

# App自动化测试框架Appium

---

## 1 Appium环境搭建

---

### Appium简介

Appium是一个开源测试自动化框架，可用于原生，混合和移动Web应用程序测试。它使用WebDriver协议驱动iOS，Android和Windows应用程序。

- [Appium官网](#)
- [Appium\\_github主页](#)
- [官方中文文档](#)
- [Appium官方论坛](#)

### 1.1 环境搭建步骤

Appium环境安装

#### 第一步-安装 appium 桌面版客户端

群文件中 Appium-1.12.1.dmg(MAC环境)/Appium-windows-1.12.1.exe

#### 第二步-安装 Appium-Python-Client

pip install Appium-Python-Client

pip3 install Appium-Python-Client -i <https://pypi.tuna.tsinghua.edu.cn/simple>

[inghua.edu.cn/simple](https://pypi.tuna.tsinghua.edu.cn/simple)

注意：本版要>=0.29

通过pip list 可以查看本版号

#### 第三步-安装 selenium

pip install selenium

注意：本版要>=3.141.0

#### 第四步-安装 Android SDK

下载Android SDK，并配置环境变量

#### 第五步-安装 夜神模拟器

运行nox\_setup\_v6.2.8.0\_full.exe

MAC 运行 .dmg

#### 第六步-将Android SDK中 platform-tools路径下的adb.exe 复制到模拟器安装路径下的bin目录

替换原有的adb.exe 删除原有的nox\_adb.exe,将新的adb.exe复制一份改名nox\_adb.exe

#### 第七步- 启动设置

**第八步-打开模拟器**，启动appium 桌面版客户端服务  
运行以下代码

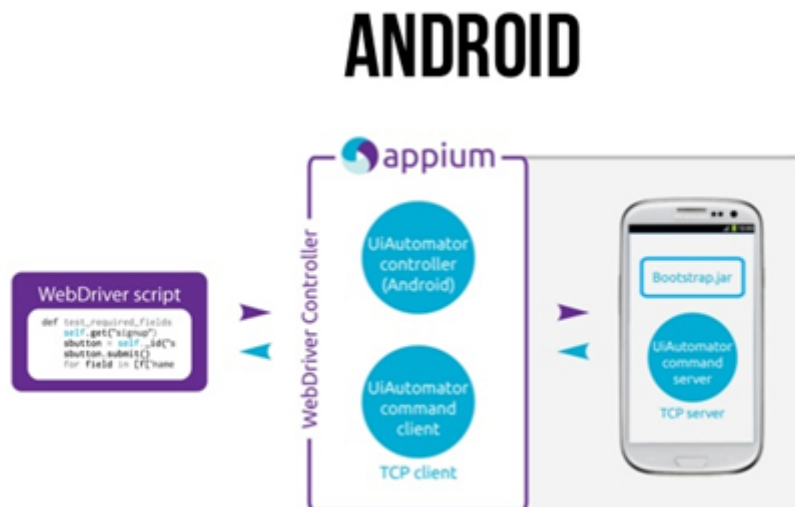
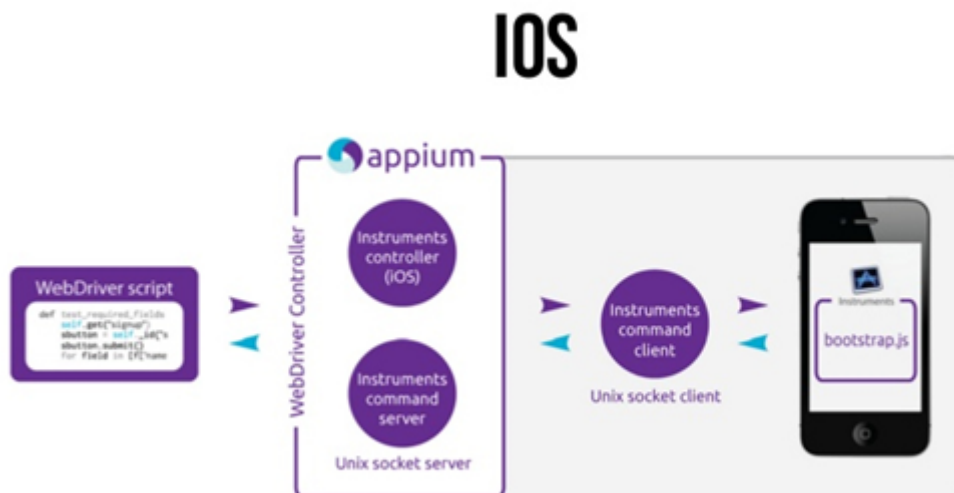
This sample code uses the Appium python client  
pip install Appium-Python-Client  
Then you can paste this into a file and simply run with Python

```
from appium import webdriver

caps = {}
caps["platformName"] = "Android"
caps["platformVersion"] = "5.1.1"
caps["deviceName"] = "127.0.0.1:62001"
caps["appPackage"] = "com.android.settings"
caps["appActivity"] = ".Settings"

driver = webdriver.Remote("http://localhost:4723/wd/hub", caps)
```

