
# 2. Appium Desktop与ADB

# 安装工具

开发前，我们需要两个工具：

1. Appium Desktop
2. Android设备（真机或Android Studio）

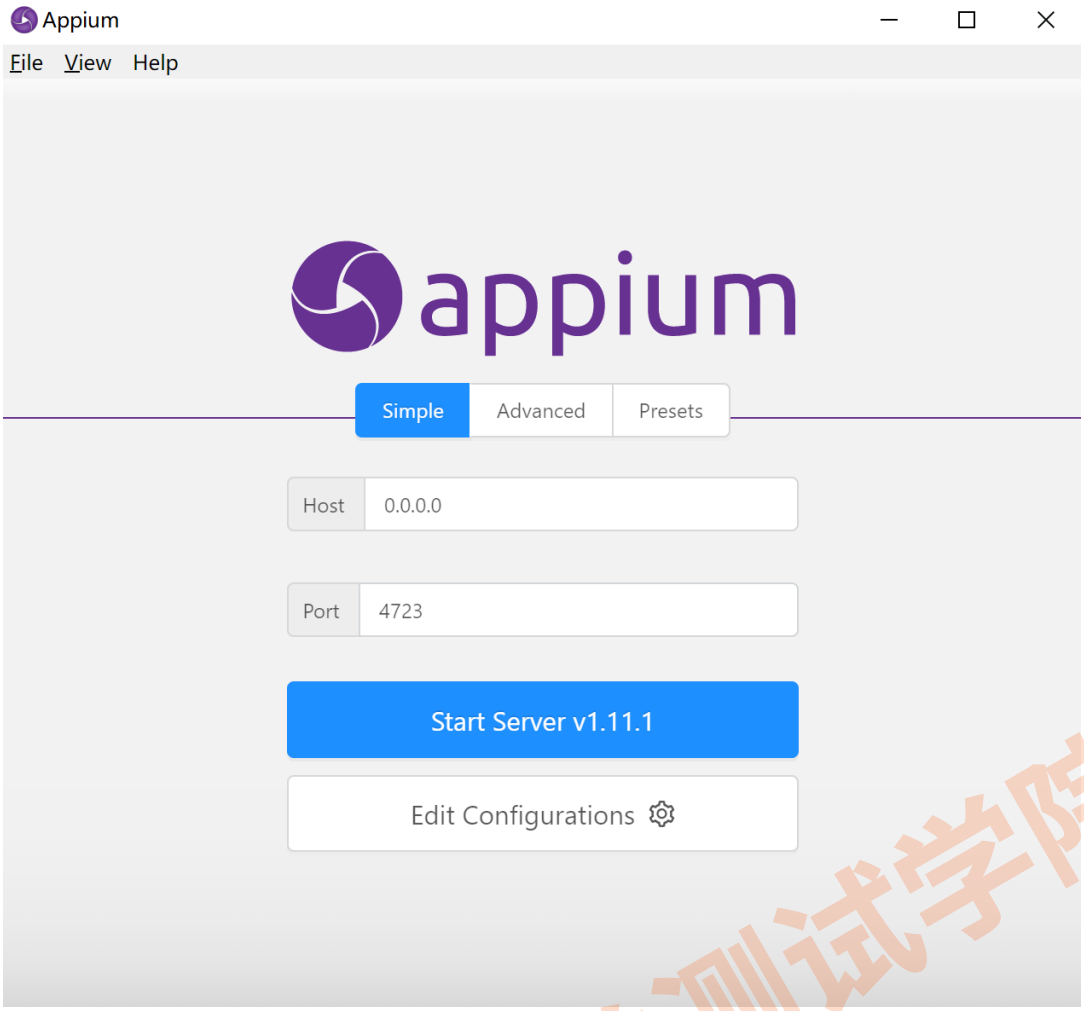
# 下载与安装

- Appium-desktop项目地址：https://github.com/appium/appium-desktop
- 下载地址：https://github.com/appium/appium-desktop/releases

