目录

主流跨平台自动化框架：Appium

- 最新版本：`v1.0`
- 更新时间：`20210703`

简介

介绍主流自动化工具Appium。包括对Appium的概述，简单介绍Appium的自动化操作iOS；详细介绍自动化操作Android，包括如何初始化搭建环境、如何用Appium Inspector调试页面元素，以及总结一些常见问题和经验心得。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/popular_automation_framework_appium: 主流跨平台自动化框架：Appium

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 主流跨平台自动化框架：Appium book.crifan.com
- 主流跨平台自动化框架：Appium crifan.github.io

离线下载阅读

- 主流跨平台自动化框架：Appium PDF
- 主流跨平台自动化框架：Appium ePub
- 主流跨平台自动化框架：Appium Mobi

版权说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

Appium概览

移动端测试框架中，流行度最广的是： `Appium`

- `Appium`
  - 概述
    - Appium是最主流的自动化测试框架，支持自动化操作 `iOS` 手机、 `Android` 手机和 `Windows` 系统中的 `原生` 、 `移动Web` 和 `混合` 的应用。
      - `原生应用` ：用 `iOS` 、 `Android` 或 `Windows SDKs` 编写的应用
      - `移动Web应用` ：用移动端浏览器访问的应用
        - `iOS` 上的 `Safari` 、 `Chrome`
        - `Android` 上的内置浏览器
      - `混合应用` ：带有一个 `Webview` 的包装器，用来和Web内容交互的原生控件
  - 主页
    - GitHub
      - appium/appium: Automation for iOS, Android, and Windows Apps.
    - 官网
      - Appium: Mobile App Automation Made Awesome.

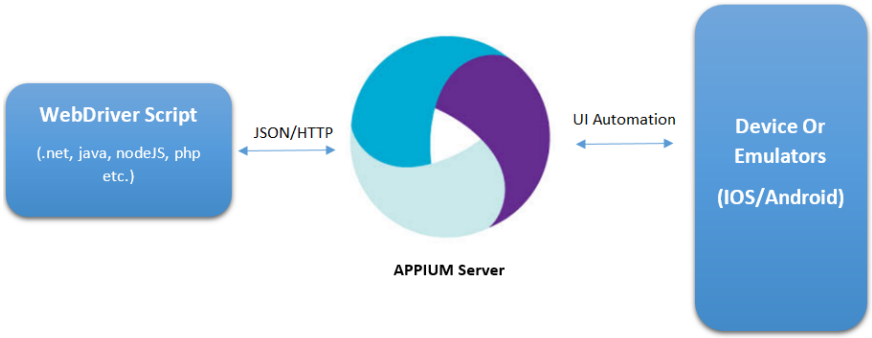
Appium依赖于底层的系统框架

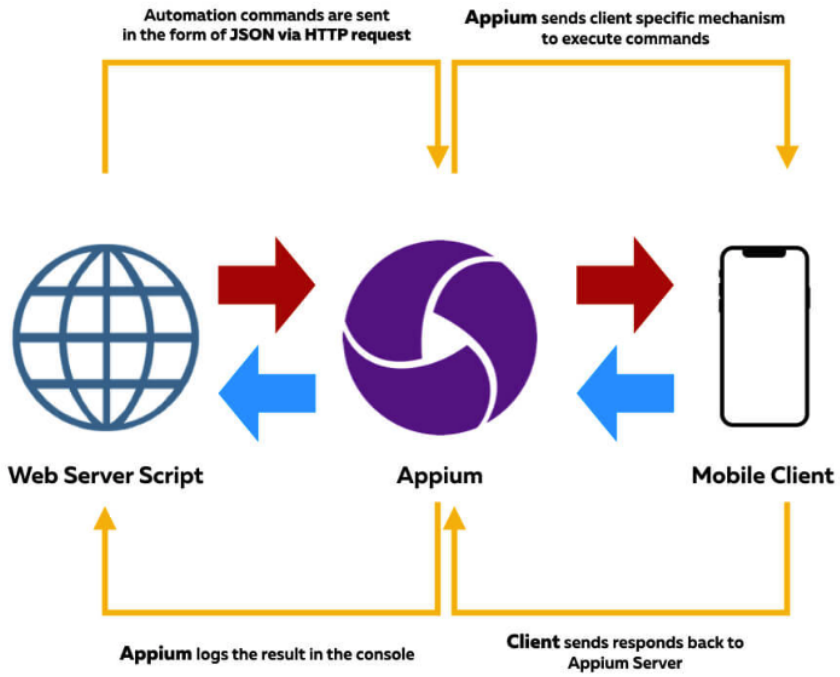
- iOS
  - Appium在 `iOS >9.3` 后全面采用 `WebDriverAgent` 的方案
- `Android >= 4.3` : Google's UiAutomator / UiAutomator2
- `Windows` : Microsoft's WinAppDriver