## Appium 原理



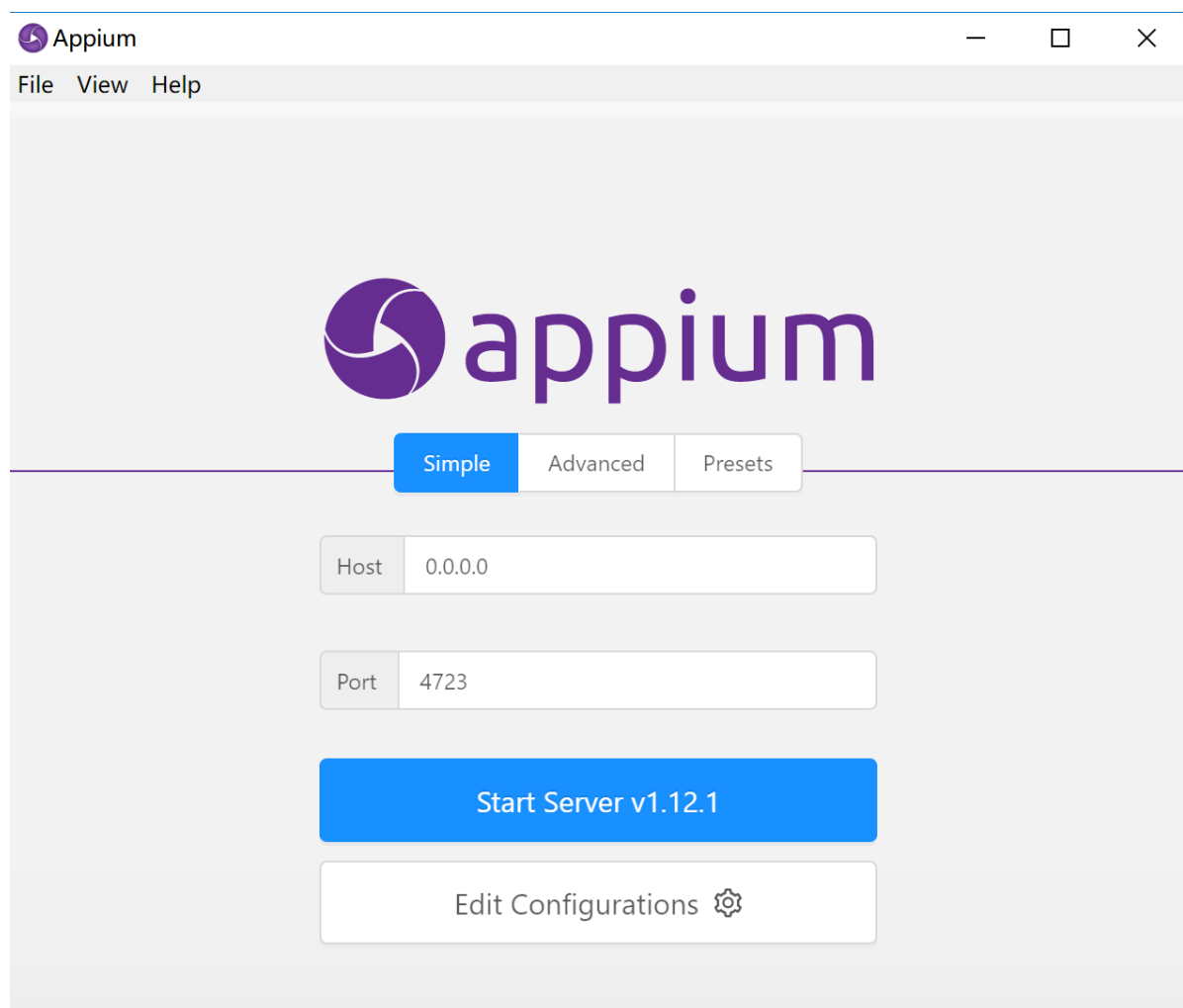
## 2 Capability

### 2.1 Appium Desktop

Appium Desktop是一款适用于Mac, windows和Linux的开源应用程序，它以美观而灵活的用户界面为您提供Appium自动化服务器的强大功能。它是几个Appium相关工具的组合：

Appium Server的图形界面。您可以设置选项，启动/停止服务器，查看日志等...您也不需要使用Node的NPM来安装Appium，因为Node运行时与Appium Desktop捆绑在一起。

您可以使用Inspector查看应用程序的元素，获取有关它们的基本信息，并与它们进行基本的交互



Appium-desktop主界面包含三个菜单Simple,Advanced、Presets

#### Simple

host

设置Appium server的ip地址，本地调试可以将ip地址修改为127.0.0.1

port

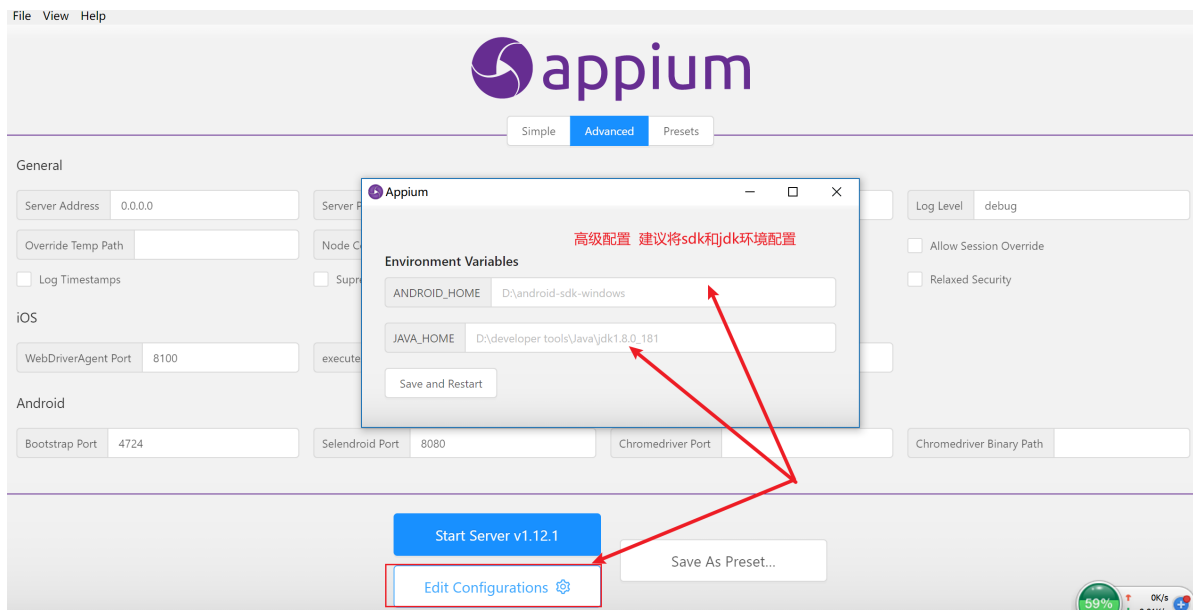
设置端口号，默认是4723不用修改

start server

启动 Appium server

## Advanced

高级参数配置修改，主要是一些Android和iOS设备，log路径等相关信息的配置。



## 第一个appium脚本

```
导包
from appium import webdriver

创建字典，存放启动参数
caps = {}
    手机平台 —— 操作系统 不区分大小写
caps["platformName"] = "Android"
    手机操作系统版本号
caps["platformVersion"] = "5.1.1"
    手机设备号
caps["deviceName"] = "127.0.0.1:62001"
    启动程序包名
caps["appPackage"] = "com.android.settings"
    启动程序的界面名称
caps["appActivity"] = ".Settings"
    连接服务
driver=webdriver.Remote("http://localhost:4723/wd/hub", caps)
```

