装了。环境安装过程中切勿浮躁, 按照步骤一个个来。

环境装好后, 可以用真机连电脑, 也可以用 android-sdk 里面的模拟器(当然这个模拟器不是很好用), 我一般喜欢真机, 真机比较快。

1.2.1 必备软件安装

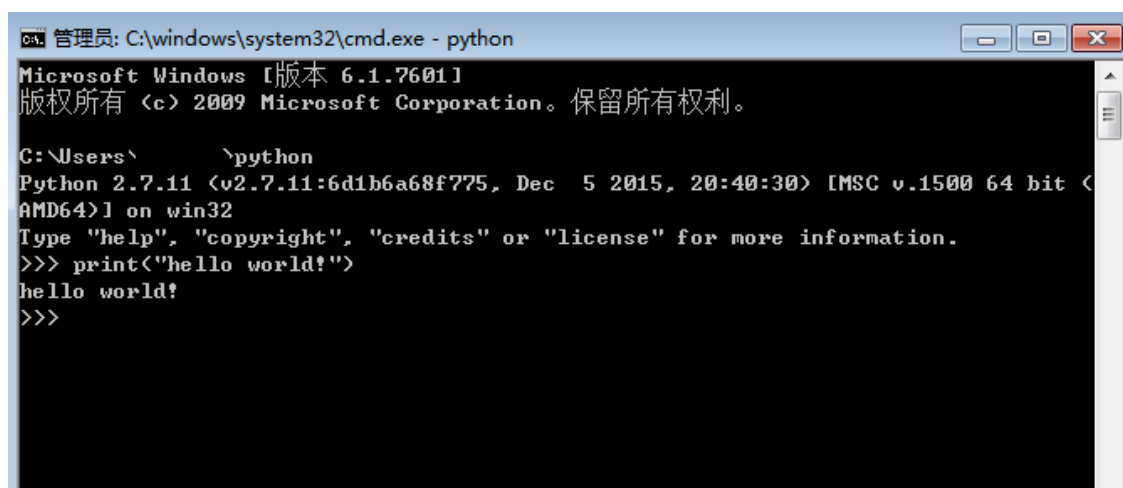
小编的环境是 Windows 7 版本 64 位系统 (32 位的同学自己想办法哦)

1. jdk1.6.0 (64 位)
2. android-sdk_r24.3.4-windows
3. python:2.7 (3.6 也可以)

4. appium: 1.4.13.1
5. Node.js: node-v4.4.7-x64
6. Appium-Python-Client

1.2.1 Python 安装

1. 小编的电脑操作系统: win7 64 位系统
2. 下载 Python 安装包, 选择 2.7 版本和 3.6 版本都可以
官网下载地址: <https://www.python.org/15>
3. Python 安装, 双击傻瓜式安装 (别安装在 c 盘哦)
4. 小编的安装目录在 d 盘: D:\python
5. 安装完成后, 看下这个目录 D:\python\Scripts, 有没 pip.exe 和 easy_install.exe (一般都有)
6. 将 D:\python 和 D:\python\Scripts, 添加到环境变量 path 下
7. 打开 cmd 输入 python, 出现版本号, 然后输入 print("hello world!")



```
管理员: C:\windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。


C:\Users\...\python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:40:30) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world!")
hello world!
>>>
```


1.2.3 安装 node.js


1. 下载官网地址: <https://node.js.org/en/download/23> (不会下载的在群文件找吧 appium 交流 QQ 群: 512200893)

LTS
Recommended For Most Users

Current
Latest Features


Windows Installer
node-v4.5.0-x64.msi


Macintosh Installer
node-v4.5.0.pkg


Source Code
node-v4.5.0.tar.gz

Windows Installer (.msi)

Windows Binary (.exe)

Mac OS X Installer (.pkg)

Mac OS X Binaries (.tar.gz)

Linux Binaries (.tar.xz)

Source Code

2. 下载后一路傻瓜式安装，安装完成后，运行 cmd，输入 `node -v` 查看版本号，然后输入 `npm`

```
管理员: C:\windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\>node -v
v4.4.7

C:\Users\>npm

Usage: npm <command>

where <command> is one of:
  access, add-user, adduser, apihelp, author, bin, bugs, c,
  cache, completion, config, ddp, dedupe, deprecate, dist-tag,
  dist-tags, docs, edit, explore, faq, find, find-dupes, get,
  help, help-search, home, i, info, init, install, issues, la,
  link, list, ll, ln, login, logout, ls, outdated, owner,
  pack, ping, prefix, prune, publish, r, rb, rebuild, remove,
  repo, restart, rm, root, run-script, s, se, search, set,
  show, shrinkwrap, star, stars, start, stop, t, tag, team,
  test, tst, un, uninstall, unlink, unpublish, unstar, up,
  update, upgrade, v, version, view, whoami

npm <cmd> -h      quick help on <cmd>
npm -l            display full usage info

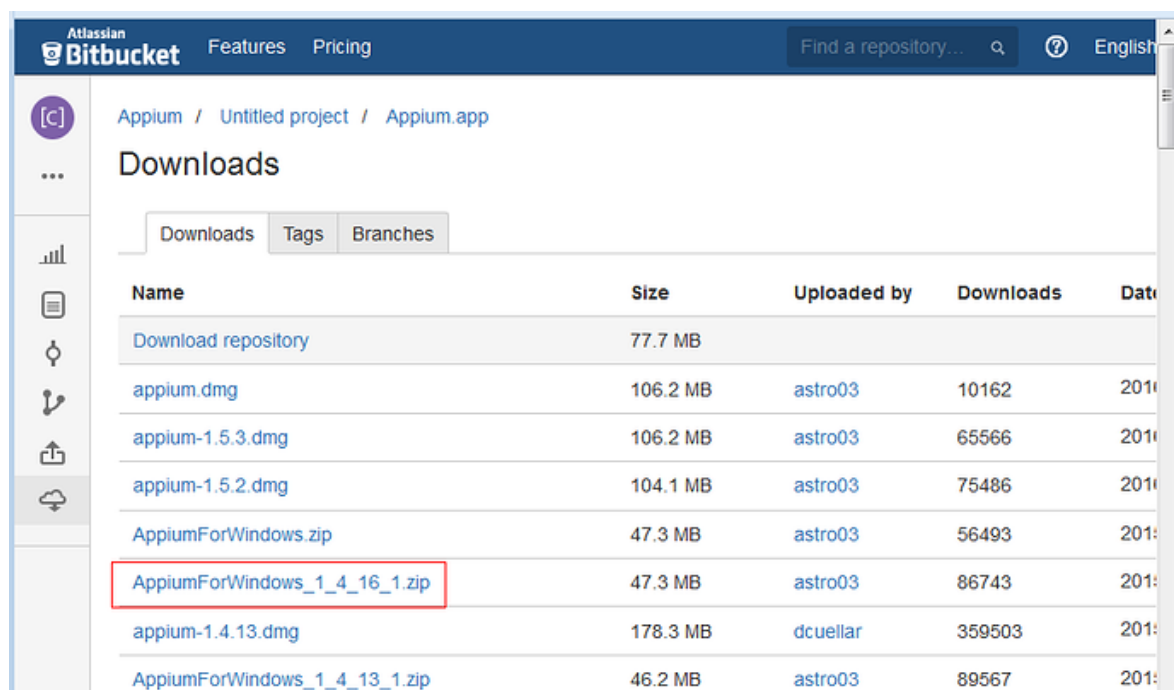
半:
```

3. 出现如上图信息，表示 node.js 安装成功。npm 是一个 node 包管理和分发工具，有了 npm，后面就可以输入指令在线安装 appium(打开 cmd 输入：`npm install -g appium`但是一般不推荐这种，下载比较慢，所以用下面这种客户端安装)

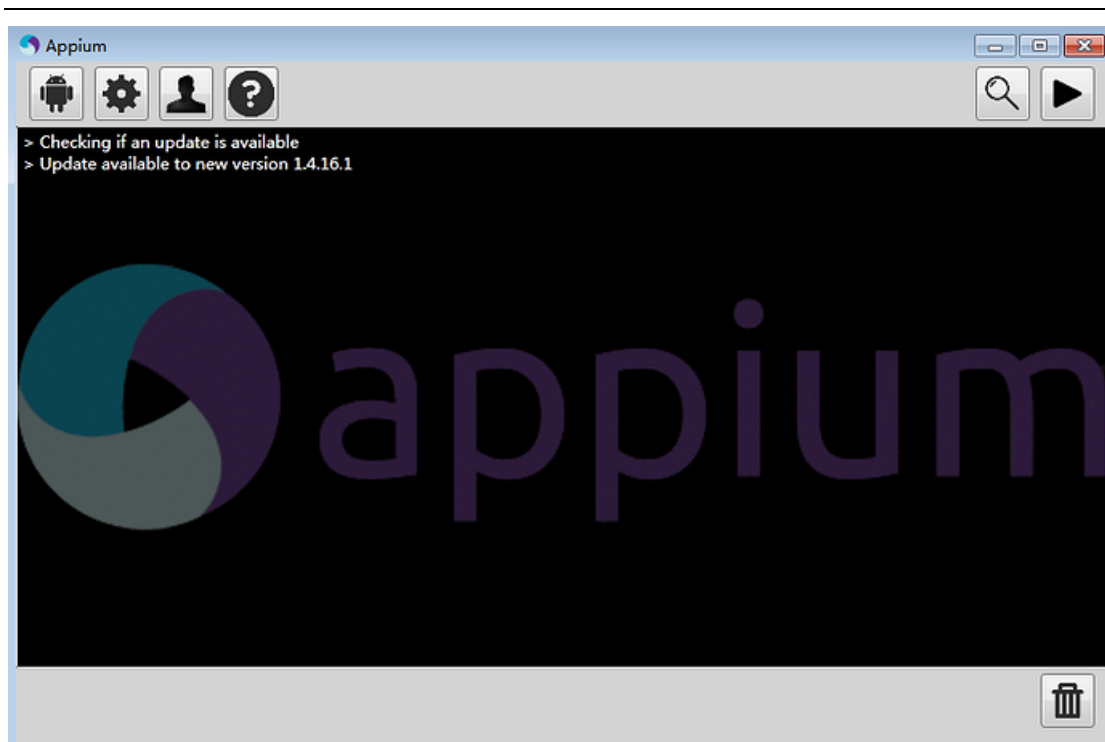
1.2.4 安装 appium

1. 下载安装地址：

<https://bitbucket.org/appium/appium.app/downloads/16>

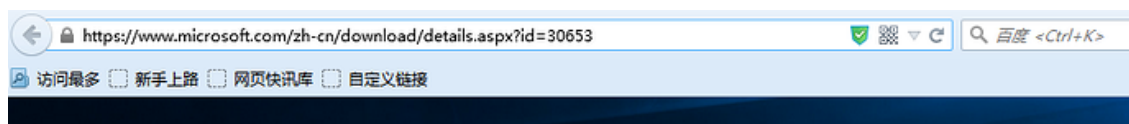


2. 直接双击 appium-installer.exe 文件安装就好，桌面会生成一个 appium 的图标，启动后界面显示如下

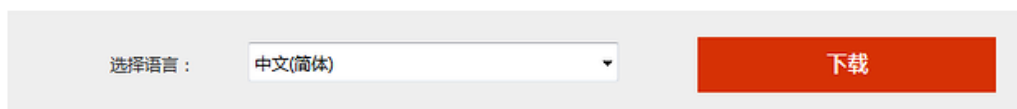


1.2.5 安装.net framework

1. Appium是用.net 开发的,所以需要安装.net framework4.5,下载地址:
<https://www.microsoft.com/zh-cn/download/details.aspx?id=30653>



Microsoft .NET Framework 4.5

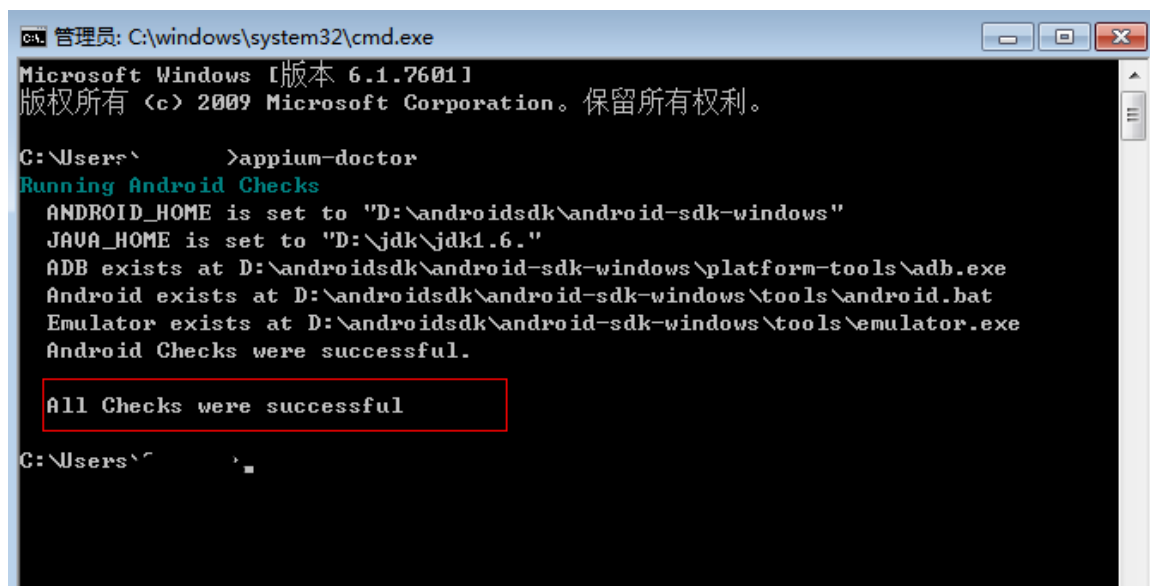


.NET Framework 4.5 是一个针对 .NET Framework 4 的高度兼容的就地更新。

(+) 详情

1.2.6 appium-doctor

1. appium 安装好后，找到这个文件目录
D:\appium\AppData\Local\appium\node_modules\.bin
2. 将上面的地址添加到环境变量 path 下
3. 打开 cmd，输入 appium-doctor，检查环境是否 OK，出现如下图所示，说明环境 OK