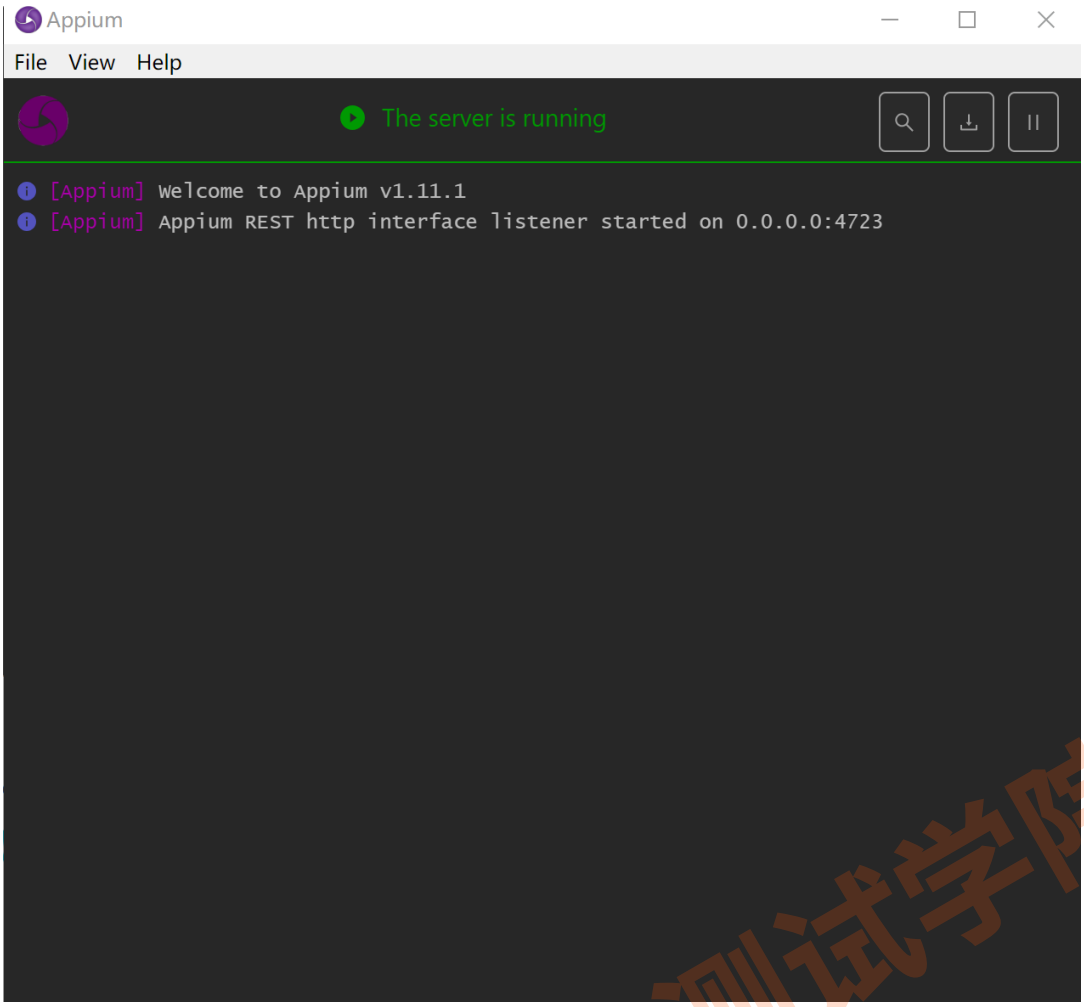
根据自己的平台选择相关的包进行下载。windows选择 appium-desktop-Setup-xxxx.exe 文件进行下载。双击exe文件，全程不用任何设置，等待安装完成即可。

## 环境配置

Appium Desktop既包含了**Appium-server**又包含了一些有用的工具(**Inspector**)。安装完成后会生成一个紫色的appium图标，打开它。