## 2.2 capability - 参数介绍

### 公共 capability

键	描述	值
automationName	自动化测试的引擎	Appium (默认) 或者 Selendroid
platformName	使用的手机操作系统	iOS, Android, 或者 FirefoxOS
platformVersion	手机操作系统的版本	例如 7.1, 4.4
deviceName	使用的手机或模拟器类型	iPhone Simulator, iPad Simulator, iPhone Retina 4-inch, Android Emulator, Galaxy S4, 等等.... 在 iOS 上, 使用 Instruments 的 instruments -s devices 命令可返回一个有效的设备的列表。在 Andorid 上虽然这个参数目前已被忽略, 但仍然需要添加上该参数
app	本地绝对路径 或 远程 http URL 所指向的一个安装包 ( .ipa, .apk, 或 .zip 文件)。Appium 将其安装到合适的设备上。请注意, 如果您指定了 appPackage 和 appActivity 参数 (见下文), Android 则不需要此参数了。该参数也与 browserName 不兼容。	/abs/path/to/my.apk 或 http://myapp.com/app.ipa
browserName	做自动化时使用的浏览器名字。如果是一个应用则只需填写个空的字符串	'Safari' 对应 iOS, 'Chrome', 'Chromium', 或 'Browser' 则对应 Android
newCommandTimeout	用于客户端在退出或者结束 session 之前, Appium 等待客户端发送一条新命令所花费的时间 (秒为单位)	例如 60
language	(Sim/Emu-only) 为模拟器设置语言	例如 fr
locale	(Sim/Emu-only) 为模拟器设置所在区域	例如 fr_CA
udid	连接真机的唯一设备号	例如 1ae203187fc012g
orientation	(Sim/Emu-only) 模拟器当前的方向	竖屏 或 横屏
autoWebview	直接转换到 Webview 上下文 (context)。默认值为 false	true, false
noReset	在当前 session 下不会重置应用的状态。默认值为 false	true, false
fullReset	(iOS)删除所有的模拟器文件夹。(Android)要清除 app 里的数据, 请将应用卸载才能达到重置应用的效果。在 Android, 在 session 完成之后也会将应用卸载掉。默认值为 false	true, false

### Android独有capability

---

Android 独有

键	描述	值
<u>appActivity</u>	Activity 的名字是指从你的包中所要启动的 Android activity。他通常需要再前面添加 <code>.</code> （例如使用 <code>.MainActivity</code> 代替 <code>MainActivity</code> ）	<code>MainActivity</code> , <code>.Settings</code>
<u>appPackage</u>	运行的 Android 应用的包名	<code>com.example.android.myApp</code> , <code>com.android.settings</code>
<u>appWaitActivity</u>	用于等待启动的 Android Activity 名称	<code>SplashActivity</code>
<u>appWaitPackage</u>	用于等待启动的 Android 应用的包	<code>com.example.android.myApp</code> , <code>com.android.settings</code>
<u>appWaitDuration</u>	用于等待 <code>appWaitActivity</code> 启动的超时时间（以毫秒为单位）（默认值为 <code>20000</code> ）	<code>30000</code>
<u>deviceReadyTimeout</u>	用于等待模拟器或真机准备就绪的超时时间	<code>5</code>
<u>androidCoverage</u>	用于执行测试的 instrumentation 类。 传送 <code>-w</code> 参数到如下命令 <code>adb shell am instrument -e coverage true -w</code>	<code>com.my.Pkg/com.my.Pkg.instrumentation.MyInstrumentation</code>
<u>enablePerformanceLogging</u>	（仅适用于 Chrome 与 webview） 开启 Chromedriver 的性能日志。（默认值为 <code>false</code> ）	<code>true</code> , <code>false</code>
<u>androidDeviceReadyTimeout</u>	用于等待设备在启动应用后准备就绪的超时时间。以秒为单位。	例如 <code>30</code>
<u>androidInstallTimeout</u>	用于等待在设备中安装 apk 所花费的时间（以毫秒为单位）。默认值为 <code>90000</code>	例如 <code>90000</code>

IOS 独有capability

ios 独有

键	描述	值
<u>calendarFormat</u>	（仅支持模拟器）为iOS的模拟器设置日历格式	例如 <code>gregorian</code>
<u>bundleId</u>	被测应用的 bundle ID 。用于在真实设备中启动测试，也用于使用其他需要 bundle ID 的关键词启动测试。在使用 bundle ID 在真实设备上执行测试时，你可以不提供 <code>app</code> 关键词，但你必须提供 <code>udid</code> 。	例如 <code>io.appium.TestApp</code>
<u>udid</u>	连接的真实设备的唯一设备编号 (Unique device identifier)	例如 <code>1ae203187fc012g</code>
<u>launchTimeout</u>	以毫秒为单位，在 Appium 运行失败之前设置一个等待 instruments 的时间	例如 <code>20000</code>
<u>locationServicesEnabled</u>	（仅支持模拟器）强制打开或关闭定位服务。默认值是保持当前模拟器的设定。	<code>true</code> 或 <code>false</code>
<u>locationServicesAuthorized</u>	（仅支持模拟器）通过修改 plist 文件设定是否允许应用使用定位服务，从而避免定位服务的警告出现。默认值是保持当前模拟器的设定。请注意在使用这个关键词时，你同时需要使用 <code>bundleId</code> 关键词来发送你的应用的 bundle ID。	<code>true</code> 或 <code>false</code>
<u>autoAcceptAlerts</u>	当警告弹出的时候，都会自动去点接受。包括隐私访问权限的警告（例如 定位，联系人，照片）。默认值为 <code>false</code> 。不支持基于 <code>XCUITest</code> 的测试。	<code>true</code> 或 <code>false</code>
<u>autoDismissAlerts</u>	当警告弹出的时候，都会自动去点取消。包括隐私访问权限的警告（例如 定位，联系人，照片）。默认值为 <code>false</code> 。不支持基于 <code>XCUITest</code> 的测试。	<code>true</code> 或 <code>false</code>
<u>nativeInstrumentsLib</u>	使用原生 instruments 库（即关闭 instruments-without-delay）。	<code>true</code> 或 <code>false</code>

启动过程【了解】

appium的启动实际上是在本机使用了4723端口开启了一个服务

我们写的 python 代码会访问本机的 appium 服务器，并获取 driver 对象

appium 会将我们的 driver 对象调用的方法转化成 post 请求，提交给appium服务器

appium 通过接收到的 post 请求发送给手机，再由手机进行执行

## 3 Appium元素信息的获取

### 3.1 UIAutomatorViewer 的使用

#### UIAutomatorViewer

用来扫描和分析 Android 应用程序的 UI 控件的工具

#### 使用步骤

1. 进入SDK目录下的目录  
mac 在 tools/bin 目录下，打开 uiautomatorviewer  
windows 在 tools 目录下，打开 uiautomatorviewer.bat
2. 电脑连接真机或打开android模拟器
3. 启动待测试app
4. 点击 uiautomatorviewer 的左上角 Device Screenshot （从左数第二个按钮）
5. 点击希望查看的控件