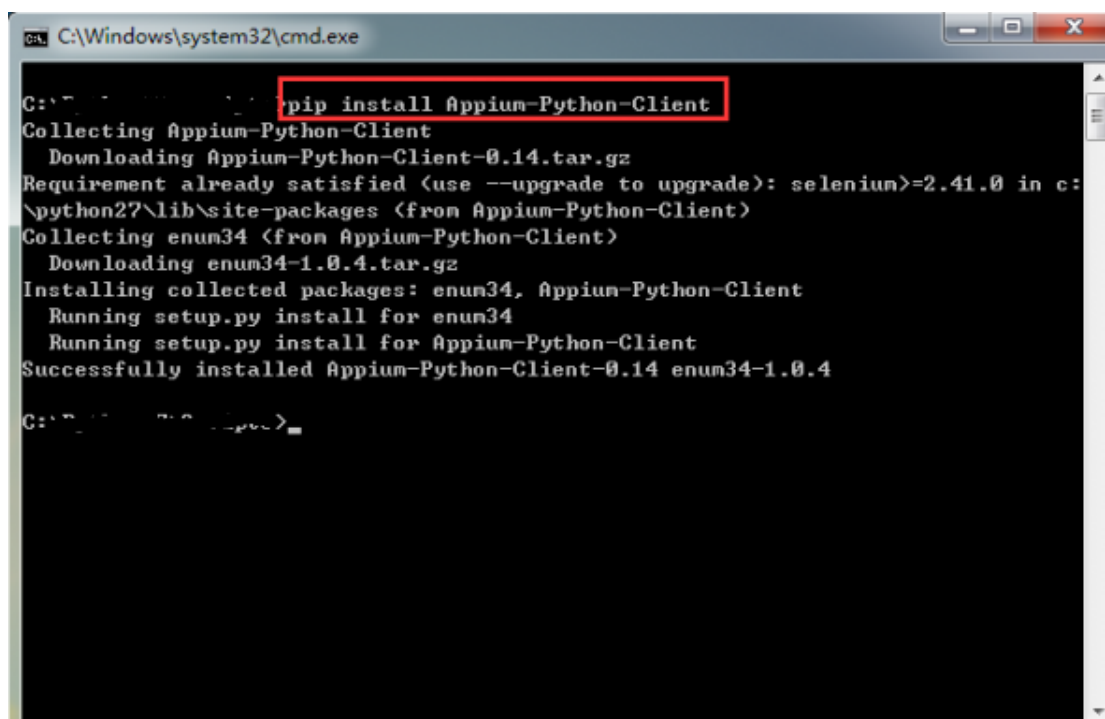
```
C:\Users\...\>appium-doctor
Running Android Checks
  ANDROID_HOME is set to "D:\androidsdk\android-sdk-windows"
  JAVA_HOME is set to "D:\jdk\jdk1.6."
  ADB exists at D:\androidsdk\android-sdk-windows\platform-tools\adb.exe
  Android exists at D:\androidsdk\android-sdk-windows\tools\android.bat
  Emulator exists at D:\androidsdk\android-sdk-windows\tools\emulator.exe
  Android Checks were successful.

All Checks were successful

C:\Users\...\>
```

七、安装 Appium-Python-Client

1. 前面 python 环境安装，已经准备好 pip 了，所以这里直接打开 cmd，输入：pip install Appium-Python-Client



```
C:\Windows\system32\cmd.exe

C:\> pip install Appium-Python-Client
Collecting Appium-Python-Client
  Downloading Appium-Python-Client-0.14.tar.gz
Requirement already satisfied (use --upgrade to upgrade): selenium>=2.41.0 in c:\python27\lib\site-packages (from Appium-Python-Client)
Collecting enum34 (from Appium-Python-Client)
  Downloading enum34-1.0.4.tar.gz
Installing collected packages: enum34, Appium-Python-Client
  Running setup.py install for enum34
  Running setup.py install for Appium-Python-Client
Successfully installed Appium-Python-Client-0.14 enum34-1.0.4

C:\>
```

到本篇结束，该安装的软件都已经安装好，接下来就是怎么去用了。小伙伴们环境装好后，已经迫不及待的想进入实战环节了，也别太着急，工欲善其事必先利其器，环境都装好了，接下来的想怎么玩，那不 是 so easy 么。

第 2 章 API 详解

2.1 启动 app

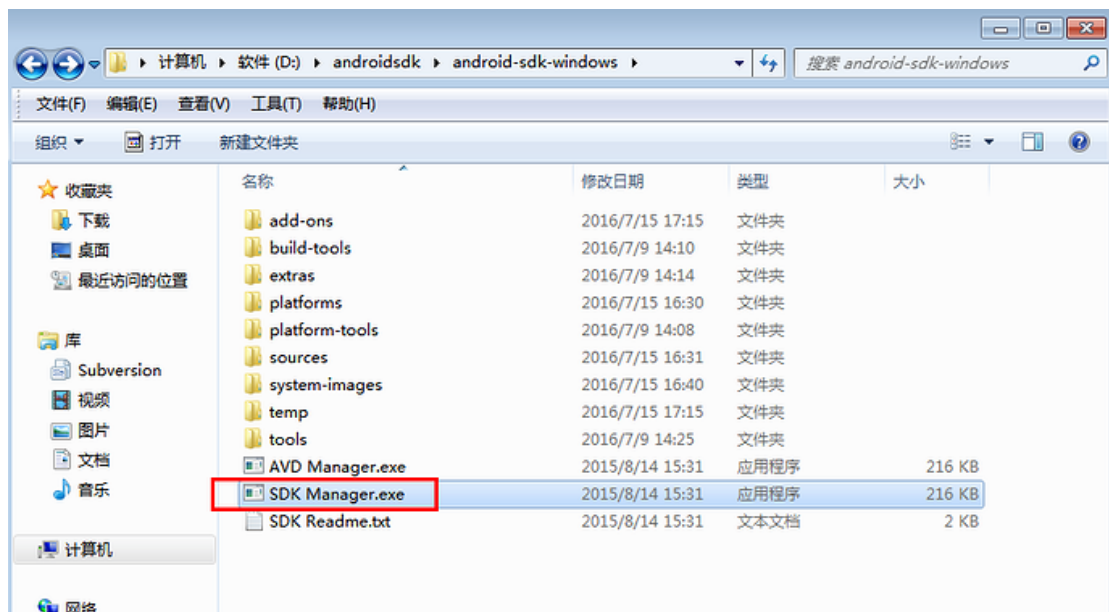
前言

前面两篇环境已经搭建好了，接下来就是需要启动 APP，如何启动 app 呢？首先要获取包名，然后获取 launcherActivity。获取这两个关键东西的方法很多，这里就不一一多说，小伙伴们可以各显神通。小编这里主要给大家推荐一个 sdk 自带的实用工具 aapt。

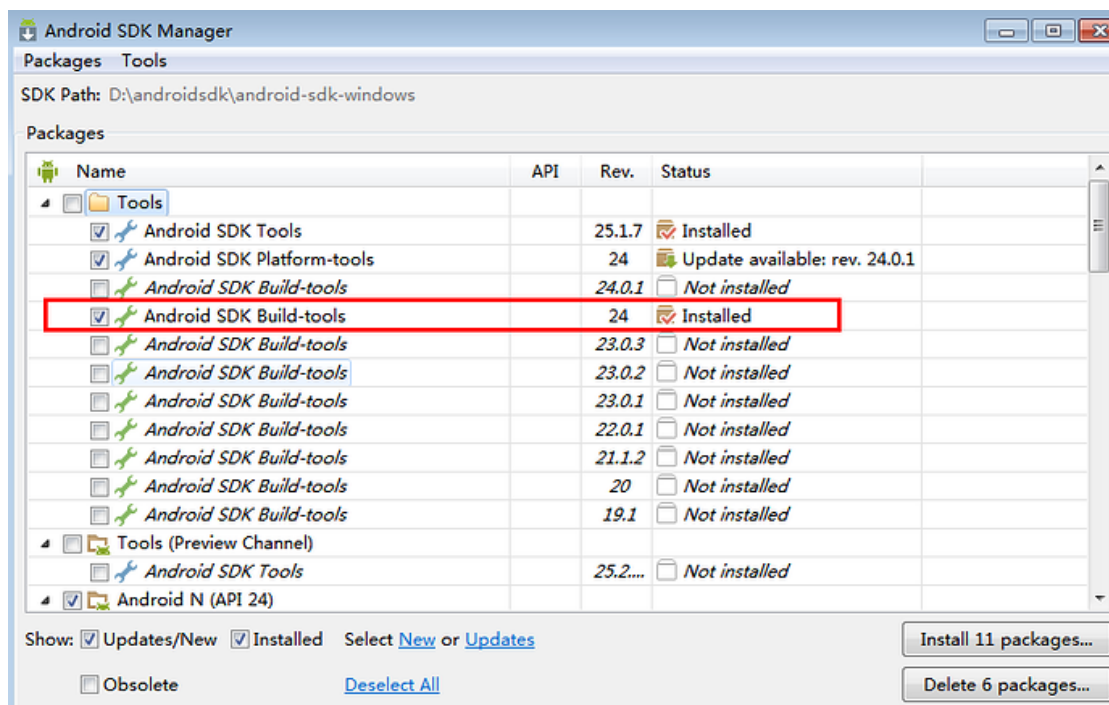
aapt 即 Android Asset Packaging Tool，在 SDK 的 build-tools 目录下。该工具可以查看 apk 包名和 launcherActivity，当然还有更多的功能，有兴趣的可以查看相关资料。

2.1.1 下载 aapt

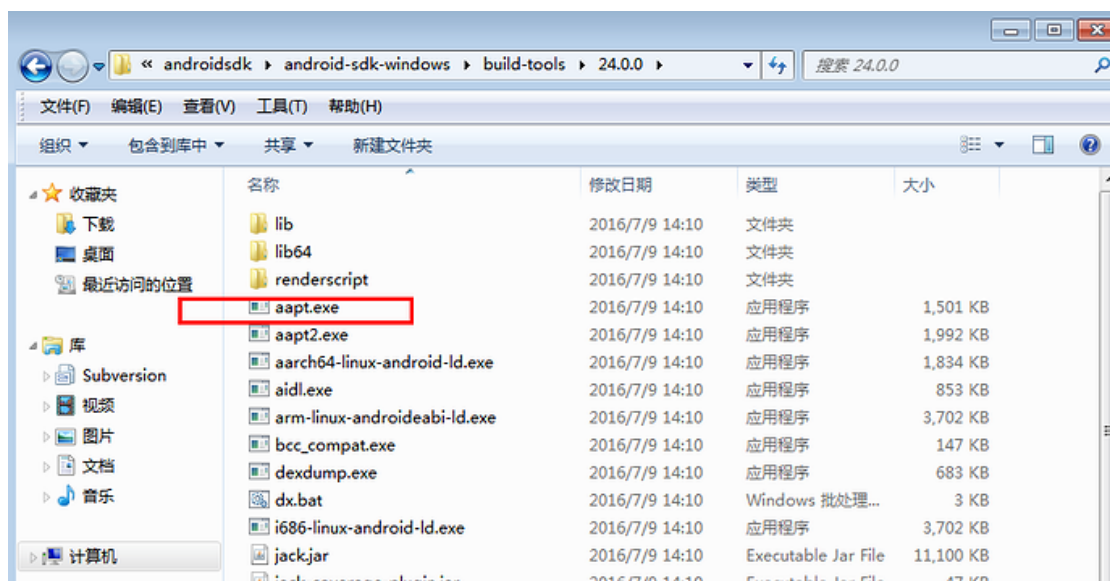
1. 在 android-sdk 里面双击 SDK-manager, 下载 buidl-tools



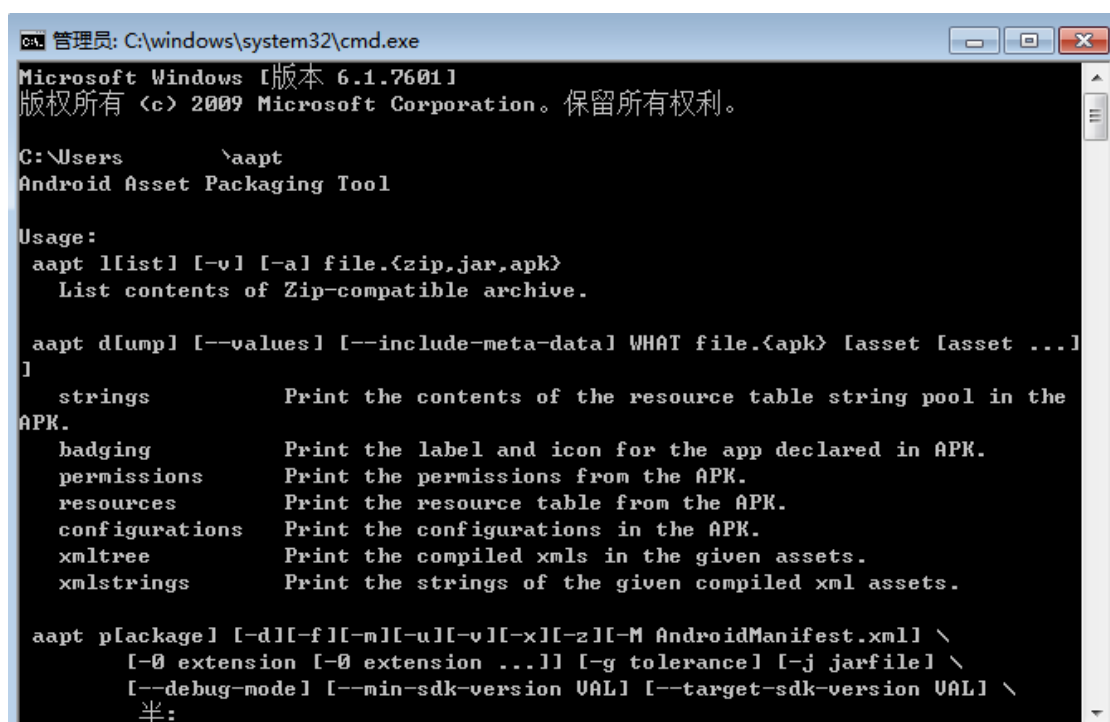
2. 勾选 build-tools, 随便选一个版本, 我这里选的是 24 的版本



3. 下载完成后，在
D:\androidsdk\android-sdk-windows\build-tools\24.0.0 目录下找到
aapt.exe，将这个路径设置环境变量，添加到 path 下