映入眼帘的是**Appium-Server**，默认了监控的host和port，我们**点击Start Server v1.11.1**来启动它。



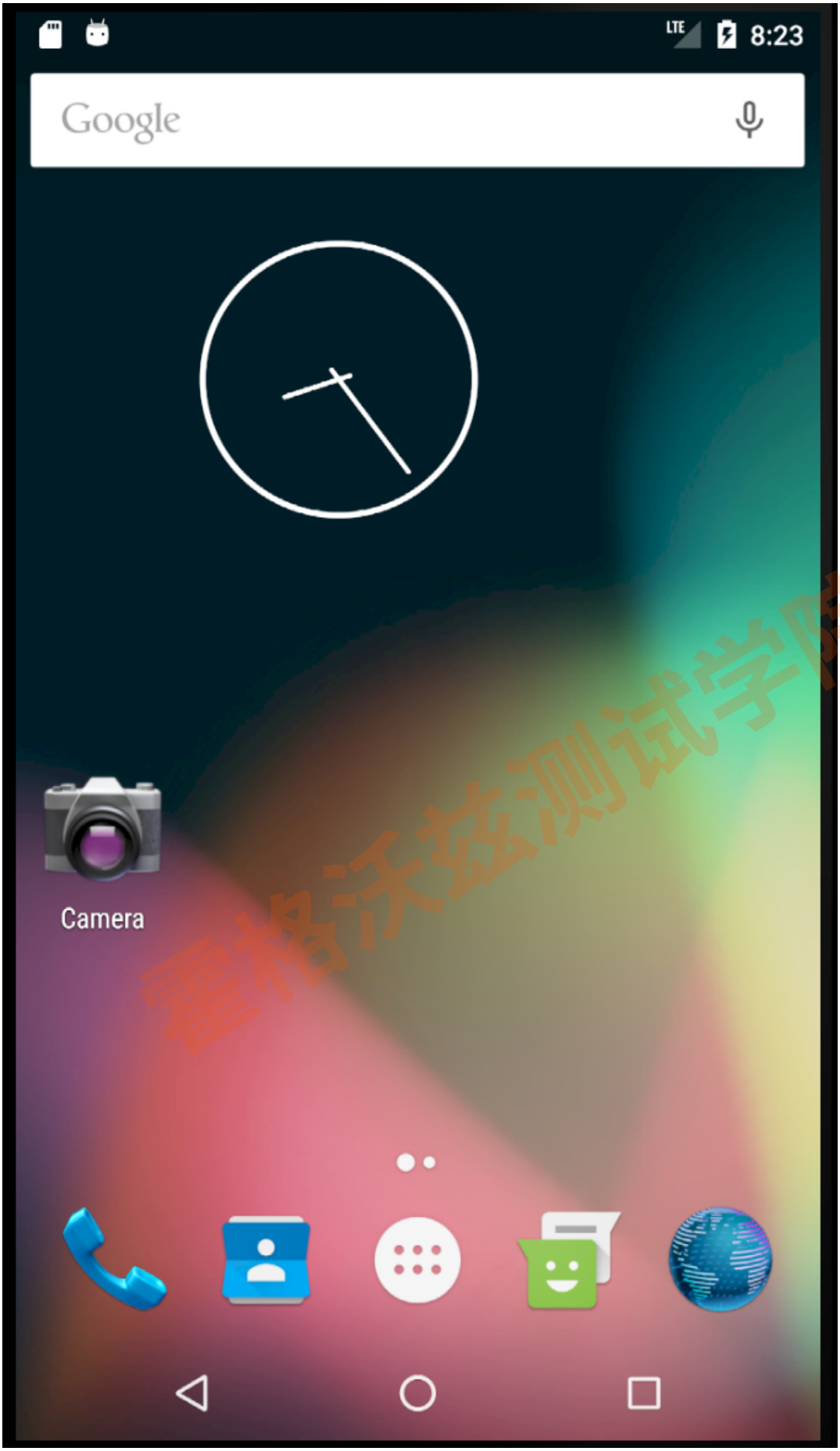

成功的启动了Appium-Server，现在连接你的**移动设备**

- 有真机的同学：配置SDK环境，然后用数据线连接到电脑，开启开发者模式，然后点击允许USB调试。
- 没有真机的同学：下载Android Studio，新建虚拟设备

安装完成后还需要配置adb环境，adb的全称为Android Debug Bridge，起到调试桥的作用，通过adb，可以让用户在电脑上控制Android设备，对手机进行全面操作。借助adb工具，我们可以管理设备或手机模拟器的状态，还可以进行很多手机操作，如安装软件、卸载软件、系统升级、运行shell命令等等。android studio的SDK中带有adb工具，我们只需加入环境变量即可。
我的adb所在目录：