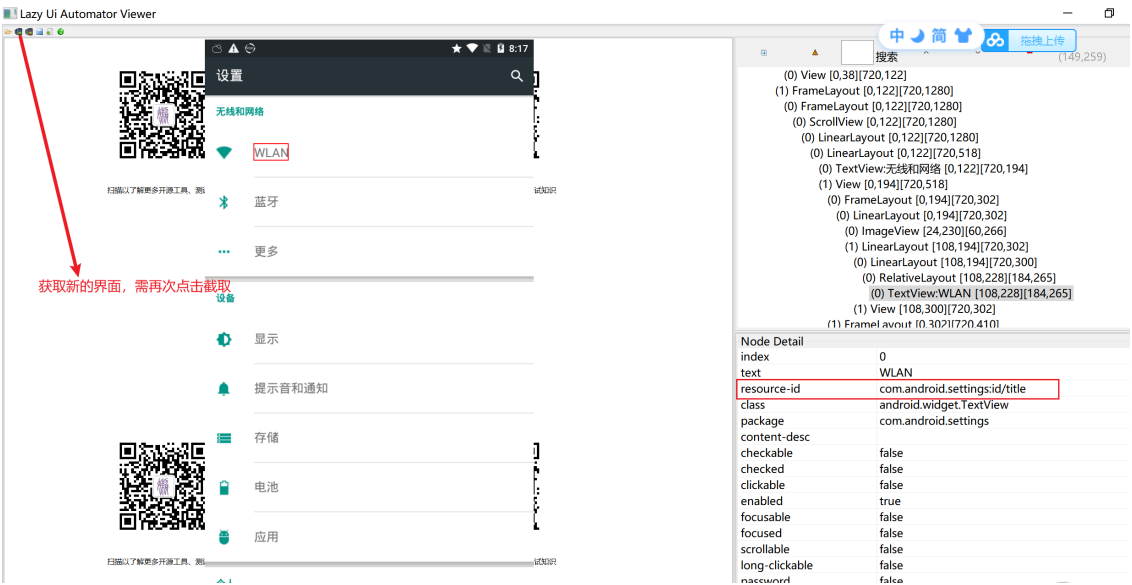
查看右下角 Node Detail 相关信息

示例

查看《设置》应用程序右上角“放大镜”按钮的“resource-id”

打开 uiautomatorviewer

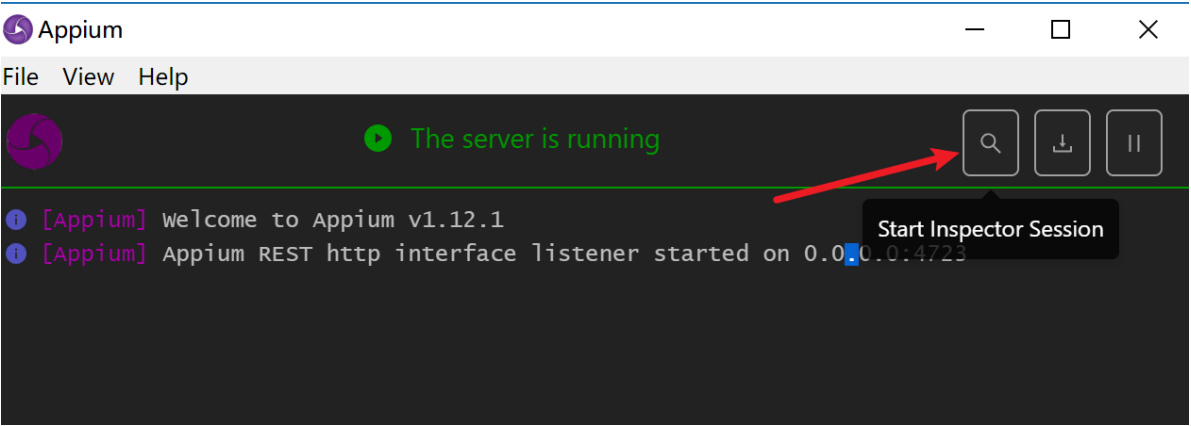
- 打开 android 模拟器
- 启动《设置》应用程序
- 点击 Device Screenshot 按钮
- 点击 “WLAN” 按钮
- 查看 Node Detail 中的 “resource-id” 信息



### 3.2 Appium inspector使用

#### 操作步骤

如图



配置应用信息





```

desired_caps = {}

desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '5.1.1'

desired_caps['deviceName'] = '127.0.0.1:62001'
desired_caps['appPackage'] = 'com.android.settings'
desired_caps['appActivity'] = '.Settings'
desired_caps['unicodeKeyboard'] = True
desired_caps['resetKeyboard'] = True

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

driver.implicitly_wait(5)

    启动设置    点击搜索按钮
driver.find_element_by_id('com.android.settings:id/search').click()

```

### 4.1.2 \*accessibility\_id定位

案例同上，元素值唯一

参数: content-desc

```
driver.find_element_by_accessibility_id('搜索').click()
```

### 4.1.3 xpath定位

案例同上

```
driver.find_element_by_xpath('//*[@content-desc="搜索"]').click()
```

## 4.2 定位一组元素

**class定位**

案例：打印设置文本信息

```

    定位一组元素，打印文本值
    找到共用属性

els = driver.find_elements_by_class_name('android.widget.TextView')

for el in els:
    print(el.text)

```

## 5 Appium定位后元素操作

## 点击操作

方法名：driver.find\_element('xx').click()

示例

打开《通讯录》

点击放大镜按钮

```
C:\Users\ThinkPad>adb shell dumpsys activity|findstr Focused  
mFocusedActivity: ActivityRecord{b69cec0 u0 [com.android.contacts/.activities.PeopleActivity t95]  
mFocusedStack=ActivityStack{15163467 stackId=1, 67 tasks} mLastFocusedStack=ActivityStack{15163467 stackId=1, 67 tasks}
```

## 代码

定位搜索框 点击

```
e11 = driver.find_element_by_accessibility_id("搜索")  
e11.click()
```

## 输入操作

方法名：driver.find\_element('xx').send\_keys('v')

示例

输入框输入内容

代码

定位输入框 输入张三

```
e12 = driver.find_element_by_id("com.android.contacts:id/search_view")  
e12.send_keys("张三")
```

## 清空操作

方法名：driver.find\_element('xx').clear

示例

清空输入框输入内容

代码

输入框清空

```
e12.clear()
```

## 获取元素信息（坐标、文本信息）

方法名：`driver.find_element('xx').text`  
`driver.find_element('xx').get_attribute('xx')`

示例

获取输入框元素的文本值、获取输入框class属性值

代码

```
    获取输入框文本信息
print(e12.text)
    获取输入框的属性信息
print(e12.get_attribute('className'))
```