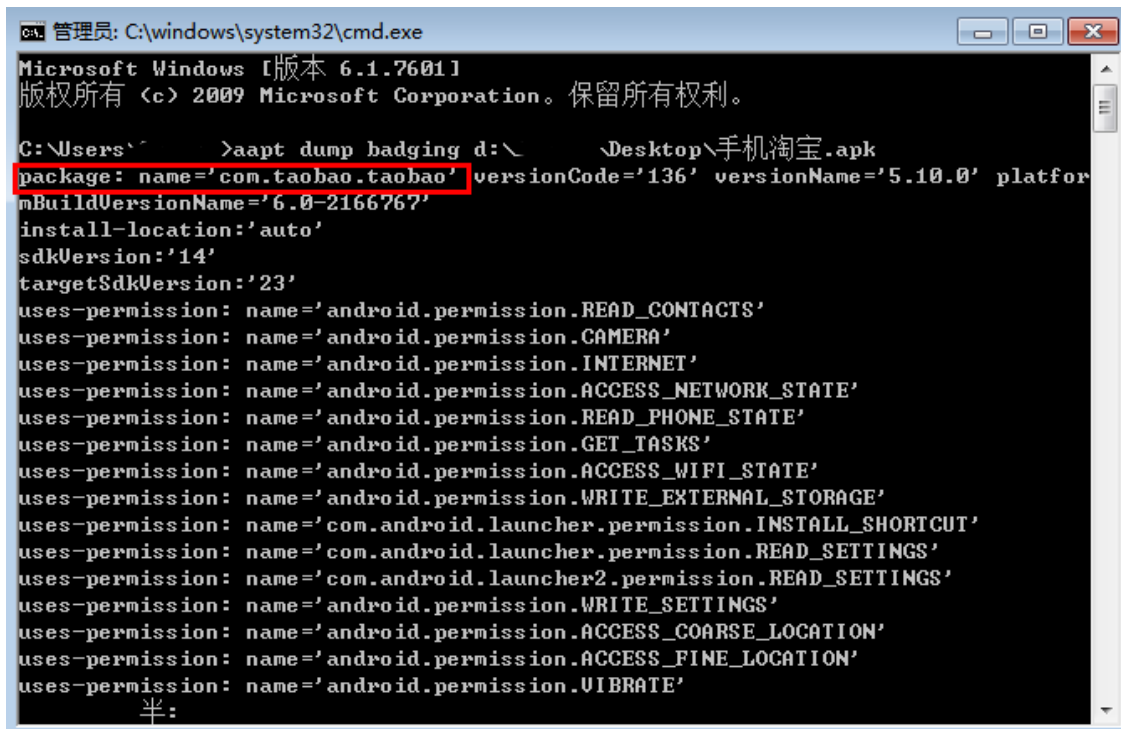


4. 打开 cmd。输入 aapt 出现如下界面，说明环境 OK 了



2.1.2 获取 apk 包名

1. 将准备测试的 APK 放到 D 盘某个目录，如 D:\test
2. 打开 cmd, 输入指令 `aapt dump badging D:\test\xxx.apk` (APK 的全名, 如手机淘宝.apk)
3. 以手机淘宝.apk 为例，如下图



```
管理员: C:\windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

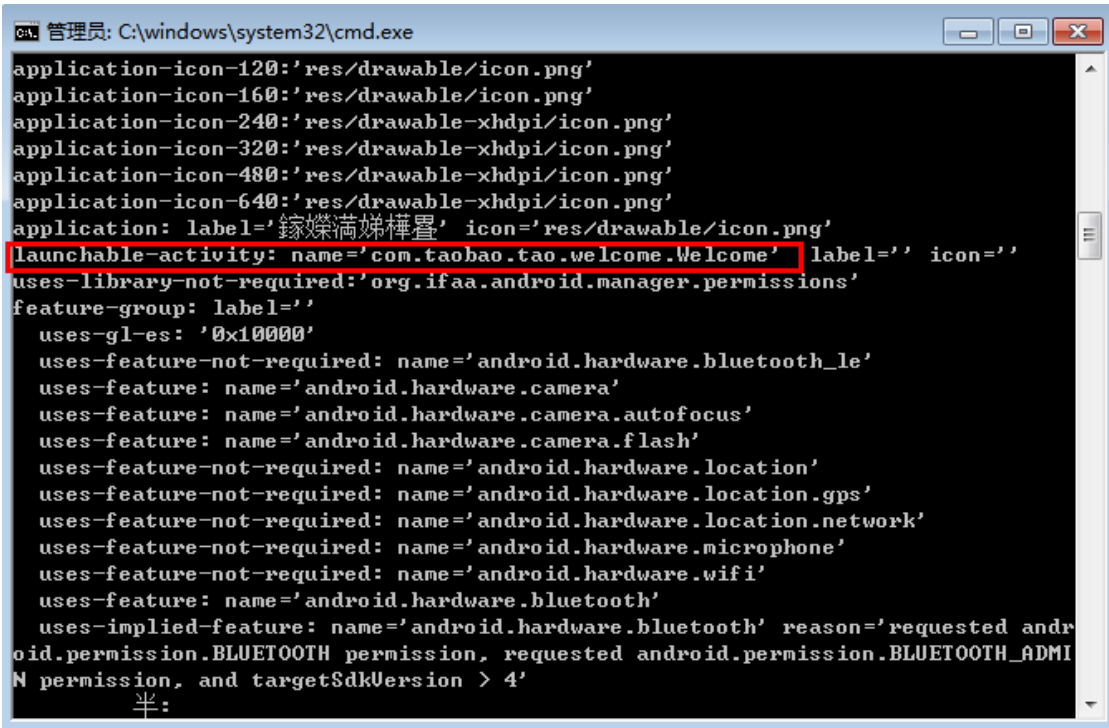
C:\Users\>aapt dump badging d:\Desktop\手机淘宝.apk
package: name='com.taobao.taobao' versionCode='136' versionName='5.10.0' platformBuildVersionName='6.0-2166767'
install-location:'auto'
sdkVersion:'14'
targetSdkVersion:'23'
uses-permission: name='android.permission.READ_CONTACTS'
uses-permission: name='android.permission.CAMERA'
uses-permission: name='android.permission.INTERNET'
uses-permission: name='android.permission.ACCESS_NETWORK_STATE'
uses-permission: name='android.permission.READ_PHONE_STATE'
uses-permission: name='android.permission.GET_TASKS'
uses-permission: name='android.permission.ACCESS_WIFI_STATE'
uses-permission: name='android.permission.WRITE_EXTERNAL_STORAGE'
uses-permission: name='com.android.launcher.permission.INSTALL_SHORTCUT'
uses-permission: name='com.android.launcher.permission.READ_SETTINGS'
uses-permission: name='com.android.launcher2.permission.READ_SETTINGS'
uses-permission: name='android.permission.WRITE_SETTINGS'
uses-permission: name='android.permission.ACCESS_COARSE_LOCATION'
uses-permission: name='android.permission.ACCESS_FINE_LOCATION'
uses-permission: name='android.permission.VIBRATE'
```

4. 这里就可以看到 apk 的包名: `com.taobao.taobao`

注：老司机可以直接把 apk 放在桌面上，输入指令后拖到 cmd 框

2.1.3 获取 launcherActivity

1. 接着上一步操作，cmd 屏幕拖到中间 1 找到 `launcherActivity`
2. 这里可以看到，淘宝的 `launcherActivity` 值为 `com.taobao.tao.welcome.Welcome`



```
C:\windows\system32\cmd.exe
application-icon-120:'res/drawable/icon.png'
application-icon-160:'res/drawable/icon.png'
application-icon-240:'res/drawable-xhdpi/icon.png'
application-icon-320:'res/drawable-xhdpi/icon.png'
application-icon-480:'res/drawable-xhdpi/icon.png'
application-icon-640:'res/drawable-xhdpi/icon.png'
application: label='錄蝶滿梯樺置' icon='res/drawable/icon.png'
launchable-activity: name='com.taobao.tao.welcome.Welcome' label='' icon=''
uses-library-not-required:'org.ifaa.android.manager.permissions'
feature-group: label=''
  uses-gl-es: '0x10000'
  uses-feature-not-required: name='android.hardware.bluetooth_le'
  uses-feature: name='android.hardware.camera'
  uses-feature: name='android.hardware.camera.autofocus'
  uses-feature: name='android.hardware.camera.flash'
  uses-feature-not-required: name='android.hardware.location'
  uses-feature-not-required: name='android.hardware.location.gps'
  uses-feature-not-required: name='android.hardware.location.network'
  uses-feature-not-required: name='android.hardware.microphone'
  uses-feature-not-required: name='android.hardware.wifi'
  uses-feature: name='android.hardware.bluetooth'
  uses-implied-feature: name='android.hardware.bluetooth' reason='requested android.permission.BLUETOOTH permission, requested android.permission.BLUETOOTH_ADMIN permission, and targetSdkVersion > 4'
```

2.1.4 写脚本

- 1.platformName: 这里是 android 的 apk
- 2.deviceName: 手机设备名称, 通过 adb devices 查看
- 3.platformVersion: android 系统的版本号
- 4.appPackage: apk 包名
- 5.appActivity: apk 的 launcherActivity

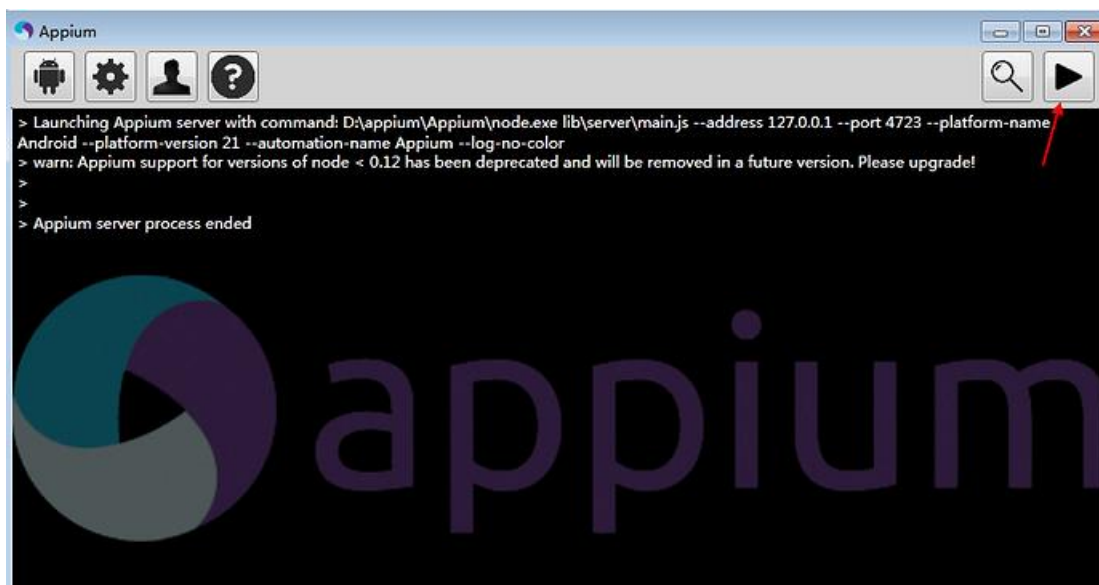
```
# coding=utf-8
from appium import webdriver

desired_caps = {
    'platformName': 'Android',
    'deviceName': '30d4e606',
    'platformVersion': '5.0',
    # apk包名
    'appPackage': 'com.taobao.taobao',
    # apk的launcherActivity
    'appActivity': 'com.taobao.tao.welcome.Welcome'
}

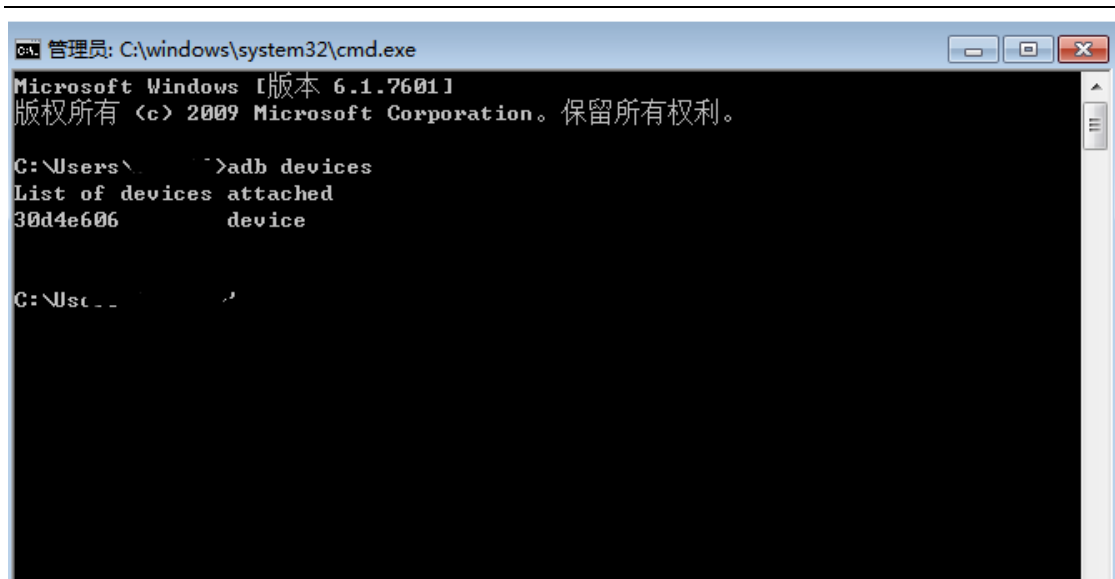
driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired_caps)
```