> C:\Users\RuotongYu\AppData\Local\Android\Sdk\platform-tools

配置完成后启动你的设备



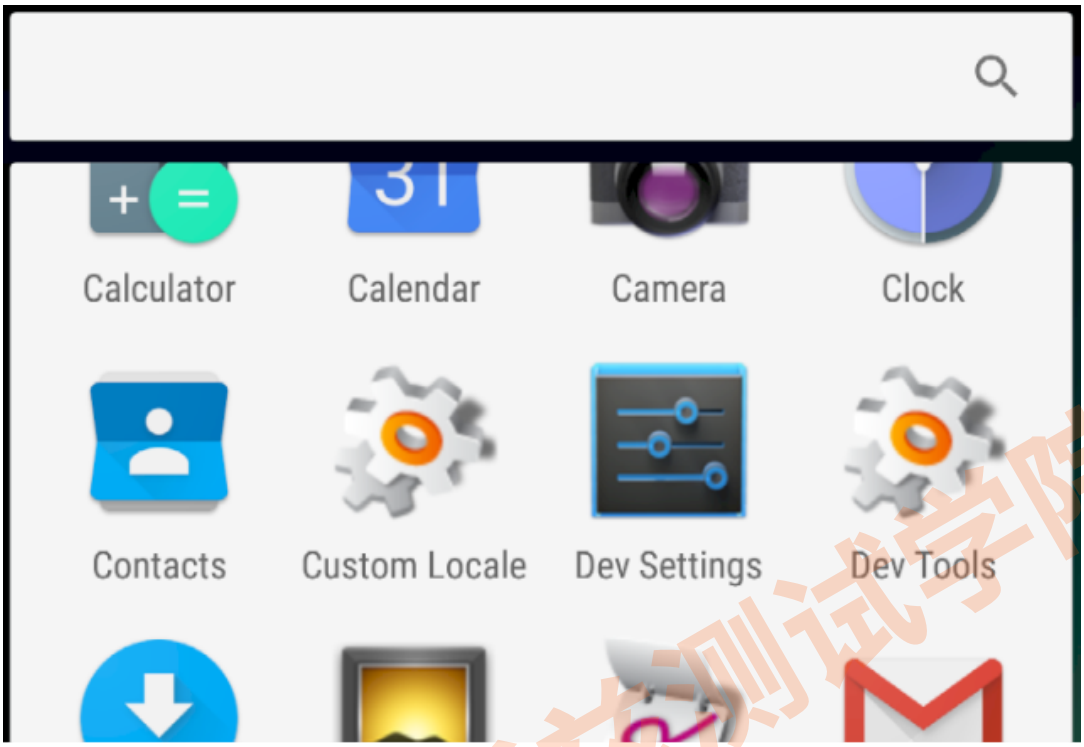



以后的教学要用到**雪球app**，下面进行安装：

- 下载地址：https://sj.qq.com/myapp/detail.htm?apkName=com.xueqiu.android

下载到本地后，可以利用adb进行安装： 打开命令行，输入adb install xxxx.apk。其中，xxxx.apk是你文件所在的绝对路径，否则在安装的时候，就会出现找不到安装包的情况报错。