## 案例代码汇总

```
from appium import webdriver

caps = {}
caps["platformName"] = "Android"
caps["platformVersion"] = "5.1.1"
caps["deviceName"] = "127.0.0.1:62001"
caps["appPackage"] = "com.android.contacts"
caps["appActivity"] = ".activities.PeopleActivity"
caps['unicodekeyboard'] = True
caps['resetkeyboard'] = True

driver = webdriver.Remote("http://127.0.0.1:4723/wd/hub", caps)
    定位搜索框  点击
e11 = driver.find_element_by_accessibility_id("搜索")
e11.click()
    定位输入框  输入张三
e12 = driver.find_element_by_id("com.android.contacts:id/search_view")
e12.send_keys("张三")
    输入框清空
e12.clear()

    获取输入框文本信息
print(e12.text)
    获取输入框的属性信息
print(e12.get_attribute('className'))
    获取输入框的位置信息
print(e12.location)

driver.quit()

driver.find_element()
```

## 6 Appium元素等待

### 6.1 元素等待概念

#### 元素等待

概念：在定位页面元素时如果未找到，会在指定时间内一直等待的过程

#### 为什么要设置元素等待？

网络速度慢

电脑配置低

服务器处理请求慢

#### Selenium中元素等待有几种类型呢？

三种元素等待类型

### 6.2 三种等待类型

#### 6.2.1 Time（强制等待）

案例 通讯录 针对输入框输入进行时间等待5s

```
import time

from appium import webdriver

caps = {}
caps["platformName"] = "Android"
caps["platformVersion"] = "5.1.1"
caps["deviceName"] = "127.0.0.1:62001"
caps["appPackage"] = "com.android.contacts"
caps["appActivity"] = ".activities.PeopleActivity"
caps['unicodeKeyboard'] = True
caps['resetKeyboard'] = True

driver = webdriver.Remote("http://127.0.0.1:4723/wd/hub", caps)

el1 = driver.find_element_by_accessibility_id("搜索")
el1.click()

# 定位搜索框，点击
# 针对输入框输入进行时间等待
# 三种等待方式
# 强制等待
time.sleep(5)

time
```

```
e12 = driver.find_element_by_id("com.android.contacts:id/search_view")

e12.send_keys("张三")
```

## 6.2.2 Implicitly\_Wait() (隐式等待)

### 概念

定位元素时，如果能定位到元素则直接返回该元素，不触发等待；如果不能定位到该元素，则间隔一段时间后再去定位元素；如果在达到最大时长时还没有找到指定元素，则抛出元素不存在的异常 `NoSuchElementException`

- 案例 案例 通讯录 针对输入框输入进行时间等待5s

```
from appium import webdriver

caps = {}
caps["platformName"] = "Android"
caps["platformVersion"] = "5.1.1"
caps["deviceName"] = "127.0.0.1:62001"
caps["appPackage"] = "com.android.contacts"
caps["appActivity"] = ".activities.PeopleActivity"
caps['unicodeKeyboard'] = True
caps['resetKeyboard'] = True

driver = webdriver.Remote("http://127.0.0.1:4723/wd/hub", caps)
    设置全局等待时间
driver.implicitly_wait(5)
    定位搜索框  点击
e11 = driver.find_element_by_accessibility_id("搜索")
e11.click()

    针对输入框输入进行时间等待

e12 = driver.find_element_by_id("com.android.contacts:id/search_view")

e12.send_keys("张三")
```

## 6.2.3 WebDriverWait() (显示等待)

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

导包

```
from selenium.webdriver.support.wait import WebDriverWait
```

```
WebDriverWait(driver, timeout, poll_frequency=0.5)
```

`driver`: 浏览器驱动对象

`timeout`: 超时的时长，单位：秒

`poll_frequency`: 检测间隔时间，默认为0.5秒

调用方法 `until(method)`：直到...时

`method`：函数名称，该函数用来实现对元素的定位

一般使用匿名函数来实现

```
lambda x:driver.find_element_by_name('username')
```

```
e1 = WebDriverWait(driver, timeout=5).until(lambda x: x.find_element_by_name("username"))
# 等待直到找到指定的元素
element = WebDriverWait(driver, timeout=5).until(
    EC.presence_of_element_located((By.ID, 'com.android.contacts:id/search_view')))
```

- 案例 通讯录 针对输入框输入进行时间等待5s

```
from appium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

caps = {}
caps["platformName"] = "Android"
caps["platformVersion"] = "5.1.1"
caps["deviceName"] = "127.0.0.1:62001"
caps["appPackage"] = "com.android.contacts"
caps["appActivity"] = ".activities.PeopleActivity"
caps['unicodekeyboard'] = True
caps['resetkeyboard'] = True

driver = webdriver.Remote("http://127.0.0.1:4723/wd/hub", caps)

    定位搜索框    点击
e11 = driver.find_element_by_accessibility_id("搜索")
e11.click()

    针对输入框输入进行时间等待
    针对元素制定等待时间
e12 = WebDriverWait(driver, timeout=5).until(lambda
x: driver.find_element(By.ID, 'com.android.contacts:id/search_view'))
e12.send_keys("张三")
```

## 7 Appium滑动操作

### 滑动和拖拽事件

#### 应用场景

我们在做自动化测试的时候，有些按钮是需要滑动几次屏幕后才会出现，此时，我们需要使用代码来模拟手指的滑动，也就是我们将来学习的滑动和拖拽事件

## 7.1 swipe 滑动事件

### 概念

从一个坐标位置滑动到另一个坐标位置，只能是两个点之间的滑动。

### 方法名

```
swipe(start_x,start_y,end_x, end_y,duration)
```

从一个坐标位置滑动到另一个坐标位置，只能是两个点之间的滑动

参数：

start\_x: 起点X轴坐标

start\_y: 起点Y轴坐标

end\_x: 终点X轴坐标

end\_y: 终点Y轴坐标

duration: 滑动这个操作一共持续的时间长度，单位：ms

```
driver.swipe(start_x, start_y, end_x, end_y, duration=None)
```

模拟手指从（200, 200），滑动到（200, 800）的位置，持续5秒

核心代码

```
driver.swipe(start_x=200,start_y=800,end_x=200,end_y=200)
```

注\* 由于惯性会影响滑动的范围

距离相同时，持续时间越长(滑的越慢)，惯性越小

持续时间相同时，手指滑动的距离越大（滑的越快），实际滑动的距离也就越大

## 7.2 scroll 滑动事件

### 概念

从一个元素滑动到另一个元素，直到页面自动停止

### 方法名

```
driver.scroll(origin_el, destination_el)
```

### 示例

从“应用”滑动到“WLAN”

核心代码

```
e11 = driver.find_element_by_xpath('//*[text="应用"]')
e12 = driver.find_element_by_xpath('//*[text="WLAN"]')
driver.scroll(e11,e12)
```



## 7.3 drag\_and\_drop 拖拽事件

### 概念

从一个元素滑动到另一个元素，第二个元素替代第一个元素原本屏幕上的位置。

### 方法名

`drag_and_drop(self, origin_el, destination_el)`

从一个元素滑动到另一个元素，第二个元素替代第一个元素原本屏幕上的位置

### 示例

从“应用”滑动到“WLAN”