2.1.5 运行 appium

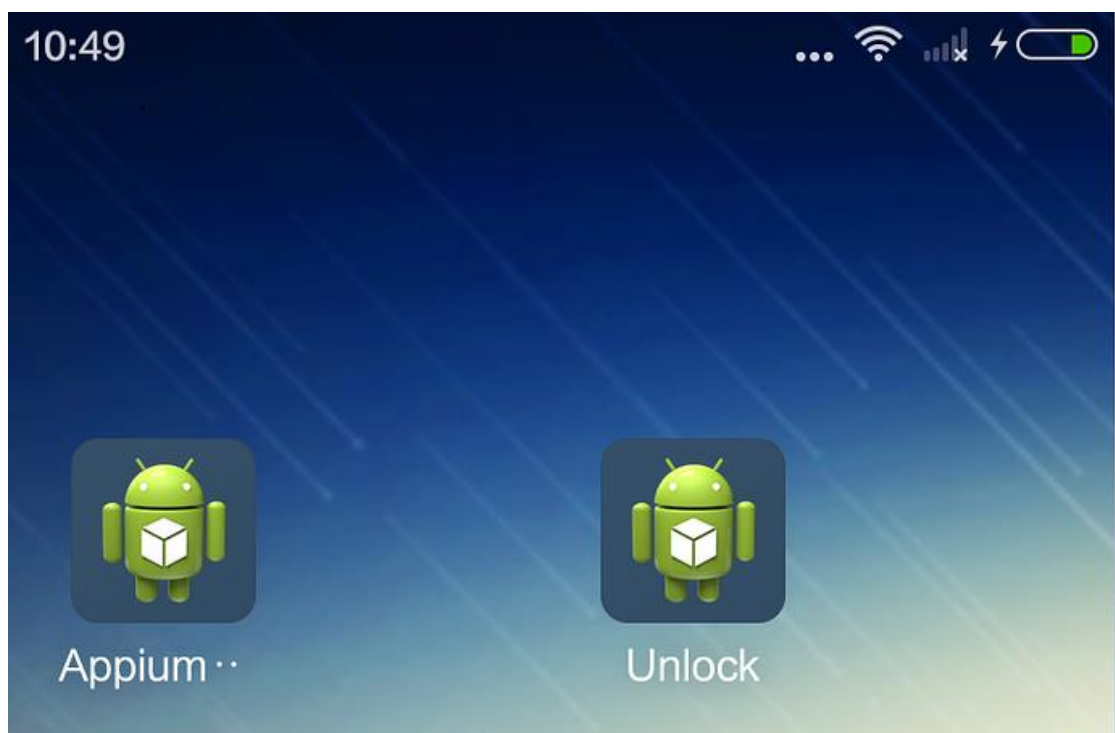
1. 启动 appium, 右上角点三角形按钮，变成正方形，就是启动状态。



2. 确认手机连上电脑



3. 在 pycharm 运行脚本，随后在手机上会弹出安装下面两个软件的提示，安装后，桌面上多两个图标。那么恭喜你启动成功！



4. 接着会看到淘宝 app 已经启动啦，有木有小激动~~

2.1.6 最终代码如下

```
# coding=utf-8

from appium import webdriver

desired_caps = {

    'platformName': 'Android',

    'deviceName': '30d4e606',

    'platformVersion': '5.0',

    # apk 包名

    'appPackage': 'com.taobao.taobao',

    # apk 的 launcherActivity

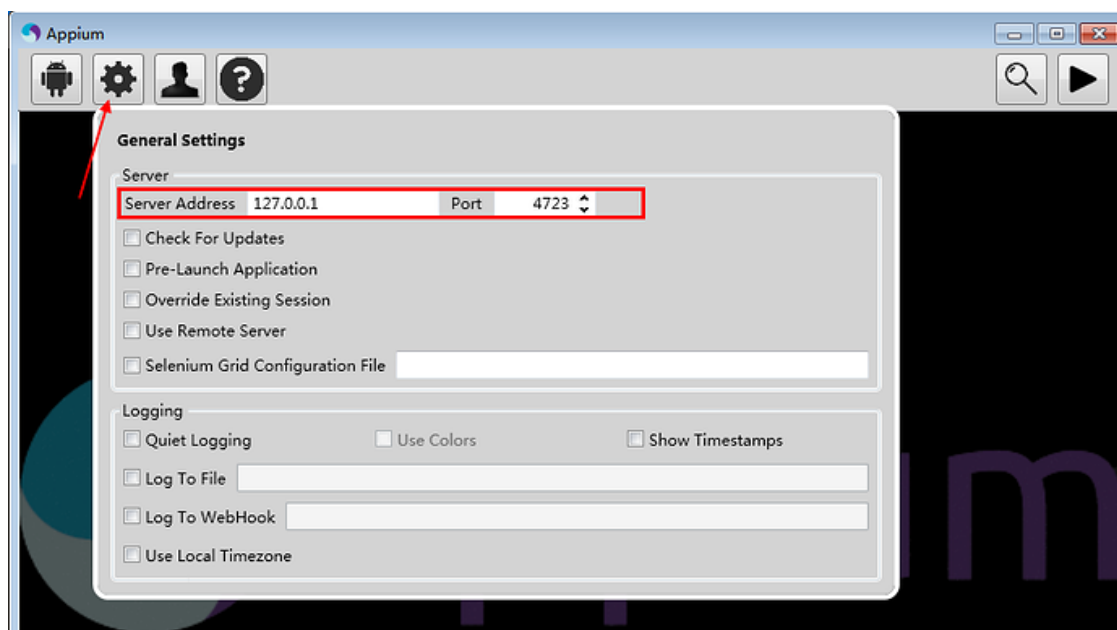
    'appActivity':

'com.taobao.tao.welcome.Welcome'

}

driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub',
desired_caps)
```

这个地址是怎么来的呢？



这一篇主要学会使用 aapt 工具，然后启动 app 的一个流程，启动 app 后，下一步就是要定位元素了，定位元素 android sdk 里面用一个自带的 uiautomatorviewer，appium 里面也有一个 Inspector，下篇会详细介绍。

2.2 元素定位 uiautomatorviewer

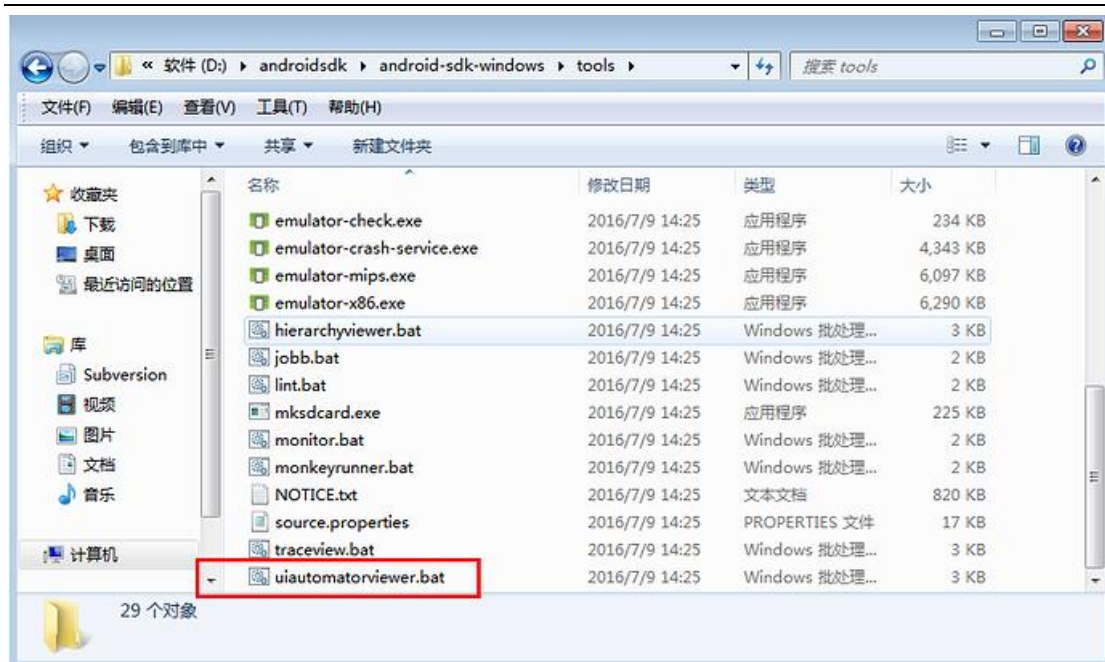
前言

环境搭建好了，下一步元素定位，元素定位本篇主要介绍如何使用 uiautomatorviewer，通过定位到页面上的元素，然后进行相应的点击等操作。

uiautomatorviewer 是 android-sdk 自带的一个元素定位工具，非常简单好用，使用 uiautomatorviewer，你可以检查一个应用的 UI 来查看应用的布局和组件以及相关的属性。

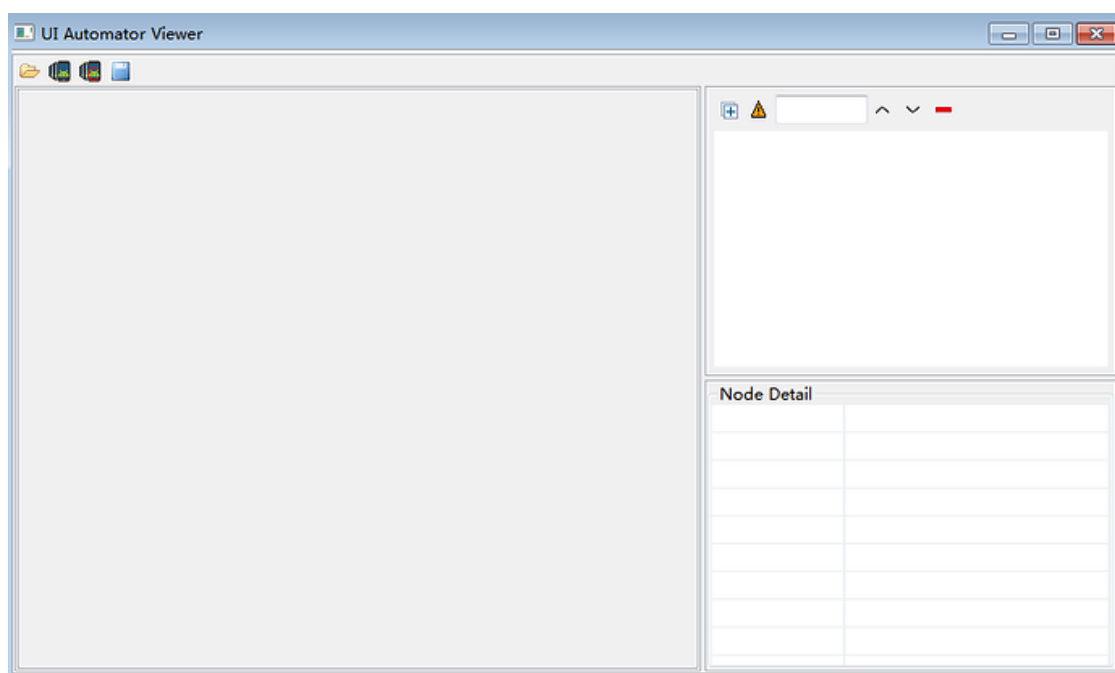
2.2.1 启动 uiautomatorviewer.bat

1. 打开目录 D:\androidsdk\android-sdk-windows\tools



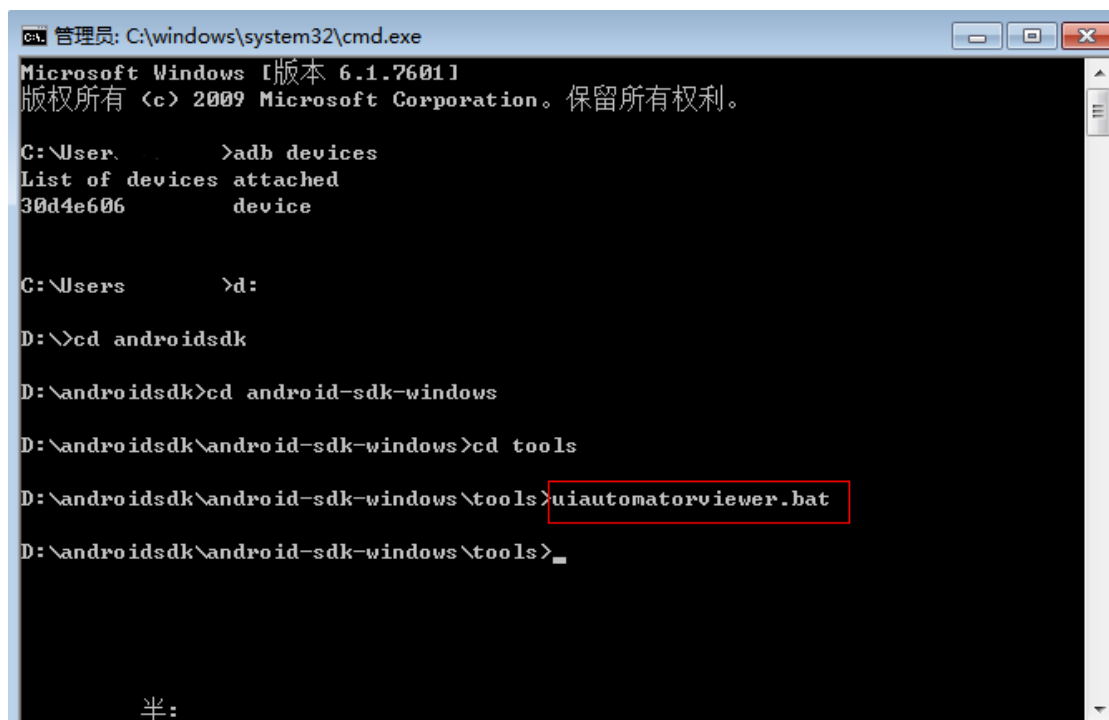
1472105236126324.png850x502 53 KB

2. 双击启动，启动之后出现如下界面



3 如果不喜欢双击启动的话，也可以在 cmd 里面通过指令启动

先 cd 到 tools 目录下，然后输入 uiautomatorviewer.bat 回车后启动服务



```
管理员: C:\windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\User.      >adb devices
List of devices attached
30d4e606      device

C:\Users      >d:

D:\>cd androidsdk

D:\androidsdk>cd android-sdk-windows

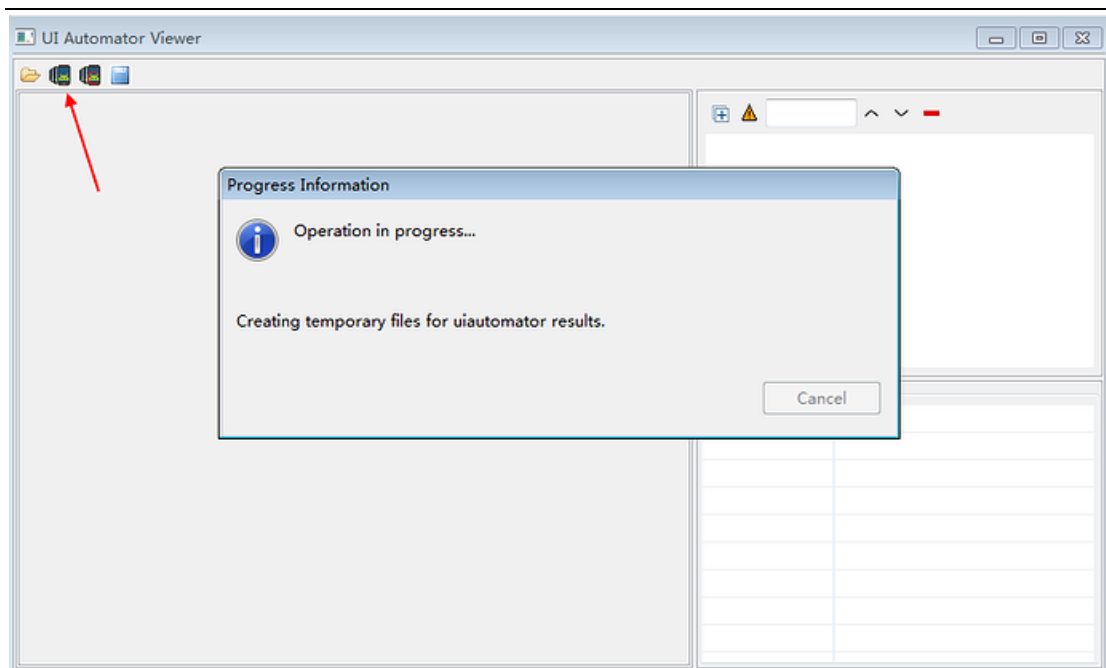
D:\androidsdk\android-sdk-windows>cd tools

D:\androidsdk\android-sdk-windows\tools>uiautomatorviewer.bat
D:\androidsdk\android-sdk-windows\tools>_

半:
```