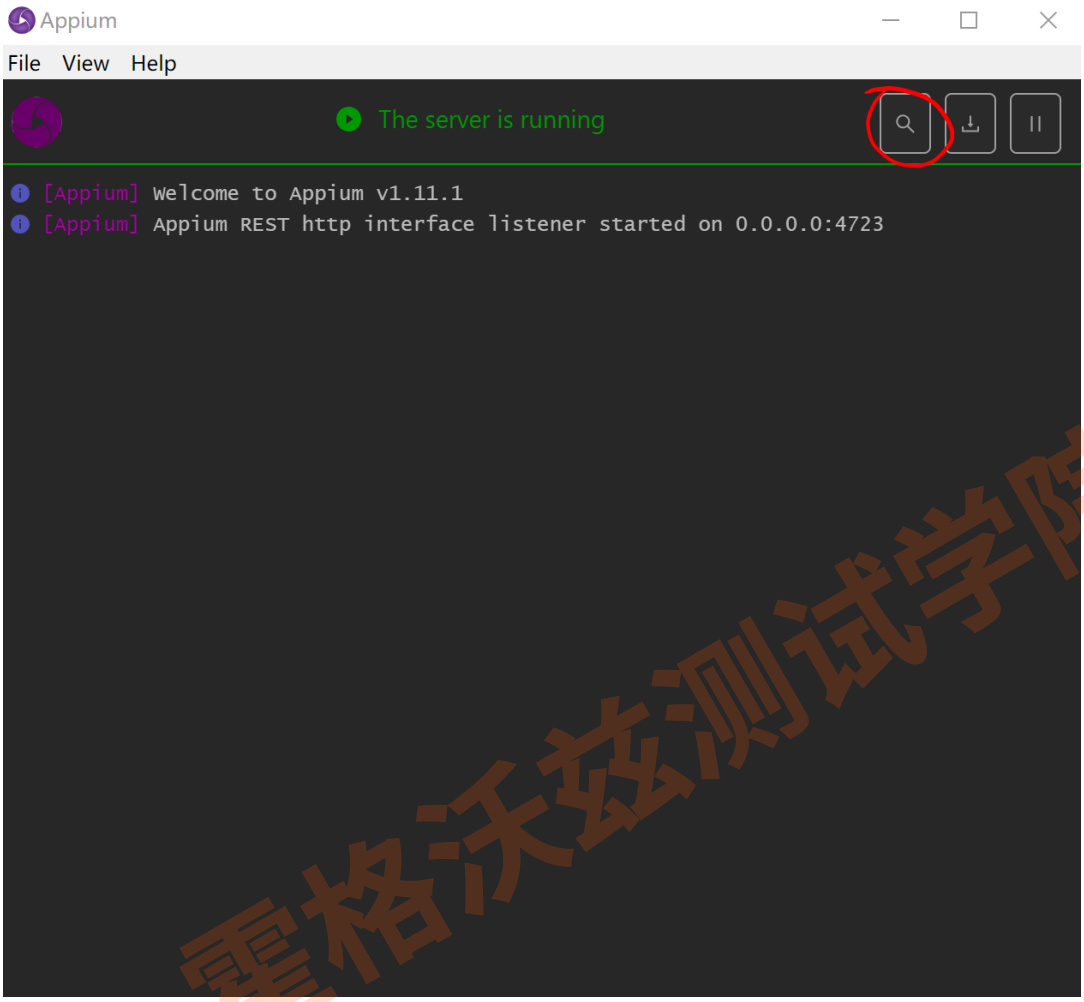
```
adb install D:\下载\com.xueqiu.android_11.17_203.apk
```