### 核心代码

```
# drag_and_drop(self, origin_el, destination_el)
driver.drag_and_drop(e11,e12)
```

## 8 Appium绘制九宫格

### 应用场景

TouchAction 可以实现一些针对手势的操作，比如滑动、长按、拖动等。我们可以将这些基本手势组合成一个相对复杂的手势。比如，我们解锁手机或者一些应用软件都有手势解锁的这种方式

### 使用步骤

创建 TouchAction 对象

通过对象调用想执行的手势

通过 perform() 执行动作

## 8.1 轻敲操作

1、**方法名** TouchAction(driver).tap(element=None, x=None, y=None).perform()

### 示例

打开《设置》

轻敲“WLAN”

### 核心代码

```
from appium import webdriver
import time
```

```

from appium.webdriver.common.touch_action import TouchAction
caps = {}
# 平台
caps['platformName'] = 'Android'

caps['platVersion'] = '5.1.1'

caps['deviceName'] = '127.0.0.1:62201'
caps['appPackage'] = 'com.android.settings'
caps['appActivity'] = '.Settings'

#中文
caps['unicodeKeyboard'] = True
#键盘
caps['resetKeyboard'] = True

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

driver.implicitly_wait(3)

action = TouchAction(driver)

# tap 元素定位 点击
action.tap(x=200,y=251).perform()

```

## 2、方法名

使用元素定位方法：action.tap('xxx').perform()

代码：

```
action.tap(driver.find_element_by_xpath('//*[@text="WLAN"]')).perform()
```

## 8.2 按下和抬起操作

### 应用场景

模拟手指一直按下，模拟手指抬起。可以用来组合成轻敲或长按的操作

**方法名** action.press('x').perform()

使用元素定位 长按WLAN

```
action.press(driver.find_element_by_id('android:id/title')).perform()
```

## 8.3 等待操作

### 应用场景

点击WLAN位置长按，暂停 4 秒，并抬起

### 核心代码

```
action.press(driver.find_element_by_id('android:id/title')).wait(4000).release()  
.perform()
```

## 8.4 移动操作

### 应用场景

九宫格是一种比较常见的图案加密方式，目前很多App都支持设置图案锁，Android原生系统也支持设九宫格图案锁屏。那么我们该如何使用Appium进行滑动操作呢

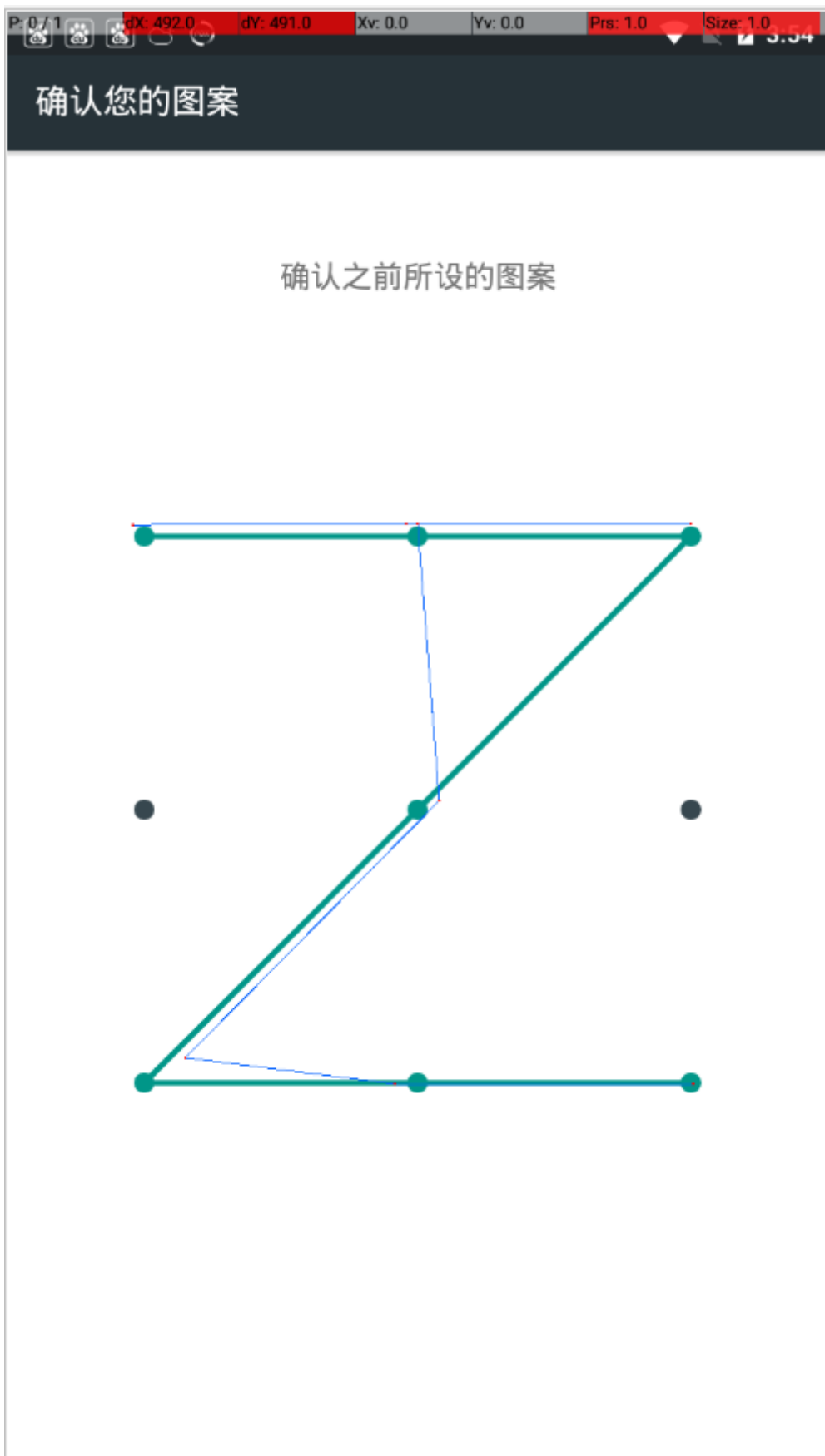
### 方法名

TouchAction(driver).move\_to(el=None, x=None, y=None).perform()

x 和 y 是相对于当前位置的像素坐标

### 示例

在手势解锁中，画一个如下图的案例，并截图



包名界面名为

```
C:\Users\ThinkPad>adb shell dumpsys activity|findstr Focused
mFocusedActivity: ActivityRecord{2edf6f84 u0 com.android.settings/.ConfirmLockPattern t116}
mFocusedStack=ActivityStack{15163467 stackId=1, 87 tasks} mLastFocusedStack=ActivityStack{15163467 stackId=1, 87 tasks}
```