2.2.2 连接手机

1. cmd 打开输入 adb devices, 确认手机已连上
2. 打开手机淘宝页面, 让屏幕处于点亮状态
3. 点左上角安卓机器人按钮 Devices Screenshot 按钮刷新页面



2.2.3 定位元素

1. 移动鼠标到需要定位的元素上，如搜索输入框



2. 右下角可以看到元素对应的属性

text: 搭配新宠不能缺

```
resource-id:com.taobao.taobao:id/home_searchedit
```

```
class:android.widget.EditText
```

2.2.4 点搜索框

1. 前面一骗启动 app 后，休眠五秒，等待页面加载完成
2. 通过 id 来定位到搜索框，然后点击

```
# coding=utf-8
from appium import webdriver
import time
desired_caps = {
    'platformName': 'Android',
    'deviceName': '30d4e606',
    'platformVersion': '5.0',
    'appPackage': 'com.taobao.taobao',
    'appActivity': 'com.taobao.tao.welcome.Welcome',
}

driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired_caps)
# 休眠五秒等待页面加载完成
time.sleep(5)
driver.find_element_by_id("com.taobao.taobao:id/home_searchedit").click()
```

2.2.5 参考代码

```
# coding=utf-8

from appium import webdriver

import time

desired_caps = {

    'platformName': 'Android',
```

```
        'deviceName': '30d4e606',

        'platformVersion': '5.0',

        'appPackage': 'com.taobao.taobao',

        'appActivity':
'com.taobao.tao.welcome.Welcome',

    }

driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub',
desired_caps)

# 休眠五秒等待页面加载完成

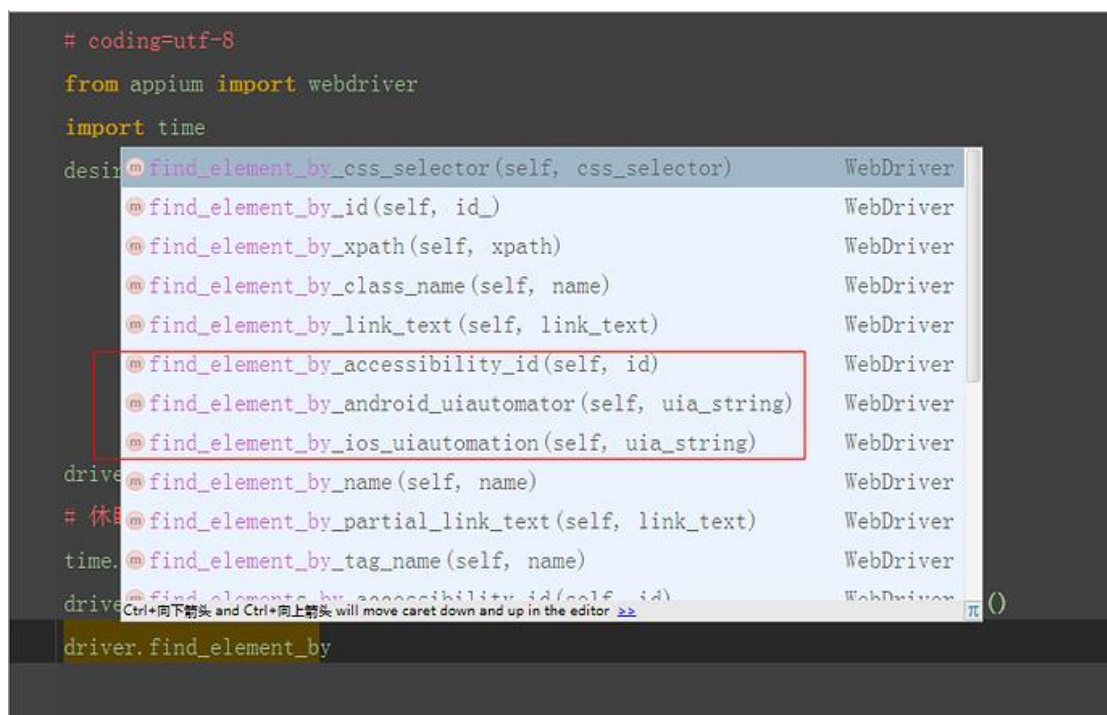
time.sleep(5)

driver.find_element_by_id("com.taobao.taobao:id/home_searchedit").click()
```

2.2.6 元素定位

（此处敲黑板，做笔记！！！！）

1. appium 的 webdriver 提供了 11 种元素定位方法，在 selenium 的基础上扩展了三个，可以在 pycharm 里面输入 driver.find_element_by 然后会自动匹配出来



2. 多的三种如下，在后面的会详细介绍

`driver.find_element_by_accessibility_id()`

`driver.find_element_by_android_uiautomator()`

`driver.find_element_by_ios_uiautomation()`

（第三个是 ios 的可以暂时不用管）

uiautomatorviewer 是 android sdk 自带的，下篇介绍 appium 自带个元素定位工具：Inspector. 在

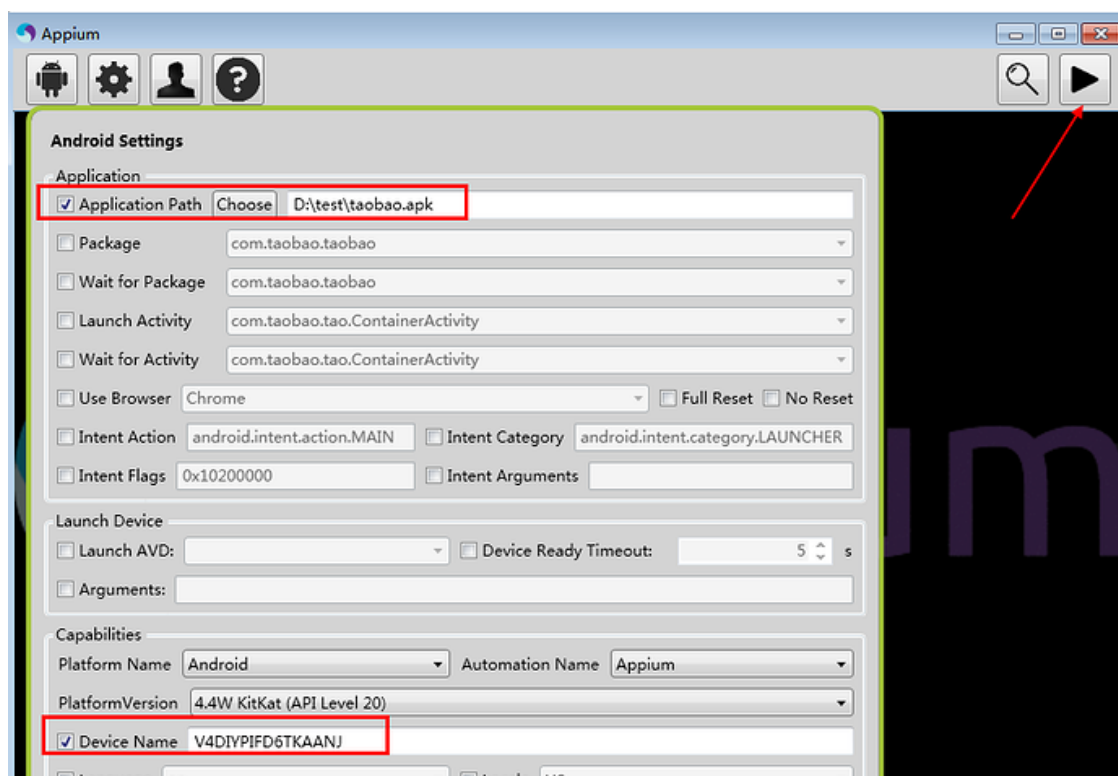
2.3 Appium Inspector

前言

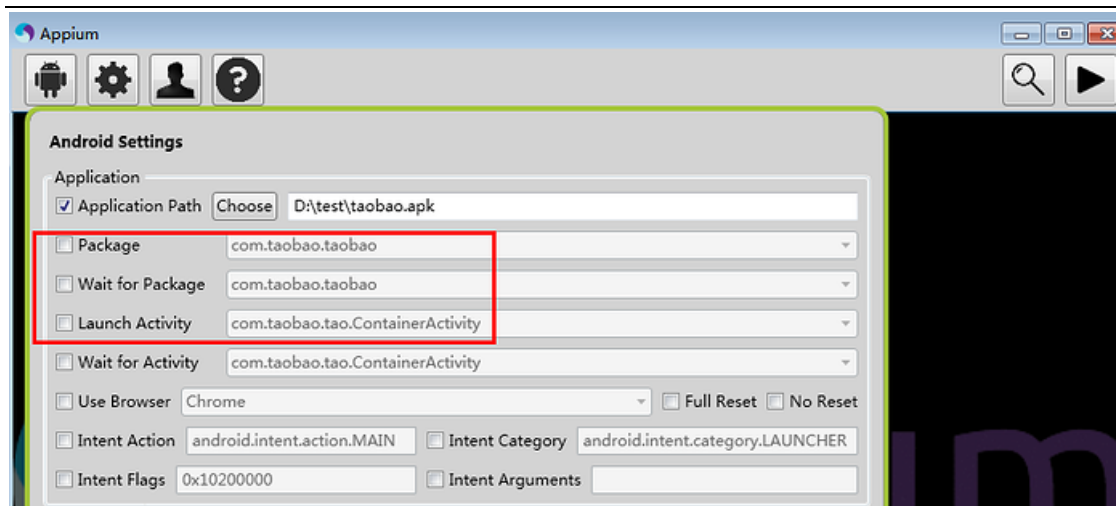
Appium Inspector 是 appium 自带的一个元素定位工具。appium Inspector 的 windows 版本不太好用，但是 Mac 上的功能还是很强大的，一般 mac 上用的比较多。

2.3.1 设置 appium

1. 先不要启动 appium
2. 点开 android setting 界面（机器人图标）
3. 勾选 Application Path, 添加被测 app 的路径
4. Devices name 处添加设备名称（adb devices 查看到的）



（敲黑板，记重点：通过这种方法也可查看到 apk 的包名和 Launch Activity）



2.3.2 开启 appium

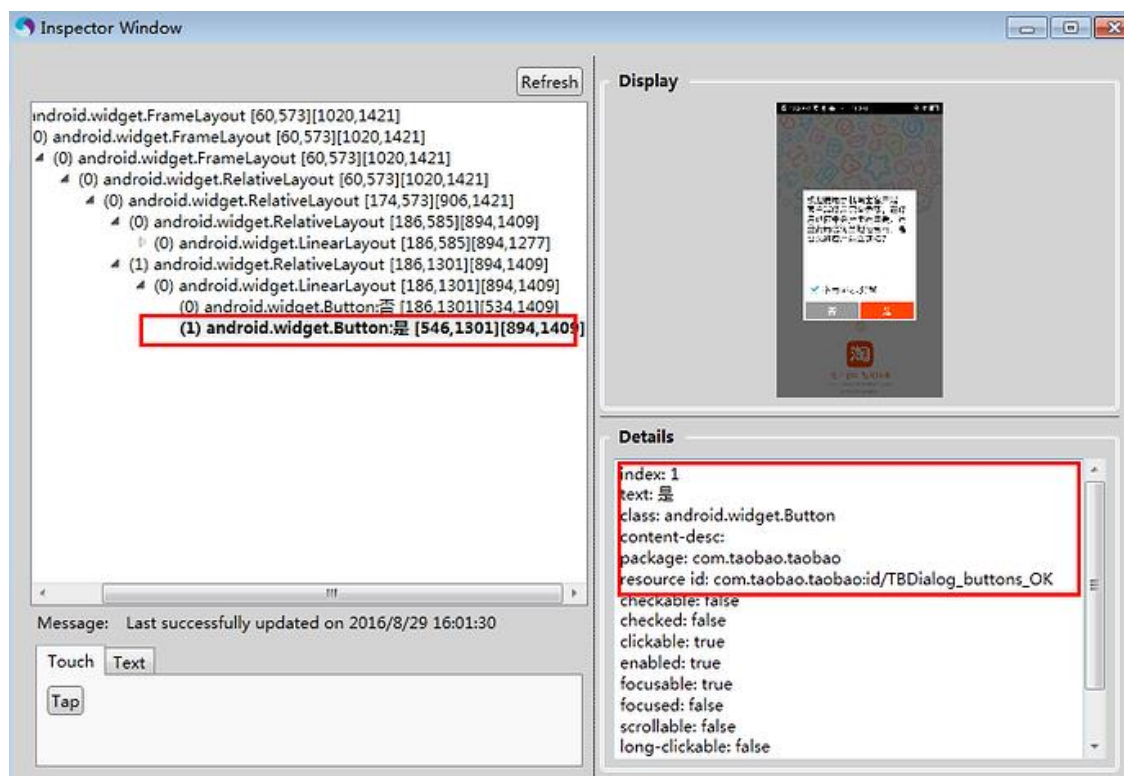
1. 手机确保连接电脑（adb devices 查看）
2. 点 appium 右上角三角形图标，会启动服务
3. 这时候可以看到手机上安装淘宝应用，并会启动淘宝
4. 点 appium 右上角的搜索图标
5. 点 Inspector Window 界面的 Refresh 按钮刷新界面

2.3.3 Inspector Window

1. 手机上打开需要单位元素的界面，然后点 Refresh 按钮刷新
2. 左边菜单树，可以挨个点开
3. 如果想单位界面上的“是”和“否”按钮，从菜单树就可以看到这两个元素的结构

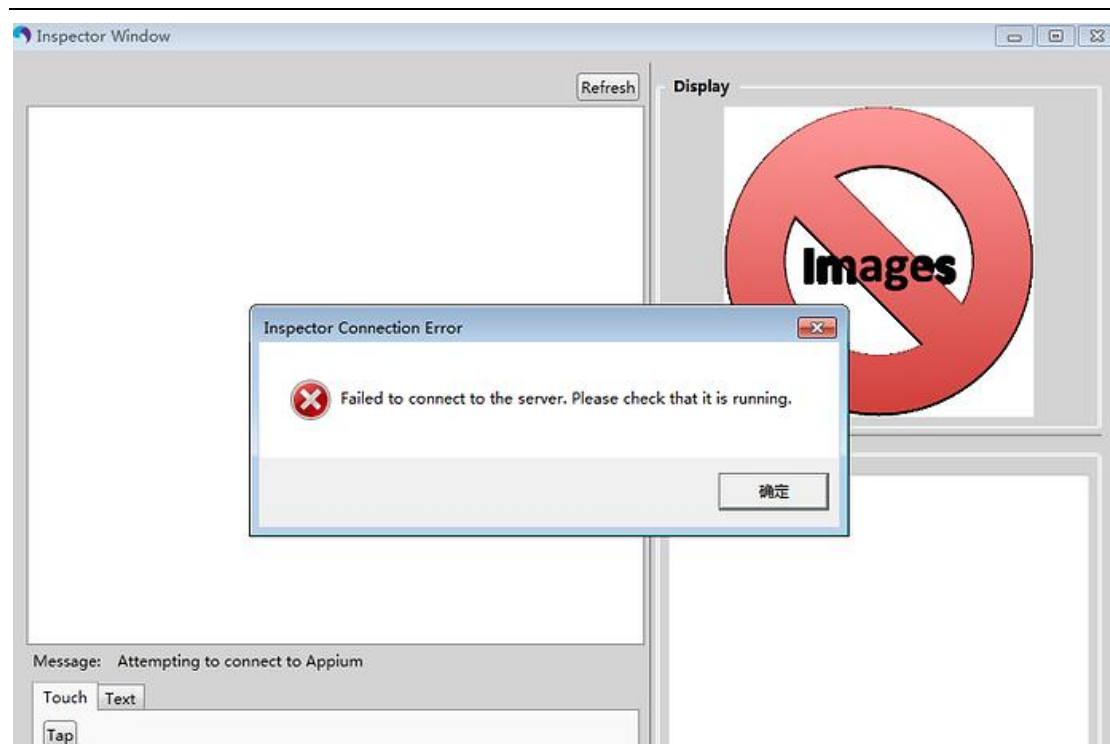
2.3.4 查看属性

1. 选中左侧菜单树对应的元素，在右下角查看对应属性



2.5.5 常见异常

1. 在使用过程中，你会发现经常会报以下这个错误
2. 每次启动都会给你手机上重新安装一次应用
3. 并且不能用鼠标指定某个元素，没有 uiautomatorviewer 使用方便