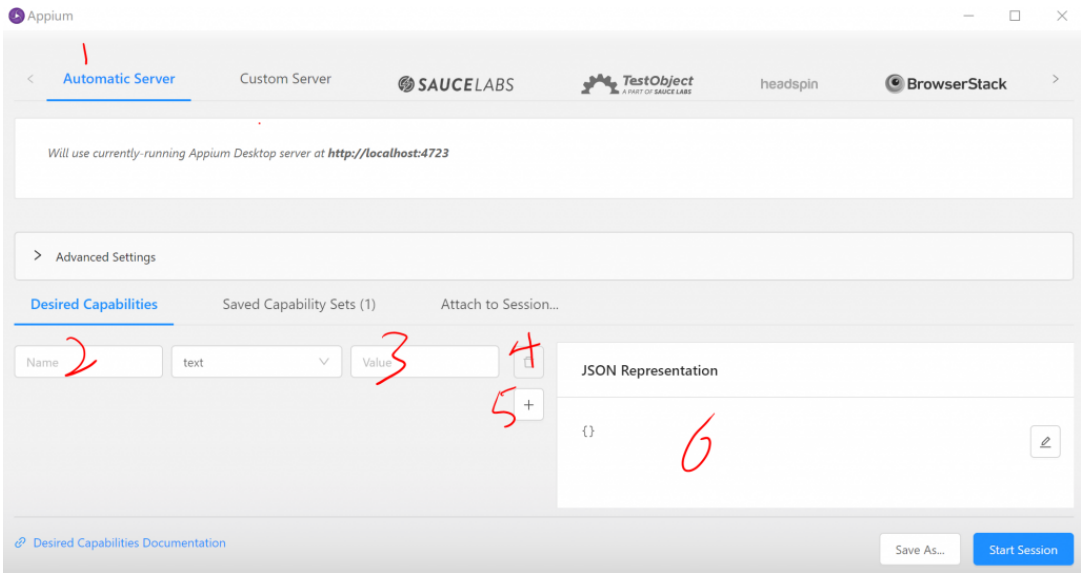
安装完成后，会看到雪球app帅气的身影。

# Inspector工具的使用

使用Inspector前，请确保打开了Appium-Server和android设备。

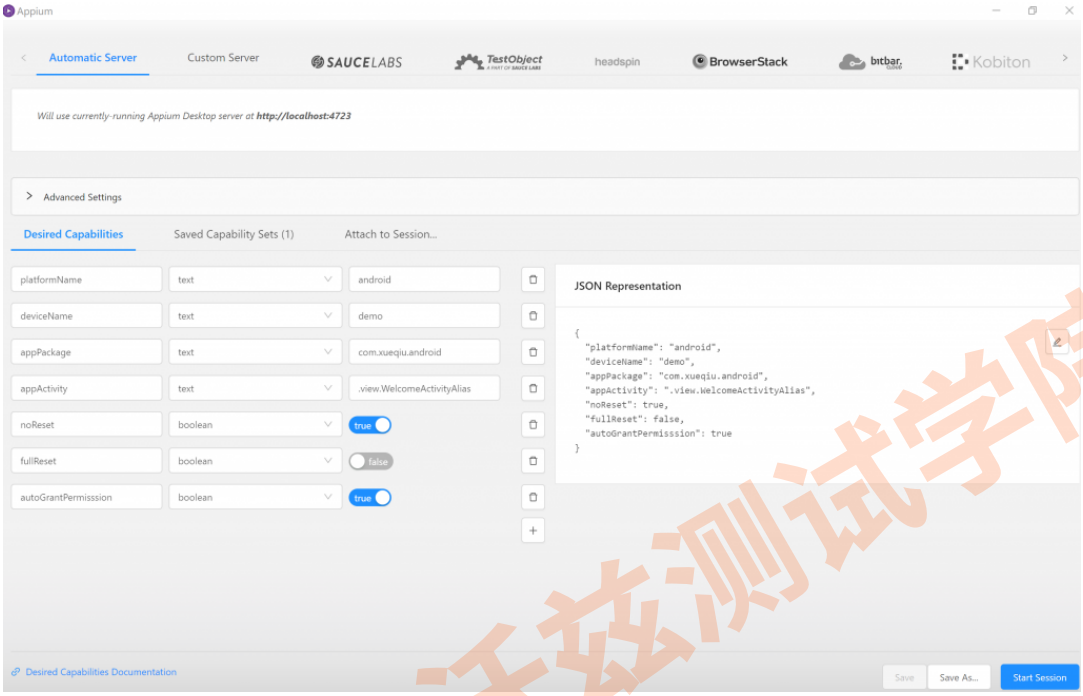



我们点击红圈来启动Inspector。



1. 目前我们只会用到这个工具来审查元素
2. 字段名（后面可选字段类型）
3. 字段值
4. 删除字段
5. 增加新的字段
6. 对应JSON表示

接下来，我们追加下面几个字段：




这些字段可以从下面的表格中查询到，需要注意的是，你必须填写`platformName`,`deviceName`,`appPackage`和`appActivity`字段，不然会报错。

| Appium 服务关键字 | | |
|---|---|---|
| 关键字 | 描述 | 实例 |
| automationName | 你想使用的自动化测试引擎 | Appium (默认) 或 Selendroid |
| platformName | 你要测试的手机操作系统 | iOS, Android, 或 FirefoxOS |
| platformVersion | 手机操作系统版本 | 例如：7.1，4.4 |
| deviceName | 使用的手机类型或模拟器类型 | iPhone Simulator, iPad Simulator, iPhone Retina 4-inch, Android Emulator, Galaxy S4, 等。在 iOS 上，这个关键字的值必须是使用 instruments -s devices 得到的可使用的设备名称之一。在 Android 上，这个关键字目前不起作用。 |
| | .ipa or .apk文件所在的本地绝对路径或者远程路径,也可以是一个包括两者之一的.zip, | |

