**Appium Blaine Summary 20211022**

**一、环境配置**

**安装Appium Desktop**

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

**Python3**

安装python->安装Appium-Python-Client库

可以使用pip命令进行安装pip install Appium-Python-Client,安装完成后在python中运行此段代码from appium import webdriver校验是否安装成功

**JDK**

**Google没有适配最新的JDK版本，最好用1.8**

* 在系统环境变量中创建JAVA\_HOME变量，值为JDK安装的跟目录路径，例如：C:\Program Files\Java\jdk1.8.0\_231
* 在系统环境变量中创建CLASSPATH变量，值为 .;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar
* 在Path变量中添加%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin
* 验证JDK环境配置是否成功，CMD命令行下输入java -version回车，可以看到当前JDK的版本信息，说明JDK已经安装并配置成功

**Android SDK Google不单独提供了，需要下载Android studio**

* 在系统环境变量中创建ANDROID\_HOME变量，值为SDK安装的根目录路径，例如：E:\android\_sdk
* 将%ANDROID\_HOME%\platform-tools加入到环境变量Path中

可以安装appium-doctor检查环境: npm install appium-doctor -g

Source code: <https://github.com/appium/appium-doctor>

文本

描述已自动生成

**二、启动**

安装完成后，启动appium，host和port默认即可，在Edit Configuration中将Android SDK和Java JDK加到环境变量中。

**图形用户界面, 应用程序, Teams

描述已自动生成**

设置好后保存并重启Appium，然后启动服务，出现以下画面，说明你的appium server已经正常启动了，可以开始真机测试了。

**文本

描述已自动生成**

点击Start Inspector Session对进行Desired Capabilities的配置，在JSON Representation中输入以下json数据点击保存进行快速的配置，也可以在左边一项一项的添加。**platformName、platformVersion、deviceName、appPackage、appActivity这五个参数是必填项**。

图形用户界面, 应用程序

描述已自动生成

adb shell getprop ro.build.version.release # 查看Android版本（platformVersion）

adb devices -l # deviceName

adb shell dumpsys activity | findstr "mResume" # appPackage & appActivity

**E.G.**

{

"platformName": "Android", # 声明是ios还是Android系统

"platformVersion": "11", # Android内核版本号

"deviceName": "Redmi\_Note\_8\_Pro", # 连接的设备名称

"appPackage": "com.trainnote.rn", # apk的包名

"appActivity": ".MainActivity", # apk的launcherActivity

"resetKeyboard": True,

"noReset": True # 在开始会话之前不要重置应用程序状态

}

配置好Desired Capabilities后，点击Start Session，app启动运行之后可以在界面上点击选择查看当前页面所有元素控件的详细信息，右侧Selected Element下方有三个按钮。

* **Tap**：执行选中元素的点击操作
* **Send Keys**：为文本框等input对象传值
* **Clear**：清除输入框中的文本

**三、编写自动化脚本**

* 启动session成功运行app之后，点击你想要操作的元素，在右边可以看到这个元素的相关信息，这里我们可以通过id对该元素进行操作，在python中使用find\_element\_by\_id()方法定位该元素，还有很多方法可以定位元素。
* 如：find\_element\_by\_xpath()、driver.find\_element\_by\_name()、driver.find\_element\_by\_partial\_link\_text()等，使用click()方法对元素进行点击操作，使用swip()方法对屏幕进行滑动操作。

图形用户界面, 文本, 应用程序

描述已自动生成

下面是python实现操作的源码展示，运行之后可以看到手机自动完成了启动app、点击、滑动和退出的操作后，说明python脚本运行成功。需要注意的是在完成上一步操作之后由于手机性能的差异，app的响应速度不同，建议等待几秒后再进行下一步操作。

import time

from appium import webdriver

caps = {

"platformName": "Android", # 声明是ios还是Android系统

"platformVersion": "8.1.0", # Android内核版本号

"deviceName": "MI\_5X", # 连接的设备名称

"appPackage": "com.tencent.qqmusic", # apk的包名

"appActivity": ".activity.AppStarterActivity", # apk的launcherActivity

"resetKeyboard": True,

"noReset": True # 在开始会话之前不要重置应用程序状态

}

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

time.sleep(15)

el1 = driver.find\_element\_by\_id("com.tencent.qqmusic:id/clt") # 定位<分类歌单>元素

el1.click() # 点击

time.sleep(5)

driver.swipe(500, 1550, 500, 800) # 从(500, 1500)滑动到(500, 800)