2.4 Remote 远程控制

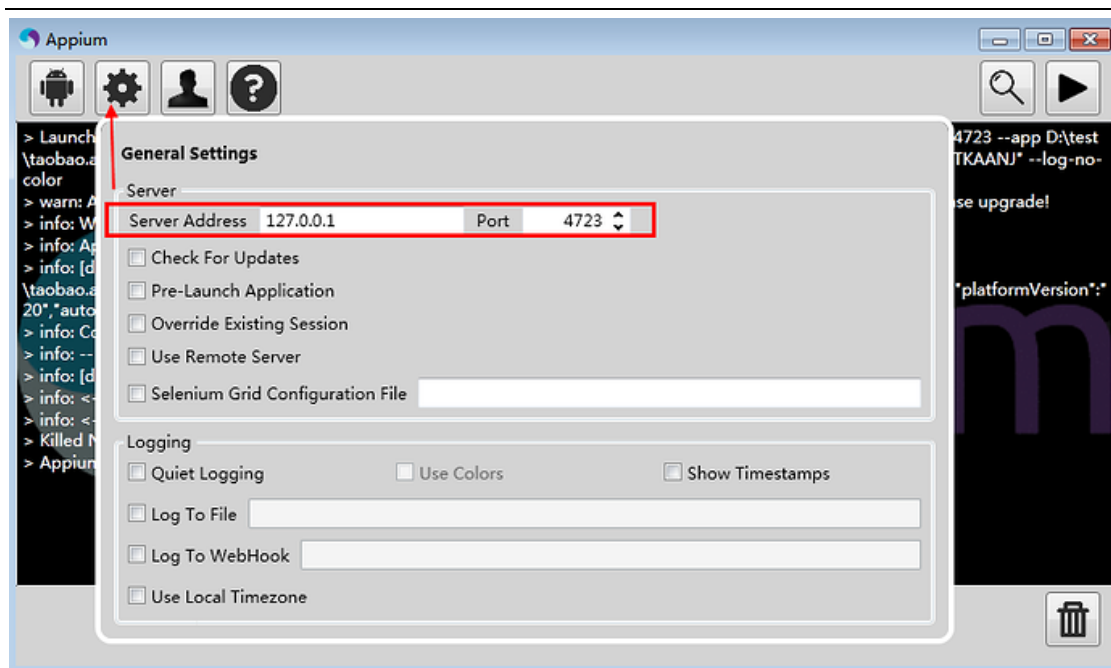
前言

在启动 app 的时候有这样一行代码 `driver = webdriver.Remote('http://192.168.1.1:4723/wd/hub', desired_caps)`，很多小伙伴不知道这个 ip 和端口哪里来的，于是小编决定写一篇关于这个 appium 的服务器 ip 文章！

一般来说 appium 中 127.0.0.1 这个地址的默认的不需要修改。在做自动化过程中，如果遇到需要远程操作的话，这个功能就可以派上用场了。想想看，如果公司给你单独配置一台跑自动化测试电脑，然后自己工作的电脑写脚本，在自动化机器上运行脚本，这样工作自动化两不误，是不是很爽呢？

2.4.1 设置 IP

1. 打开 appium>General Setting 界面

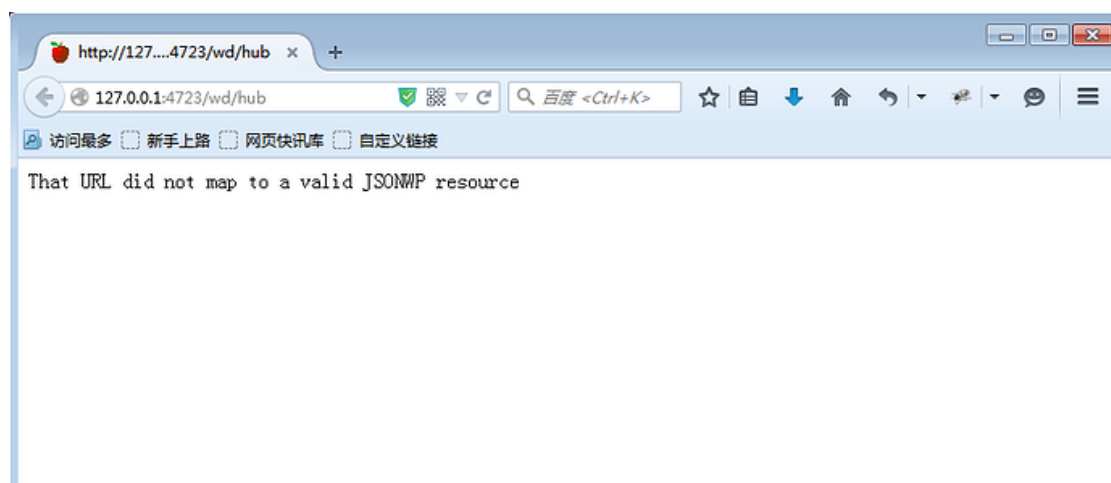


2. 这里用个 appium 默认的服务端地址 127.0.0.1 端口 4723，一般在自己机器上调试，无需修改

2.4.2 访问地址

1. 代码里面的那个地址到底指向的是哪里呢？

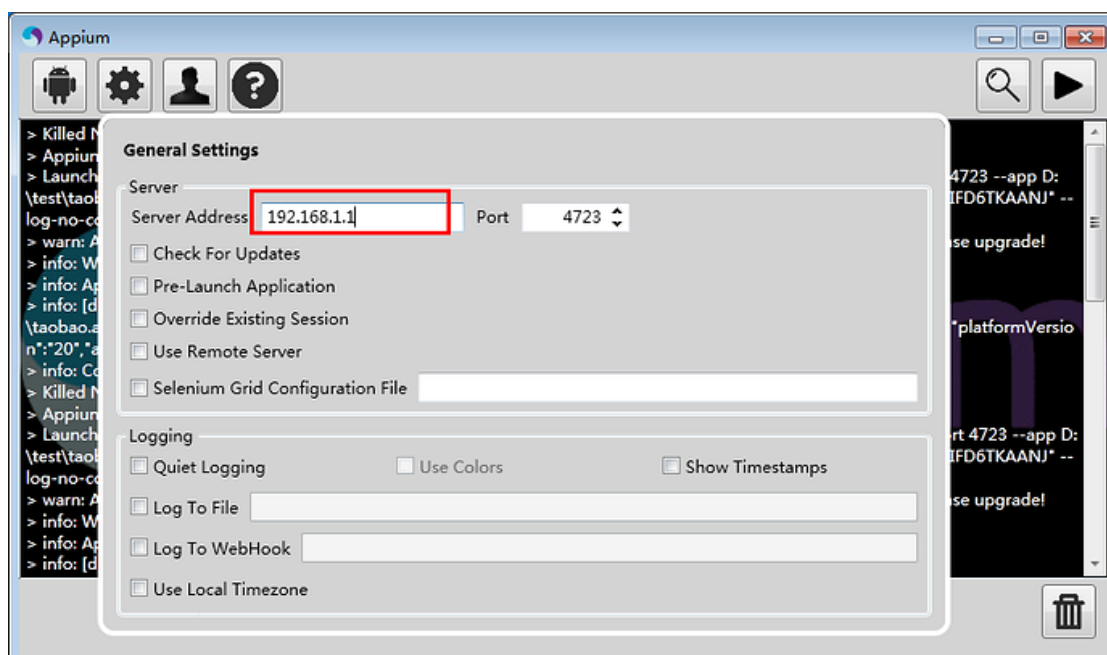
启动 appium 服务后，在浏览器输入：<http://127.0.0.1:4723/wd/hub>。出现如下图所示，说明服务启动成功，可以把 appium 看出是一个服务端。



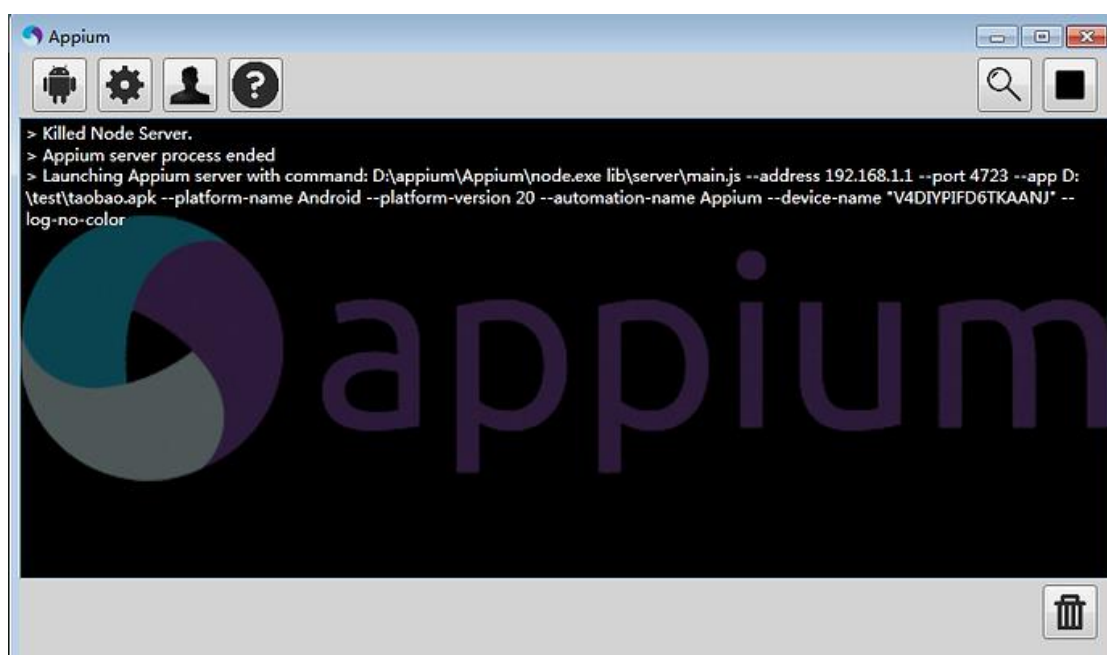
2.4.3 配置测试机

1. 一台工作电脑，一台跑自动化测试的电脑，如何用工作的电脑远程控制自动化测试的那台电脑呢？

2. 测试电脑上的 appium 服务地址改成本机 IP 地址如：192.168.1.1（敲黑板，记重点！这里要是本机的 ip 地址）

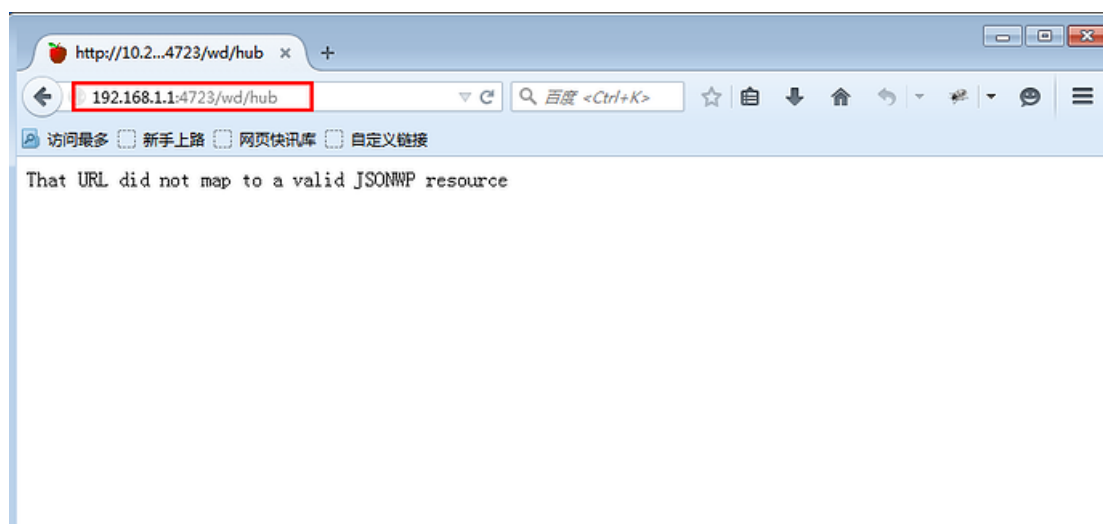


3. 启动测试电脑上的 appium。



2.4.4 远程操作

1. 在自己工作电脑上打开浏览器输入：192.168.1.1:4723/wd/hub。这一步很重要，一定要看到如下界面，确认远程链接成功。



2. 脚本里面的代码修改下地址

```
# coding=utf-8
from appium import webdriver
import time

desired_caps = {
    'platformName': 'Android',
    'deviceName': '30d4e606',
    'platformVersion': '5.0',
    'appPackage': 'com.taobao.taobao',
    'appActivity': 'com.taobao.tao.welcome.Welcome',
    'unicodeKeyboard': True,
    'resetKeyboard': True
}

driver = webdriver.Remote('http://192.168.1.1:4723/wd/hub', desired_caps)
```