代码示例

```

from appium import webdriver
from appium.webdriver.common.touch_action import TouchAction

class Test:
    def setup(self):
        caps = {}
        # 平台
        caps['platformName'] = 'Android'

        caps['platVersion'] = '5.1.1'

        caps['deviceName'] = '127.0.0.1:62201'
        caps['appPackage'] = 'com.android.settings'
        caps['appActivity'] = '.ChooseLockPattern'

        #中文
        caps['unicodeKeyboard'] = True
        #键盘
        caps['resetKeyboard'] = True

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

        self.action = TouchAction(self.driver)

    def test_case01(self):
        action=self.action
        driver = self.driver

        action.press(x=110,y=451)\
            .move_to(x=350,y=450)\
            .move_to(x=600,y=450)\
            .move_to(x=360,y=450)\
            .move_to(x=379,y=693)\
            .move_to(x=156,y=919)\
            .move_to(x=340,y=942)\
            .move_to(x=602,y=942)\
            .release().perform()
        driver.get_screenshot_as_file('./a.png')

```

## 9 Appium Toast处理

### 应用场景

在日常使用App过程中，经常会看到App界面有一些弹窗提示（如下图所示）这些提示元素出现后等待3秒左右就会自动消失，那么我们该如何获取这些元素文字内容呢？



请输入邮箱

请输入邮箱验证码

获取验证码

确定

点击确定，即表示您已阅读并同意《会员服务条款》

输入邮箱有误

## Toast简介

Android中的Toast是一种简易的消息提示框。当视图显示给用户，在应用程序中显示为浮动。

Toast类的思想就是尽可能不引人注意，同时还向用户显示信息，希望他们看到。而且Toast显示的时间有限，一般3秒左右就消失了。因此使用传统的元素定位工具，我们是无法定位到Toast元素的

## Appium Toast内容获取

Add ability to verify TOAST messages (these can't be interacted with, only text retrieval allowed)

Appium支持识别Toast内容，主要是基于UiAutomator2，因此需要在Capability配置如下参数：

```
desired_caps['automationName']='uiautomator2'
```

安装node.js (使用 npm 或 node 验证)

```
node-v8.11.3-x64.msi(windows) 或 node-v8.10.0.pkg(mac) 进行安装
```

安装cnpm (使用cnpm验证)

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

安装appium-uiautomator2-driver:

```
cnpm install appium-uiautomator2-driver
```

安装成功后可以在 C:\Users\XXXX\node\_modules看到对应的文件:

- \_appium-uiautomator2-driver@1.12.0@appium-uiautomator2-driver
- [\\_appium-uiautomator2-server@1.10.0@appium-uiautomator2-server](#)

npm install -g cnpm --registry=<https://registry.npm.taobao.org> 安装cnpm

mac本需要自主下载两个apk, (appium-uiautomator2-server-v0.1.8.apk 、 appium-uiautomator2-server-debug-androidTest.apk)

需要放在本机 path /usr/local/lib/node\_modules/appium/node\_modules/appium-uiautomator2-driver/uiautomator2/ 目录下

下载地址: <https://github.com/appium/appium-uiautomator2-server/releases>

注意 \*\* 安装过程可能需要对C盘进行读写, 请确保使用的是管理员权限, 网络的原因影响安装

## 测试场景

进入登录界面点击登录, 获取 Toast内容:

toast文本信息获取:

必须通过xpath的方法

```
推荐//*[@class="android.widget.Toast"]
```

```
//*[@contains(@text,"邮箱有误")]
```

## 代码实现

```
from appium import webdriver
from appium.webdriver.common.touch_action import TouchAction
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

caps = {}
caps["platformName"] = "Android"
caps["platformVersion"] = "5.1.1"
caps["deviceName"] = "127.0.0.1:62001"
caps["appPackage"] = "com.flutter_luckin_coffee"
caps["appActivity"] = ".MainActivity"
caps["unicodeKeyboard"] = True
caps["resetKeyboard"] = True
caps['noReset'] = True
caps['unicodeKeyboard'] = True
caps['resetKeyboard'] = True
caps['automationName'] = 'UiAutomator2'
driver = webdriver.Remote("http://localhost:4723/wd/hub", caps)

action = TouchAction(driver)
def find(by,loc):
    return WebDriverWait(driver,timeout=90).until(lambda x:
x.find_element(by,loc))

def click(by,loc):
    return find(by,loc).click()

def input(by,loc,v):
    return find(by,loc).send_keys(v)

def text(by,loc):
    return find(by,loc).text

# 点击我的
click(By.XPATH, '//*[@contains(@text,"我的")]')

# 立即登录

click(By.XPATH, '//*[@text="立即登录"]')
# 点击确定
click(By.XPATH, '//*[@text="确定"]')

# 获取toast提示信息
# toast = find(By.XPATH, '//*[@contains(@text,"邮箱有误")]')
toast = find(By.XPATH, '//*[@class="android.widget.Toast"]')
print(toast.text)
```

## 10 Appium Webview处理



## 应用场景

在混合开发的App中，经常会有内嵌的H5页面。那么这些H5页面元素该如何进行定位操作呢？

### 解决思路

针对这种场景直接使用前面所讲的方法来进行定位是行不通的，因为前面的都是基于Android原生控件进行元素定位，而Web网页是单独的B/S架构，两者的运行环境不同因此需要进行上下文（context）切换，然后对H5页面元素进行定位操作

## context

### 简介

Context的中文翻译为：语境; 上下文; 背景; 环境，在开发中我们经常说“上下文”

## WebView

Android内置webkit内核的高性能浏览器,而WebView则是在这个基础上进行封装后的一个 控件,WebView直译网页视图,我们可以简单的看作一个可以嵌套到界面上的一个浏览器控件!