| Android特有 | | |
|---|---|---|
| 关键字 | 描述 | 实例 |
| 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 |
| appPackage | 你想运行的Android应用的包名 | 比如com.example.android.myApp, com.android.settings |
| appWaitActivity | 你想要等待启动的 Android Activity 名称 | SplashActivity |
| deviceReadyTimeout | 设置等待一个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。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 Chromedriver 的性能日志。 (默认 false) | true, false |
| 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 |
| 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 |
| avd | 需要启动的 AVD (安卓虚拟设备) 名称。 | 如 api19 |
| 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 |
| 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 |
| avdArgs | 启动 AVD 时需要加入的额外的参数。 | 如 -netfast |
| 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 |
| keystorePath | 自定义 keystore 的路径。默认： ~/.android/debug.keystore | 如 /path/to.keystore |
| keystorePassword | 自定义 keystore 的密码。 | 如 foo |
| keyAlias / keyPassword | key 的别名/ key 的密码 | 如 androiddebugkey / 如 foo |
| 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 |
| autoWebviewTimeout | 以毫秒为单位，等待 Webview 上下文激活的时间。默认值 2000 | 如 4 |
| 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 |
| 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 |
| intentFlags | 用于启动 activity 的标识 ( flags ) (默认值 0x10200000) | 如 0x10200000 |
| optionalIntentArguments | 用于启动 activity 的额外 intent 参数。请查看 Intent 参数 | 如 --esn <EXTRA_KEY>, --ez <EXTRA_KEY> <EXTRA_BOOLEAN_VALUE> |
| 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 |
| unicodeKeyboard | 使用 Unicode 输入法。默认值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 false | true 或 false |
| 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 | 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 false | true 或 false |

我相信很多人会有疑问，如何获取app的appPackage和appActivity呢？这个下面ADB命令会讲，本节先抄写我的即可，然后点击`Start session`

> 雪球的appPackage和appActivity分别为："com.xueqiu.android"和".view.WelcomeActivityAlias"

如果你使用的是真机，那么你的设备会有很多的权限申请，点击同意即可。下面进入主页面：


*下面是对以上标号的字段进行的说明，请对照图片上的数字观看：*

## 1. 选择元素模式

在选择元素模式下，所有的页面元素被解析成了**DOM结构**，点击元素，右边会显示元素的相应信息。你可以用tap来模拟点击操作，用send keys模拟键盘输入操作，用clear清除输入的内容

> DOM结构：在网页上，组织页面（或文档）的对象被组织在一个树形结构中，用来表示文档中对象的标准模型就称为DOM—-百度百科

## 2. 滑动模式

点击上方按钮进入**滑动模式**后：

分**前后**点击1和2处，可以实现**翻页滑动**功能。

## 3. 坐标点击模式

点击上方按钮进入**坐标点击模式**后

鼠标可以**直接点击(tap)** 元素，实现坐标点击，比如点击"港股"，会进入港股的相应界面。

## 4. 后退

相当于安卓手机上的back按键。

## 5. 刷新

刷新Inspector页面：如果发现元素没有找到，优先检查Inspector的页面与手机页面是否相同，如果不相同，点击此处的刷新与手机同步。

## 6. 录制模式

点击录制后，会把你对元素的操作转换成代码形式，你可以将这些代码拷贝到相应的ide中进行编译运行。

我们点击1开始录制，然后选择页面中的元素2，最后点击3(tap)，注意观察窗口4中的情况。

此时页面1是进行tap操作后的页面，你可以在2处选择转换的语言，点击按钮3可以显示代码对应的依赖，如果想复制代码，你可以点击4，点击5为清空代码。
借助录制功能，即使不懂代码，你也可以轻松的完成自动化操作，但是在实际过程中，自动生成的代码**又臭又长**，我们要借鉴其中的**语法**，学习对应api用法。

## 7. 元素搜索

元素搜索是一个很重要的功能，简单来说，是检验你是否**定位**到这个元素。具体用法如下：

比如我们想定位到元素1：

1. 点击元素1
2. 在窗口2处找到其相应的信息，比如xpath
3. 点击3搜索

在弹出的窗口，点击1可以更换检索方式，强烈推荐xpath，（思寒老师在课上有讲），下一节会讲解xpath用法。你可以在2处输入内容。

在对应位置输入后点击**Search**。

> 注意：当我们的雪球app页面不同时，查找的元素可能不一样

这里查找到了元素1，点击它，你会发现位置2处的元素被点亮，说明我们找对了!
下面有一些按钮，与之前讲的tap,send keys和clear大同小异。

## 8. 复制xm代码

复制DOM结构处的代码。

## 9. 关闭Inspector

退出Inspector。

# ADB命令

# 一，ADB是什么?

ADB的全称为Android Debug Bridge，即**调试桥**，方便**调试设备**或调试开发的Android APP。ADB是**Android Sdk**里的一个工具, 用这个工具可以直接操作管理android模拟器或者真实的android设备。你可以在Android SDK/platform-tools中找到 adb 工具或下载ADB。上一文中，我们使用了adb install这一命令，本文将介绍更多常用的adb命令，这些命令都是霍格沃兹教学中常用的命令。