3. 测试机上环境准备好后，在本机上运行脚本，于是测试机上可以自动运行了。

2.5 输入中文

前言

在做 app 自动化过程中会踩很多坑，咱们都是用的中文的 app，所以首先要解决中文输入的问题！

本篇通过屏蔽软键盘，绕过手机的软键盘方法，解决中文输入问题。

2.5.1 定位搜索

1. 打开淘宝点搜索按钮，进入到搜索页面

2. 然后定位到搜索框后用 sendkeys 方法输入“hao”，这里定位元素用第四篇讲的 uiautomatorviewer 工具就可以了

3. 脚本如下图（公众号版权所有：软件测试部落）

```
# coding=utf-8
from appium import webdriver
import time
desired_caps = {'platformName': 'Android',
                 'deviceName': '30d4e606',
                 'platformVersion': '5.0',
                 'appPackage': 'com.taobao.taobao',
                 'appActivity': 'com.taobao.tao.welcome.Welcome', }
driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired_caps)
# 休眠五秒等待页面加载完成
time.sleep(5)
driver.find_element_by_id("com.taobao.taobao:id/home_searchedit").click()
time.sleep(2)
driver.find_element_by_id("com.taobao.taobao:id/searchEdit").click()
driver.find_element_by_id("com.taobao.taobao:id/searchEdit").send_keys(u"hao")
```

2.5.2 运行脚本

1. 首先要确认手机上的输入法是用的什么输入法，如果默认是中文的输入法，启动后会出现下面情况，无法输入成功



2. 于是可以先把手机上的输入法改成英文的状态，这样就可以输入英文字符串了。那么问题来了，如果想输入中文的字符串呢？如何解决。。。

2.5.3 屏蔽软键盘

1. 通过前面的操作，大概可以知道，在 APP 里面输入字符串是调用的软键盘输入的，有没办法像 selenium 做 web 自动化时候一样，直接 sendkeys 绕过键盘输入呢？

2. 于是可以想办法屏蔽软键盘，只需在 `desired_caps` 设置里面加两个参数（敲黑板，记重点！）

```
# coding=utf-8
from appium import webdriver
import time
desired_caps = {
    'platformName': 'Android',
    'deviceName': '30d4e606',
    'platformVersion': '5.0',
    'appPackage': 'com.taobao.taobao',
    'appActivity': 'com.taobao.tao.welcome.Welcome',
    'unicodeKeyboard': True,
    'resetKeyboard': True
}
```

3. `unicodeKeyboard` 是使用 `unicode` 编码方式发送字符串

4. `resetKeyboard` 是将键盘隐藏起来（公众号版权所有：软件测试部落）

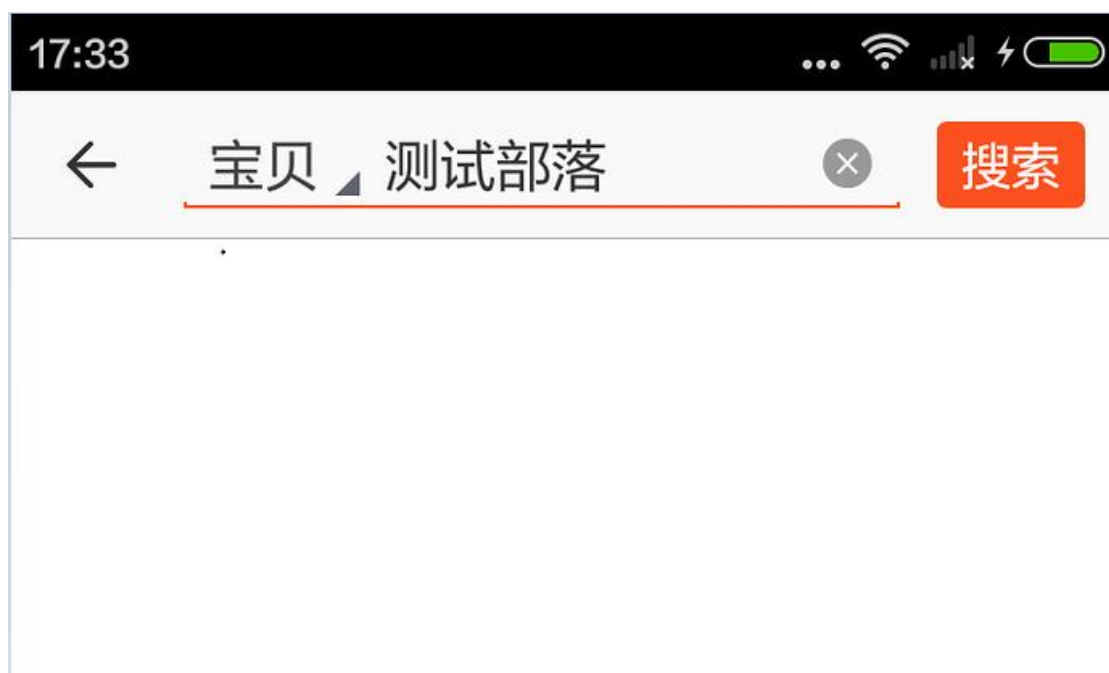
2.5.4 输入中文字符

1. 将上面代码改成输入中文 u “测试部落”，然后运行脚本，在手机上查看结果。（中文前面要加个 u）

```
        'platformVersion': '5.0',
        'appPackage': 'com.taobao.taobao',
        'appActivity': 'com.taobao.tao.welcome.Welcome',
        'unicodeKeyboard': True,
        'resetKeyboard': True
    }

driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired_caps)
# 休眠五秒等待页面加载完成
time.sleep(5)
driver.find_element_by_id("com.taobao.taobao:id/home_searchedit").click()
time.sleep(2)
driver.find_element_by_id("com.taobao.taobao:id/searchEdit").click()
driver.find_element_by_id("com.taobao.taobao:id/searchEdit").send_keys(u"测试部落")
```

2. 运行后手机上显示如下



3. 如果显示中文字符为乱码时:



前面加上“`#--coding:gb18030--`” 如下图所示:

```
1 # --coding: gb18030 --
2 # --coding: utf-8 --
3 from selenium import webdriver
```

2.5.5 还原设置

1. 当运行上面的脚本后，发现手动去输入时候，无法调出软键盘了，如何恢复呢？（公众号版权所有：软件测试部落）

2. 打开手机设置，找到输入法设置选项，会发现默认的输入法被改成 appium 的输入法了。所以只需把这里的设置，恢复成原来输入法就 OK 拉。



1472636268159561.png 713x676 31 KB

2.5.6 最终脚本如下

```
# coding=utf-8
```

```
from appium import webdriver

import time

desired_caps = {

    'platformName': 'Android',

    'deviceName': '30d4e606',

    'platformVersion': '5.0',

    'appPackage': 'com.taobao.taobao',

    'appActivity':

'com.taobao.tao.welcome.Welcome',

    'unicodeKeyboard': True,

    'resetKeyboard': True

}

driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub',
desired_caps)

# 休眠五秒等待页面加载完成

time.sleep(5)

driver.find_element_by_id("com.taobao.taobao:id/home_searchedit").click()

time.sleep(2)

driver.find_element_by_id("com.taobao.taobao:id/searchEdit").click()

driver.find_element_by_id("com.taobao.taobao:id/searchEdit").send_keys(u"yoyoketang")
```

2.6 Appium API

前言：

Appium Python API 全集，可以查看所有的帮助文档

1.contexts

`contexts(self):`

Returns the contexts within the current session.

返回当前会话中的上下文，使用后可以识别 H5 页面的控件

:Usage:

`driver.contexts`

用法 `driver.contexts`

2. current_context

`current_context(self):`

Returns the current context of the current session.

返回当前会话的当前上下文

:Usage:

`driver.current_context`

用法 `driver.current_context`

3. context

`context(self):`

Returns the current context of the current session.

返回当前会话的当前上下文。

:Usage:

`driver.context`

用法 `driver.Context`

4. find_element_by_ios_uiautomation

`find_element_by_ios_uiautomation(self, uia_string):`

Finds an element by uiautomation in iOS.

通过 iOS uiautomation 查找元素

:Args:

- `uia_string` - The element name in the iOS UIAutomation library

:Usage:

```
driver.find_element_by_ios_uiautomation('.elements()[1].cells()[2]')
```

用法 `dr. find_element_by_ios_uiautomation('elements')`

5. find_element_by_accessibility_id

`find_element_by_accessibility_id(self, id):`

Finds an element by accessibility id.

通过 accessibility id 查找元素

:Args:

- `id` - a string corresponding to a recursive element search using the

Id/Name that the native Accessibility options utilize

:Usage:

```
driver.find_element_by_accessibility_id()
```

用法 `driver.find_element_by_accessibility_id('id')`

6.scroll

`scroll(self, origin_el, destination_el):`

Scrolls from one element to another

从元素 origin_el 滚动至元素 destination_el

:Args:

- originalEl - the element from which to begin scrolling
- destinationEl - the element to scroll to

:Usage:

```
driver.scroll(el1, el2)
```

用法 driver.scroll(el1, el2)

7. drag_and_drop

drag_and_drop(self, origin_el, destination_el):

Drag the origin element to the destination element

将元素 origin_el 拖到目标元素 destination_el

:Args:

- originEl - the element to drag
- destinationEl - the element to drag to

用法 driver.drag_and_drop(el1, el2)

8.tap

tap(self, positions, duration=None):

Taps on a particular place with up to five fingers, holding for a certain time

模拟手指点击（最多五个手指），可设置按住时间长度（毫秒）

:Args:

of

- positions - an array of tuples representing the x/y coordinates

the fingers to tap. Length can be up to five.

- duration - (optional) length of time to tap, in ms

:Usage:

```
driver.tap([(100, 20), (100, 60), (100, 100)], 500)
```

用法 `driver.tap([(x, y), (x1, y1)], 500)`

9. swipe

`swipe(self, start_x, start_y, end_x, end_y, duration=None):`

Swipe from one point to another point, for an optional duration.

从 A 点滑动至 B 点，滑动时间为毫秒

:Args:

- start_x - x-coordinate at which to start
- start_y - y-coordinate at which to start
- end_x - x-coordinate at which to stop
- end_y - y-coordinate at which to stop
- duration - (optional) time to take the swipe, in ms.

:Usage:

```
driver.swipe(100, 100, 100, 400)
```

用法 `driver.swipe(x1, y1, x2, y2, 500)`

10.flick

`flick(self, start_x, start_y, end_x, end_y):`

Flick from one point to another point.

按住 A 点后快速滑动至 B 点

:Args:

- start_x - x-coordinate at which to start
- start_y - y-coordinate at which to start
- end_x - x-coordinate at which to stop
- end_y - y-coordinate at which to stop

:Usage:

```
driver.flick(100, 100, 100, 400)
```

用法 driver.flick(x1, y1, x2, y2)

11.pinch

```
pinch(self, element=None, percent=200, steps=50):
```

Pinch on an element a certain amount

在元素上执行模拟双指捏（缩小操作）

:Args:

- element - the element to pinch
- percent - (optional) amount to pinch. Defaults to 200%
- steps - (optional) number of steps in the pinch action

:Usage:

```
driver.pinch(element)
```

用法 driver.pinch(element)

12.zoom

```
zoom(self, element=None, percent=200, steps=50):
```

Zooms in on an element a certain amount

在元素上执行放大操作

:Args:

- element - the element to zoom
- percent - (optional) amount to zoom. Defaults to 200%
- steps - (optional) number of steps in the zoom action

:Usage:

```
driver.zoom(element)
```

用法 driver.zoom(element)

13.reset

reset(self):

Resets the current application on the device.

重置应用(类似删除应用数据)

用法 driver.reset()

14. hide_keyboard

hide_keyboard(self, key_name=None, key=None, strategy=None):

Hides the software keyboard on the device. In iOS, use `key_name` to press a particular key, or `strategy`. In Android, no parameters are used.

隐藏键盘, iOS 使用 key_name 隐藏, 安卓不使用参数

:Args:

- key_name - key to press
- strategy - strategy for closing the keyboard (e.g., `tapOutside`)

```
driver.hide_keyboard()
```

15. keyevent

```
keyevent(self, keycode, metastate=None):
```

Sends a keycode to the device. Android only. Possible keycodes can be found in <http://developer.android.com/reference/android/view/KeyEvent.html>.

发送按键码（安卓仅有），按键码可以上网址中找到

:Args:

- keycode - the keycode to be sent to the device
- metastate - meta information about the keycode being sent

用法 `dr.keyevent('4')`

16. press_keycode

```
press_keycode(self, keycode, metastate=None):
```

Sends a keycode to the device. Android only. Possible keycodes can be found in <http://developer.android.com/reference/android/view/KeyEvent.html>.

发送按键码（安卓仅有），按键码可以上网址中找到

:Args:

- keycode - the keycode to be sent to the device
- metastate - meta information about the keycode being sent

用法 `driver.press_keycode('4')`

`dr.keyevent('4')`与 `driver.press_keycode('4')` 功能实现上是一样的，都是按了返回键

17. long_press_keycode

```
long_press_keycode(self, keycode, metastate=None):
```

Sends a long press of keycode to the device. Android only. Possible keycodes can be

found in
<http://developer.android.com/reference/android/view/KeyEvent.html>.

发送一个长按的按键码（长按某键）

:Args:

- keycode - the keycode to be sent to the device
- metastate - meta information about the keycode being sent

用法 `driver.long_press_keycode('4')`

18. `current_activity`

`current_activity(self):`

Retrieves the current activity on the device.

获取当前的 activity

用法 `print(driver.current_activity())`

19. `wait_activity`

`wait_activity(self, activity, timeout, interval=1):`

Wait for an activity: block until target activity presents or time out.

This is an Android-only method.

等待指定的 activity 出现直到超时，interval 为扫描间隔 1 秒

即每隔几秒获取一次当前的 activity

返回的 True 或 False

:Aargs:

- activity - target activity
- timeout - max wait time, in seconds
- interval - sleep interval between retries, in seconds

用法 `driver.wait_activity('.activity.xxx' , 5, 2)`

20. background_app

`background_app(self, seconds):`

Puts the application in the background on the device for a certain duration.

后台运行 app 多少秒

:Args:

- seconds - the duration for the application to remain in the background

用法 `driver.background_app(5)` 置后台 5 秒后再运行

21.is_app_installed

`is_app_installed(self, bundle_id):`

Checks whether the application specified by `bundle_id` is installed on the device.

检查 app 是否有安装

返回 True or False

:Args:

- bundle_id - the id of the application to query

用法 `driver.is_app_installed("com.xxxx")`

22.install_app

`install_app(self, app_path):`

Install the application found at `app_path` on the device.

安装 app, app_path 为安装包路径

:Args:

- app_path - the local or remote path to the application to install

用法 driver.install_app(app_path)

23.remove_app

remove_app(self, app_id):

Remove the specified application from the device.

删除 app

:Args:

- app_id - the application id to be removed

用法 driver.remove_app(“com.xxx.”)

24.launch_app

launch_app(self):

Start on the device the application specified in the desired capabilities.

启动 app

用法 driver.launch_app()

25.close_app

close_app(self):

Stop the running application, specified in the desired capabilities, on the device.

关闭 app

用法 driver.close_app()

启动和关闭 app 运行好像会出错

26. start_activity

```
start_activity(self, app_package, app_activity, **opts):
```

Opens an arbitrary activity during a test. If the activity belongs to another application, that application is started and the activity is opened.