driver.quit() # 退出

## Appium 服务器初始化参数（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

| **键** | **描述** | **值** |
| --- | --- | --- |
| appActivity | Activity 的名字是指从你的包中所要启动的 Android acticity。他通常需要再前面添加. （例如 使用 .MainActivity 代替 MainActivity） | MainActivity, .Settings |
| appPackage | 运行的 Android 应用的包名 | com.example.android.myApp, com.android.settings |
| appWaitActivity | 用于等待启动的 Android Activity 名称 | SplashActivity |
| appWaitPackage | 用于等待启动的 Android 应用的包 | com.example.android.myApp, com.android.settings |
| appWaitDuration | 用于等待 appWaitActivity 启动的超时时间（以毫秒为单位）（默认值为 20000) | 30000 |
| deviceReadyTimeout | 用于等待模拟器或真机准备就绪的超时时间 | 5 |
| androidCoverage | 用于执行测试的 instrumentation 类。 传送 -w 参数到如下命令 adb shell am instrument -e coverage true -w | com.my.Pkg/com.my.Pkg.instrumentation.MyInstrumentation |
| enablePerformanceLogging | （仅适用于 Chrome 与 webview）开启 Chromedriver 的性能日志。（默认值为 false） | true, false |
| androidDeviceReadyTimeout | 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 30 |
| androidInstallTimeout | 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 90000 | 例如 90000 |
| adbPort | 用来连接 ADB 服务器的端口（默认值为 5037） | 5037 |
| androidDeviceSocket | 开发工具的 socket 名称。只有在被测应用是一个使用 Chromium 内核的浏览器时才需要。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。 | 例如 chrome\_devtools\_remote |
| avd | 被启动 avd 的名字 | 例如 api19 |
| avdLaunchTimeout | 用于等待 avd 启动并连接 ADB 的超时时间（以毫秒为单位），默认值为 120000。 | 300000 |
| avdReadyTimeout | 用于等待 avd 完成启动动画的超时时间（以毫秒为单位），默认值为 120000。 | 300000 |
| avdArgs | 启动 avd 时使用的额外参数 | 例如 -netfast |
| useKeystore | 使用自定义的 keystore 给 apk 签名，默认值为 false | true或false |
| keystorePath | 自定义 keystore 的路径, 默认路径为 ~/.android/debug.keystore | 例如 /path/to.keystore |
| keystorePassword | 自定义 keystore 的密码 | 例如 foo |
| keyAlias | key 的别名 | 例如 androiddebugkey |
| keyPassword | key 的密码 | 例如 foo |
| chromedriverExecutable | webdriver 可执行文件的绝对路径（如果 Chromium 内嵌一个自己提供的 webdriver，则应使用他去替换掉 Appium 自带的 chromedriver） | /abs/path/to/webdriver |
| autoWebviewTimeout | 用于等待 Webview 上下文（context）激活的时间（以毫秒为单位）。默认值为 2000 | 例如 4 |
| intentAction | 用于启动 activity 的 intent action（默认值为 android.intent.action.MAIN) | 例如 android.intent.action.MAIN, android.intent.action.VIEW |
| intentCategory | 用于启动 activity 的 intent category。（默认值为 android.intent.category.LAUNCHER) | 例如 android.intent.category.LAUNCHER, android.intent.category.APP\_CONTACTS |
| intentFlags | 用于启动 activity 的标识（flags）（默认值为 0x10200000） | 例如 0x10200000 |
| optionalIntentArguments | 用于启动 activity 的额外 intent 参数。请查看 [Intent 参数](http://developer.android.com/reference/android/content/Intent.html) | 例如 --esn <EXTRA\_KEY>, --ez <EXTRA\_KEY> <EXTRA\_BOOLEAN\_VALUE>, 等等。 |
| dontStopAppOnReset | 在使用 adb 启动应用之前，不要终止被测应用的进程。如果被测应用是被其他钩子(anchor)应用所创建的，设置该参数为 false 后，就允许钩子(anchor)应用的进程在使用 adb 启动被测应用期间仍然存在。换而言之，设置 dontStopAppOnReset 为 true 后，我们在 adb shell am start 的调用中不需要包含 -S标识（flag）。忽略该 capability 或 设置为 false 的话，就需要包含 -S 标识（flag）。默认值为 false | true或false |
| unicodeKeyboard | 使用 Unicode 输入法。 默认值为 false | true或false |
| 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。默认值为 false | true或false |
| noSign | 跳过检查和对应用进行 debug 签名的步骤。仅适用于 UiAutomator，不适用于 selendroid。 默认值为 false | true或false |
| 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 )。 默认值为 false | true 或 false |
| disableAndroidWatchers | 禁用 android 监视器（watchers）。监视器用于见识应用程序的无响应状态（anr）和崩溃（crash），禁用会降低 Android 设备或模拟器的 CPU 使用率。该 capability 仅在使用 UiAutomator 时有效，不适用于 selendroid，默认设置为 false。 | true 或 false |
| chromeOptions | 允许对 ChromeDriver 传 chromeOptions 的参数。了解更多信息请查阅 [chromeOptions](https://sites.google.com/a/chromium.org/chromedriver/capabilities) | chromeOptions: {args: ['--disable-popup-blocking']} |
| recreateChromeDriverSessions | 当移除非 ChromeDriver webview时，终止掉 ChromeDriver 的 session。默认设置为 false | true或false |
| nativeWebScreenshot | 在 web 的上下文（context），使用原生（native）的方法去截图，而不是用过代理的 ChromeDriver。默认值为 false | true或false |
| androidScreenshotPath | 在设备中截图被保存的目录名。默认值为 /data/local/tmp | 例如 /sdcard/screenshots/ |
| autoGrantPermissions | 让Appium自动确定您的应用需要哪些权限，并在安装时将其授予应用。默认设置为 false | true或false |

# 1APP自动化测试神器之Appium-页面元素的查看及定位

https://www.jianshu.com/p/d8c720f4a8ea