# 二，ADB语法

adb的语法也很简单，与常见的DOS命令一样：

```
adb [-d|-e|-s <serial-number>] <command>
```

其中的[-d|-e|-s <serial-number>] 是对多设备的操作，我们通常只用一个安卓设备，因此忽略即可，直接使用下面的格式:

```
adb <command>
```

请大家认真练习下面的命令，一些命令有我精心的注解，这些注解都是开发中常见的问题。

# 三，ADB命令

## 3.1 adb相关

### 3.1.1 查看adb版本信息

```
adb version
```

### 3.1.2 启动adb

```
adb start-server
```

一般**无需手动**执行此命令，在运行adb 命令时若发现adb server 没有启动会自动调起。

### 3.1.3 停止adb

```
adb kill-server
```

模拟器在运行一段时间后，adb服务有可能（在Windows进程中可找到这个服务，该服务用来为模拟器或通过USB数据线连接的真机服务）会出现异常。这时需要重新对adb服务关闭和重启，因此启动adb和停止adb将成为完美的cp。

### 3.1.4 以root权限运行adb

```
adb root
```

adb root是在手机系统上运行修改过的adb指令，来达到临时root的效果。

### 3.1.5 指定adb server的网络端口

```
adb -P <port> start-server
```

ADB的默认端口为 5037

### 3.1.6 查询已连接设备的情况

```
adb devices
```



## 3.2 应用管理

### 3.2.1 查看应用列表

```
adb shell pm list packages [-f] [-d] [-e] [-s] [-3] [-i] [-u] [--user USER_ID]
```

adb shell pm list packages 后面可以跟一些可选参数进行过滤查看不同的列表，可用参数及含义如下：

| 参数 | 显示列表 |
| --- | --- |
| -f | 显示应用关联的 apk 文件 |

| 参数 | 显示列表 |
| --- | --- |
| -d | 只显示 disabled 的应用 |
| -e | 只显示 enabled 的应用 |
| -s | 只显示系统应用 |
| -3 | 只显示第三方应用 |
| -i | 显示应用的 installer |
| -u | 包含已卸载应用 |
| <filter> | 包名包含 <filter> 字符串 |

### 3.2.2 安装应用

```
adb install [-l] [-r] [-t] [-s] [-d] [-g] <apk-file>
```

adb install 后面可以跟一些可选参数来控制安装 APK 的行为，可用参数及含义如下：

| 参数 | 含义 |
| --- | --- |
| -l | 将应用安装到保护目录 /mnt/asec |
| -r | 允许覆盖安装 |
| -t | 允许安装 AndroidManifest.xml 里 application 指定 android:testOnly="true 的应用 |
| -s | 将应用安装到 sdcard |
| -d | 允许降级覆盖安装 |
| -g | 授予所有运行时权限 |

实际上，adb install分三步完成：

1. push apk文件到/data/local/tmp
2. 调用pm install安装
3. 删除/data/local/tmp下对应的apk文件

### 3.2.3 卸载应用

```
adb uninstall [-k] <package-name>
```

<package-name> 表示应用的包名，-k 参数可选，表示卸载应用但**保留数据**和**缓存目录**

### 3.2.4 清除应用数据及缓存

```
adb shell pm clear <package-name>
```

<package-name>表示应用名包，这条命令的效果相当于在设置里的应用信息界面点击了「清除缓存」和「清除数据」

### 3.2.5 获取日志

```
adb logcat
```

### 3.2.6 获取app的包

```
adb logcat | grep -i displayed
```

这条命令在windows下可能出现问题：

将命令改写为

```
adb shell "logcat | grep -i displayed"
```

通常来说，出现的第一个displayed就是程序入口。

有些app的包名无法通过logcat找到，还可以使用下面的命令：

```
aapt dump badging <package> | grep launchable-activity:
```

本命令是从app安装包中获取包名

### 3.2.7 启动app

```
adb shell am start -W -S -n <app package>
```

其中app package是指某个app的包名/包名.活动名称，以雪球为例：

## 相关链接

霍格沃兹测试学院官网首页:https://testing-studio.com
Appium官网:http://appium.io/
Appium Client相关文档:https://github.com/appium/appium/blob/master/docs/en/about-appium/appium-clients.md
brew官网:https://brew.sh/
node官网:https://nodejs.org/zh-cn/
淘宝源NPM官网:https://npm.taobao.org/