This is an Android-only method.

在测试过程中打开任意活动。如果活动属于另一个应用程序，该应用程序的启动和活动被打开。

这是一个安卓的方法

:Args:

- app_package - The package containing the activity to start.
- app_activity - The activity to start.
- app_wait_package - Begin automation after this package starts (optional).
- app_wait_activity - Begin automation after this activity starts (optional).
- intent_action - Intent to start (optional).
- intent_category - Intent category to start (optional).
- intent_flags - Flags to send to the intent (optional).
- optional_intent_arguments - Optional arguments to the intent (optional).
- stop_app_on_reset - Should the app be stopped on reset (optional)?

用法 `driver.start_activity(app_package, app_activity)`

27.lock

`lock(self, seconds):`

Lock the device for a certain period of time. iOS only.

锁屏一段时间 iOS 专有

:Args:

- the duration to lock the device, in seconds

用法 `driver.lock()`

28.shake

`shake(self):`

Shake the device.

摇一摇手机

用法 `driver.shake()`

29.open_notifications

`open_notifications(self):`

Open notification shade in Android (API Level 18 and above)

打系统通知栏（仅支持 API 18 以上的安卓系统）

用法 `driver.open_notifications()`

30.network_connection

`network_connection(self):`

Returns an integer bitmask specifying the network connection type.

Android only.

返回网络类型 数值

Possible values are available through the enumeration
`appium.webdriver.ConnectionType`

用法 `driver.network_connection`

31. set_network_connection

`set_network_connection(self, connectionType):`

Sets the network connection type. Android only.

Possible values:

Value (Alias)	Data	Wifi	Airplane Mode

0 (None)	0	0	0
1 (Airplane Mode)	0	0	1
2 (Wifi only)	0	1	0
4 (Data only)	1	0	0
6 (All network on)	1	1	0

These are available through the enumeration
`appium.webdriver.ConnectionType`

设置网络类型

:Args:

- connectionType - a member of the enum
`appium.webdriver.ConnectionType`

用法 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
```

32. available_ime_engines

```
available_ime_engines(self):
```

Get the available input methods for an Android device. Package and activity are returned (e.g.,
['com.android.inputmethod.latin/.LatinIME'])

Android only.

返回安卓设备可用的输入法

用法 `print(driver.available_ime_engines)`

33.is_ime_active

```
is_ime_active(self):
```

Checks whether the device has IME service active. Returns True/False.

Android only.

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())`

34.activate_ime_engine

```
activate_ime_engine(self, engine):
```

Activates the given IME engine on the device.

Android only.

激活安卓设备中的指定输入法，设备可用输入法可以从“available_ime_engines”获取

:Args:

- engine - the package and activity of the IME engine to activate (e. g. ,

'com.android.inputmethod.latin/.LatinIME')

用法

driver.activate_ime_engine(“com.android.inputmethod.latin/.LatinIME”)

35.deactivate_ime_engine

deactivate_ime_engine(self):

Deactivates the currently active IME engine on the device.

Android only.

关闭安卓设备当前的输入法

用法 driver.deactivate_ime_engine()

36.active_ime_engine

active_ime_engine(self):

Returns the activity and package of the currently active IME engine (e. g. ,

'com.android.inputmethod.latin/.LatinIME').

Android only.

返回当前输入法的包名

用法 driver.active_ime_engine

37.toggle_location_services

toggle_location_services(self):

Toggle the location services on the device. Android only.

打开安卓设备上的位置定位设置

用法 `driver.toggle_location_services()`

38.set_location

`set_location(self, latitude, longitude, altitude):`

Set the location of the device

设置设备的经纬度

:Args:

- latitude 纬度 - String or numeric value between -90.0 and 90.00
- longitude 经度 - String or numeric value between -180.0 and 180.0
- altitude 海拔高度- String or numeric value

用法 `driver.set_location(纬度, 经度, 高度)`

39.tag_name

`tag_name(self):`

This element's ``tagName`` property.

返回元素的 tagName 属性

经实践返回的是 class name

用法 `element.tag_name()`

40.text

`text(self):`

The text of the element.

返回元素的文本值

用法 `element.text()`

41.click

`click(self):`

Clicks the element.

点击元素

用法 `element.click()`

42.submit

`submit(self):`

Submits a form.

提交表单

用法 暂无

43.clear

`clear(self):`

Clears the text if it's a text entry element.

清除输入的内容

用法 `element.clear()`

44.get_attribute

`get_attribute(self, name):`

详见[@chenhengjie123](#) 的[超级链接](#)

Gets the given attribute or property of the element.

1、获取 content-desc 的方法为 `get_attribute("name")`，而且还不能保证返回的一定是 content-desc（content-desc 为空时会返回 text 属性值）

2、`get_attribute` 方法不是我们在 `uiautomatorviewer` 看到的所有属性都能获取的（此处的名称均为使用 `get_attribute` 时使用的属性名称）：

可获取的：

字符串类型：

`name`(返回 content-desc 或 text)

`text`(返回 text)

`className`(返回 class，只有 API>=18 才能支持)

`resourceId`(返回 resource-id，只有 API>=18 才能支持)

This method will first try to return the value of a property with the

given name. If a property with that name doesn't exist, it returns the

value of the attribute with the same name. If there's no attribute with

that name, `None` is returned.

Values which are considered truthy, that is equals "true" or "false",

are returned as booleans. All other non-`None` values are returned

as strings. For attributes or properties which do not exist, `None`

is returned.

:Args:

- name - Name of the attribute/property to retrieve.

Example::

```
# Check if the "active" CSS class is applied to an element.
```

```
is_active = "active" in  
target_element.get_attribute("class")
```

用法 暂无

45.is_selected

```
is_selected(self):
```

Returns whether the element is selected.

Can be used to check if a checkbox or radio button is selected.

返回元素是否选择。

可以用来检查一个复选框或单选按钮被选中。

用法 `element.is_slected()`

46.is_enabled

```
is_enabled(self):
```

Returns whether the element is enabled.

返回元素是否可用 True of False

用法 `element.is_enabled()`

47.find_element_by_id

```
find_element_by_id(self, id_):
```

Finds element within this element's children by ID.

通过元素的 ID 定位元素

:Args:

- id_ - ID of child element to locate.

用法 `driver.find_element_by_id(“id”)`

48. find_elements_by_id

`find_elements_by_id(self, id_):`

Finds a list of elements within this element's children by ID.

通过元素 ID 定位, 含有该属性的所有元素

:Args:

- id_ - Id of child element to find.

用法 `driver.find_elements_by_id(“id”)`

49. find_element_by_name

`find_element_by_name(self, name):`

Finds element within this element's children by name.

通过元素 Name 定位（元素的名称属性 text）

:Args:

- name - name property of the element to find.

用法 `driver.find_element_by_name(“name”)`

50. find_elements_by_name

`find_elements_by_name(self, name):`

Finds a list of elements within this element's children by name.

通过元素 Name 定位（元素的名称属性 text），含有该属性的所有元素

:Args:

- name - name property to search for.

用法 `driver.find_element_by_name(“name”)`

51. find_element_by_link_text

```
find_element_by_link_text(self, link_text):
```

Finds element within this element's children by visible link text.

通过元素可见链接文本定位

:Args:

- link_text - Link text string to search for.

用法 driver.find_element_by_link_text(“text”)

52. find_elements_by_link_text

```
find_element_by_link_text(self, link_text):
```

Finds a list of elements within this element's children by visible link text

通过元素可见链接文本定位, 含有该属性的所有元素

:Args:

- link_text - Link text string to search for.

用法 driver.find_elements_by_link_text(“text”)

53. find_element_by_partial_link_text

```
find_element_by_partial_link_text(self, link_text):
```

Finds element within this element's children by partially visible link text.

通过元素部分可见链接文本定位

:Args:

- link_text - Link text string to search for.

driver. find_element_by_partial_link_text(“text”)

54. find_elements_by_partial_link_text

```
find_elements_by_partial_link_text(self, link_text):
```

Finds a list of elements within this element's children by link text.

通过元素部分可见链接文本定位, 含有该属性的所有元素

:Args:

- link_text - Link text string to search for.

```
driver. find_elements_by_partial_link_text( "text" )
```

55. find_element_by_tag_name

```
find_element_by_tag_name(self, name):
```

Finds element within this element's children by tag name.

通过查找 html 的标签名称定位元素

:Args:

- name - name of html tag (eg: h1, a, span)

用法 driver.find_element_by_tag_name("name")

56. find_elements_by_tag_name

```
find_elements_by_tag_name(self, name):
```

Finds a list of elements within this element's children by tag name.

通过查找 html 的标签名称定位所有元素

:Args:

- name - name of html tag (eg: h1, a, span)

用法 driver.find_elements_by_tag_name("name")

57. find_element_by_xpath

```
find_element_by_xpath(self, xpath):
```

Finds element by xpath.

通过 Xpath 定位元素，详细方法可参阅
<http://www.w3school.com.cn/xpath/>

:Args:

xpath - xpath of element to
locate. `"//input[@class='myelement']"`

Note: The base path will be relative to this element's location.

This will select the first link under this element.

::

```
myelement.find_elements_by_xpath("./a")
```

However, this will select the first link on the page.

::

```
myelement.find_elements_by_xpath("//a")
```

用法 `find_element_by_xpath("//*")`

58. find_elements_by_xpath

```
find_elements_by_xpath(self, xpath):
```

Finds elements within the element by xpath.

:Args:

- xpath - xpath locator string.

Note: The base path will be relative to this element's location.

This will select all links under this element.

```
::
```

```
myelement.find_elements_by_xpath("./a")
```

However, this will select all links in the page itself.

```
::
```

```
myelement.find_elements_by_xpath("//a")
```

用法 `find_elements_by_xpath(“//*”)`

59. find_element_by_class_name

```
find_element_by_class_name(self, name):
```

Finds element within this element's children by class name.

通过元素 class name 属性定位元素

```
:Args:
```

```
- name - class name to search for.
```

用法 driver.

```
find_element_by_class_name(“android.widget.LinearLayout”)
```

60. find_elements_by_class_name

```
find_elements_by_class_name(self, name):
```

Finds a list of elements within this element's children by class name.

通过元素 class name 属性定位所有含有该属性的元素

```
:Args:
```

```
- name - class name to search for.
```

用法 driver.

```
find_elements_by_class_name(“android.widget.LinearLayout”)
```

61. find_element_by_css_selector

```
find_element_by_css_selector(self, css_selector):
```

Finds element within this element's children by CSS selector.

通过 CSS 选择器定位元素

:Args:

- css_selector - CSS selector string, ex: 'a.nav#home'

62.send_keys

```
send_keys(self, *value):
```

Simulates typing into the element.

在元素中模拟输入（开启 appium 自带的输入法并配置了 appium 输入法后，可以输入中英文）

:Args:

- value - A string for typing, or setting form fields. For setting

file inputs, this could be a local file path.

Use this to send simple key events or to fill out form fields::

```
form_textfield =  
driver.find_element_by_name('username')
```

```
form_textfield.send_keys("admin")
```

This can also be used to set file inputs.

::

```
file_input = driver.find_element_by_name('profilePic')
```

```
file_input.send_keys("path/to/profilepic.gif")
```

```
# Generally it's better to wrap the file path in one of
the methods
```

```
# in os.path to return the actual path to support cross
OS testing.
```

```
#
file_input.send_keys(os.path.abspath("path/to/profilepic.gif"))

driver.element.send_keys("中英")
```

63. is_displayed

```
is_displayed(self):
```

Whether the element is visible to a user.

此元素用户是否可见。简单地说就是隐藏元素和被控件挡住无法操作的元素（仅限 Selenium，appium 是否实现了类似功能不太确定）这一项都会返回 False

用法 driver.element.is_displayed()

64. location_once_scrolled_into_view

```
location_once_scrolled_into_view(self):
```

```
"""THIS PROPERTY MAY CHANGE WITHOUT WARNING. Use this to discover
where on the screen an element is so that we can click it. This
method
```

```
should cause the element to be scrolled into view.
```

```
Returns the top lefthand corner location on the screen, or ``None``
if
```

```
the element is not visible.
```

```
暂不知道用法
```

```
"""
```

65.size

size(self):

The size of the element.

获取元素的大小（高和宽）

```
new_size["height"] = size["height"]
```

```
new_size["width"] = size["width"]
```

用法 driver.element.size

66. value_of_css_property

value_of_css_property(self, property_name):

The value of a CSS property.

CSS 属性

用法 暂不知

67.location

location(self):

The location of the element in the renderable canvas.

获取元素左上角的坐标

用法 driver.element.location

''' 返回 element 的 x 坐标, int 类型'''

```
driver.element.location.get('x')
```

''' 返回 element 的 y 坐标, int 类型'''

```
driver.element.location.get('y')
```


68.rect

`rect(self):`

A dictionary with the size and location of the element.

元素的大小和位置的字典

69. screenshot_as_base64

`screenshot_as_base64(self):`

Gets the screenshot of the current element as a base64 encoded string.

获取当前元素的截图为 Base64 编码的字符串

:Usage:

```
img_b64 = element.screenshot_as_base64
```

70.execute_script

`execute_script(self, script, *args):`

Synchronously Executes JavaScript in the current window/frame.

在当前窗口/框架（特指 Html 的 iframe ）同步执行 javascript 代码。你可以理解为如果这段代码是睡眠 5 秒，这五秒内主线程的 javascript 不会执行

:Args:

- script: The JavaScript to execute.
- *args: Any applicable arguments for your JavaScript.

:Usage:

```
driver.execute_script('document.title')
```

71.execute_async_script

`execute_async_script(self, script, *args):`

Asynchronously Executes JavaScript in the current window/frame.

插入 javascript 代码，只是这个是异步的，也就是如果你的代码是睡眠 5 秒，那么你只是自己在睡，页面的其他 javascript 代码还是照常执行

:Args:

- script: The JavaScript to execute.
- *args: Any applicable arguments for your JavaScript.

:Usage:

```
driver.execute_async_script('document.title')
```

72.current_url

`current_url(self):`

Gets the URL of the current page.

获取当前页面的网址。

:Usage:

```
driver.current_url
```

用法 `driver.current_url`

73. page_source

`page_source(self):`

Gets the source of the current page.

获取当前页面的源。

:Usage:

`driver.page_source`

74.close

`close(self):`

Closes the current window.

关闭当前窗口

:Usage:

`driver.close()`

75.quit

`quit(self):`

Quits the driver and closes every associated window.

退出脚本运行并关闭每个相关的窗口连接

:Usage:

`driver.quit()`

未完待续！！！！



更多自动化（selenium、python 接口）扫码关注

个人微信公众号: **yoyoketang** (扫二维码关注)

个人博客地址: <http://www.cnblogs.com/yoyoketang/>

QQ 交流群: **512200893**