## 10.1 UC 开发者调试工具及使用

- 下载地址
- <https://plus.ucweb.com/download/?spm=ucplus.11213647.0.0.22e62604mYD3bp#DevTool>
- 参考资料
- <https://plus.ucweb.com/docs/pwa/docs-zh/xy3whu>

### 操作步骤

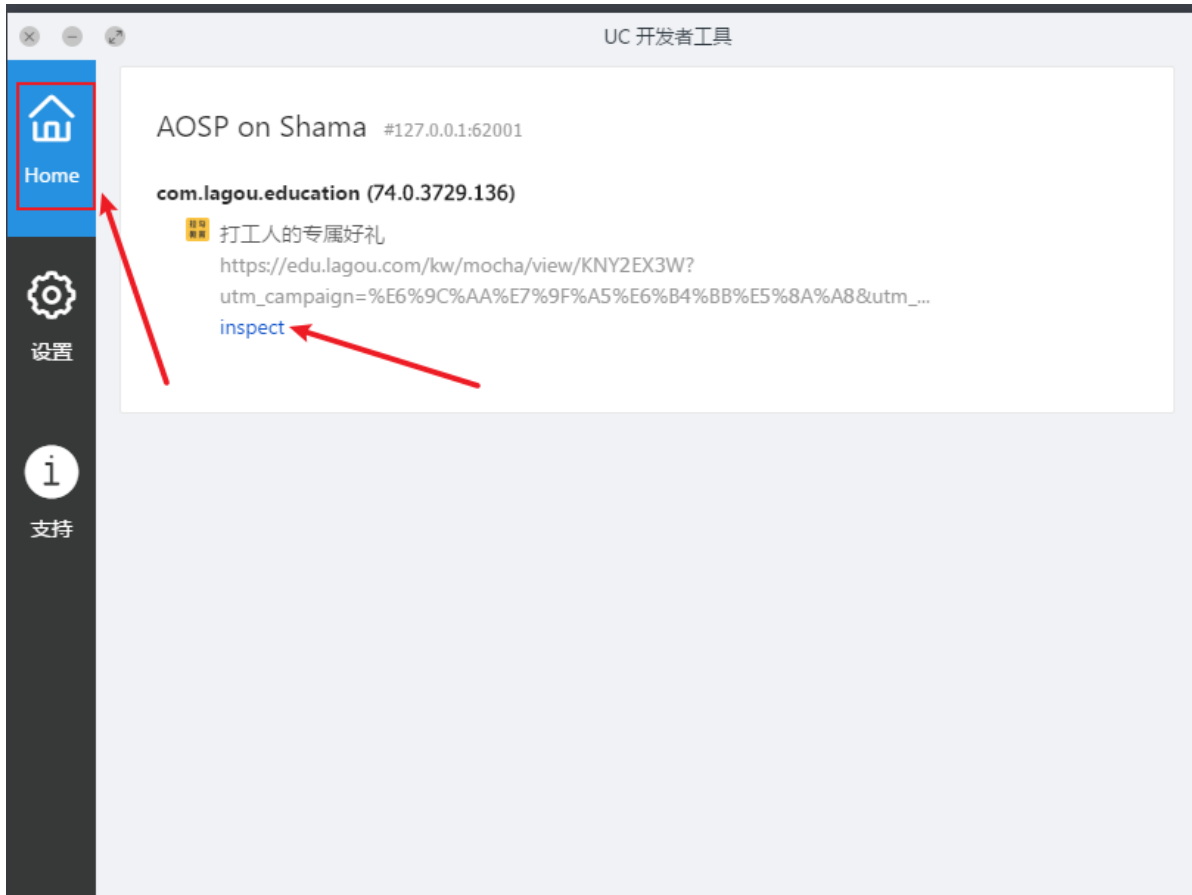
1. 设备与电脑连接，开启USB调试模式，通过adb devices可查看到此设备。（设备系统Android 5.0以上）
2. App Webview开启debug模式
3. 设置勾选本地



#### 4. 执行测试脚本

##### Webview 调试模式检查

打开app对应的h5页面，检查uc开发者工具 Home 右侧是否有提示信息 -- 点击inspect



在自动化脚本中，进入到对应的H5页面，打印输出当前context,如果一直显示为Native App，则webview未开启。

```
D:\python37\python.exe C:/Users/ThinkPad/Desktop  
['NATIVE_APP', 'WEBVIEW_com.lagou.education']
```

## 10.2 H5定位实践案例

### 测试场景

启动拉勾教育App，登录页面-- 首页 点击java课程推广页 -- 领取成长路线

# 拉勾教育



## Java工程师 高薪学习路径

直击大厂用人要求，构建 Java 核心技术能力

- 高并发/高可用场景解决方案
- 亿级数据存储行业最佳实践
- 分布式/微服务架构设计
- 性能分析及调优实战经验





拉勾教育

4月学习季

# Java工程师高薪训练营

- 拉勾背书内推
- 硬核实战技术干货
- 阿里P7技术能力



**15个月**

课程打磨

**6个月**

学习时长

**10+**

一线大厂项目实操

**1000+**

学员口碑认证

**优秀学员**

学员认可推荐

## 豪华版V2.0内容配置

百万学费大补贴

领取成长路线

- 测试设备：夜神模拟器 Android 5.1.1
- PC系统环境: Win10 64
- 测试app: 拉勾教育 V2.2.1
- H5页面地址: [https://edu.lagou.com/growth/sem/java\\_architect.html](https://edu.lagou.com/growth/sem/java_architect.html)

## 需求分析

先进入到H5页面，然后切换到context,再进行相关元素定位操作。

context切换：可以通过contexts()方法来获取到页面的所有context,然后切换到H5页面的context

在H5页面进行元素定位操作

## 获取方法实践

```
contexts=driver.contexts
print(contexts)

#打印结果
['NATIVE_APP', 'WEBVIEW_com.lagou.education']
```

## 代码实现

```
from appium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
import time

desired_caps = {}
# 设备信息
desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '5.1.1'
desired_caps['deviceName'] = '127.0.0.1:62001'
desired_caps['appPackage'] = 'com.lagou.education'
desired_caps['appActivity'] = '.ui.MainActivity'
desired_caps['automationName'] = 'UiAutomator2'

# 输入中文
desired_caps['unicodeKeyboard'] = True
desired_caps['resetKeyboard'] = True
# 是否重置应用 True:不重置 False:重置
# desired_caps['noReset'] = True
# 获取driver
driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)

driver.implicitly_wait(10)

def find(by, loc):
    return WebDriverWait(driver, timeout=60).until(lambda x:
x.find_element(by, loc))

# 点击账号密码登录

find(By.ID, 'com.lagou.education:id/tv_login_by_type').click()

# 输入账号信息
```

```
find(By.ID, 'com.lagou.education:id/et_four_guide_fragment_account').send_keys('18411079273')

driver.find_element_by_id('com.lagou.education:id/et_four_guide_fragment_pwd').send_keys('ai123456')

driver.find_element_by_id('com.lagou.education:id/login_button').click()

# 点击 x 关闭更新

find(By.ID, 'com.lagou.education:id/dlg_cancel').click()

# 跳过个人信息定制
find(By.ID, 'com.lagou.education:id/tv_skip_write').click()

# 获取webview页面

driver.find_element_by_accessibility_id('拉勾教育').click()

# 切换上下文 点击详情页
time.sleep(5)
print(driver.contexts)

driver.switch_to.context('WEBVIEW_com.lagou.education')

# 页面跳转 点击
find(By.XPATH, '//*[@id="app"]/div/div/div[8]/div/div/button').click()
```