自动化操作iOS

此处介绍如何用Appium去自动化操作iOS设备。

Appium的iOS自动化架构





用Appium实现iOS自动化

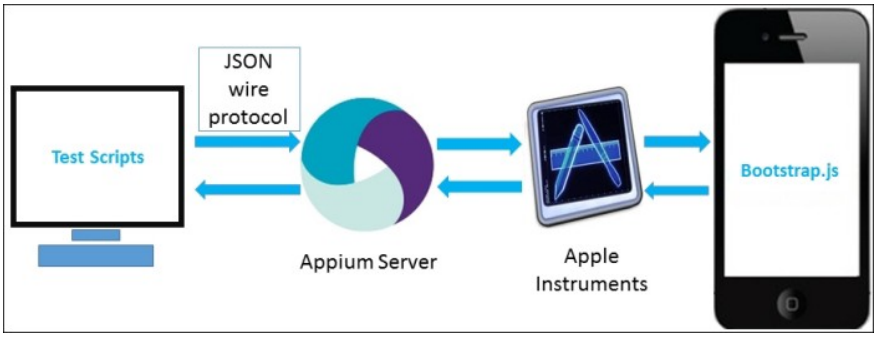
对于iOS的真机，需要预先配置好才可以：

关于Appium的真机的设置：

- 官网 英文
  - XCUITest Real Devices (iOS) - Appium
- readthedocs 中文

    - Real devices ios - appium
- GitHub
    - appium-xcuitest-driver/real-device-config.md at master · appium/appium-xcuitest-driver

iOS真机配置



自动化操作Android

此处整理如何用Appium去自动化操作Android设备。

搭建环境

Mac中搭建Appium自动化操作安卓的环境

Mac中安装Appium Desktop并启动Appium服务

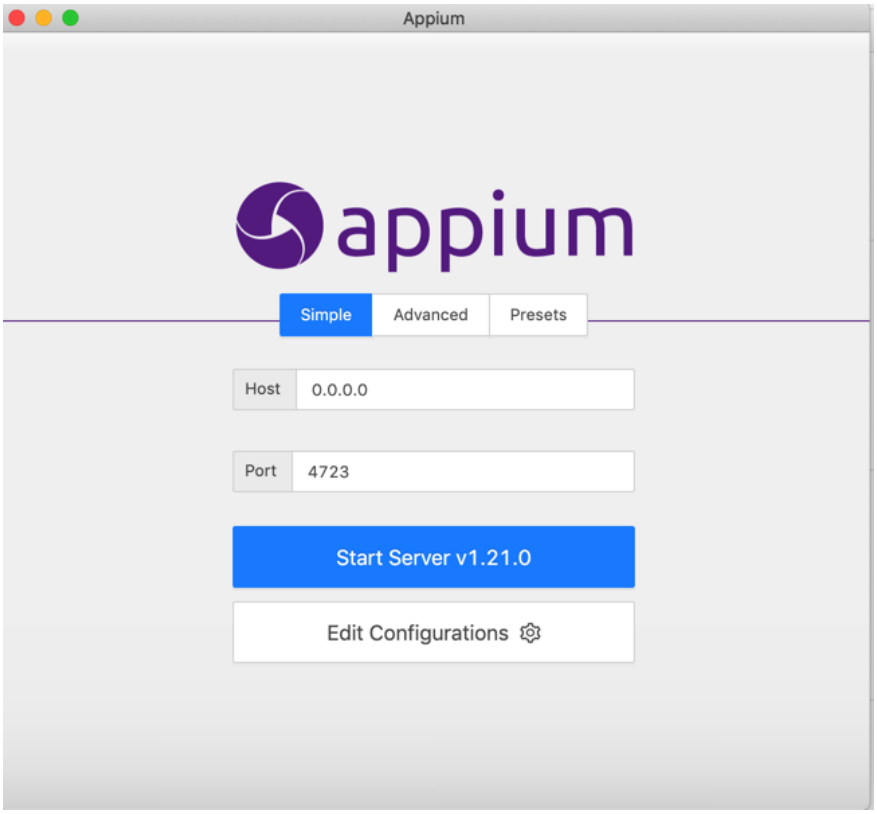
去Appium的官网，找到的Github的Release页面

下载最新版Appium的Mac客户端。

此处是：`Appium-1.21.0-mac.zip`

下载，解压，得到：`Appium.app`，拖动到`应用程序`，即可安装。

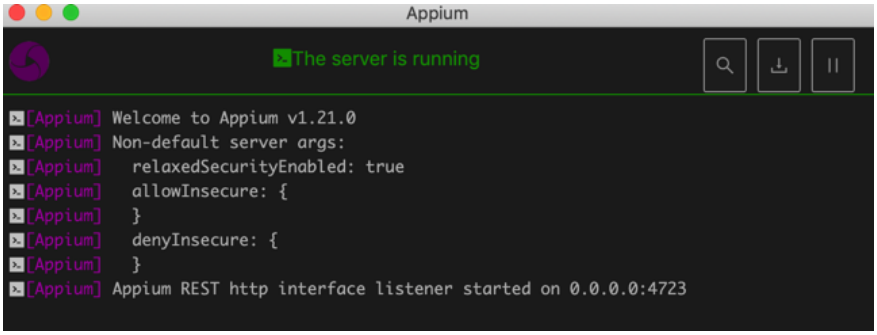
双击`Appium`打开：



- 默认参数
  - `Host`：`0.0.0.0`
  - `Port`：`4723`

点击`Start Server`，以启动Appium的服务（供client端调用接口）

看到log显示：




```
Appium REST http interface listener started on 0.0.0.0:4723
```

说明Appium的server端，启动成功

Mac中安装Appium的Python库

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

- 如果用 `pipenv` ，则是

  ```
  pipenv install Appium-Python-Client
  ```

Mac中：测试代码

```
from appium import webdriver
server = 'http://localhost:4723/wd/hub'

Android_Redmi10X_DeviceName = "crifan Redmi10X"
Android_Redmi10X_UDID = "orga4pmzee4ts47t"

desired_caps = {
    "platformName": "Android",
    "deviceName": Android_Redmi10X_DeviceName,
    "udid": Android_Redmi10X_UDID,
    "appPackage": "com.tencent.mm",
    "appActivity": ".ui.LauncherUI",
    "noReset": True
}

desired_caps['chromeOptions'] = {'androidProcess': 'com.ter
desired_caps['noReset'] = True

driver = webdriver.Remote(server, desired_caps)
```

- Capability的参数
  - 最核心的三个

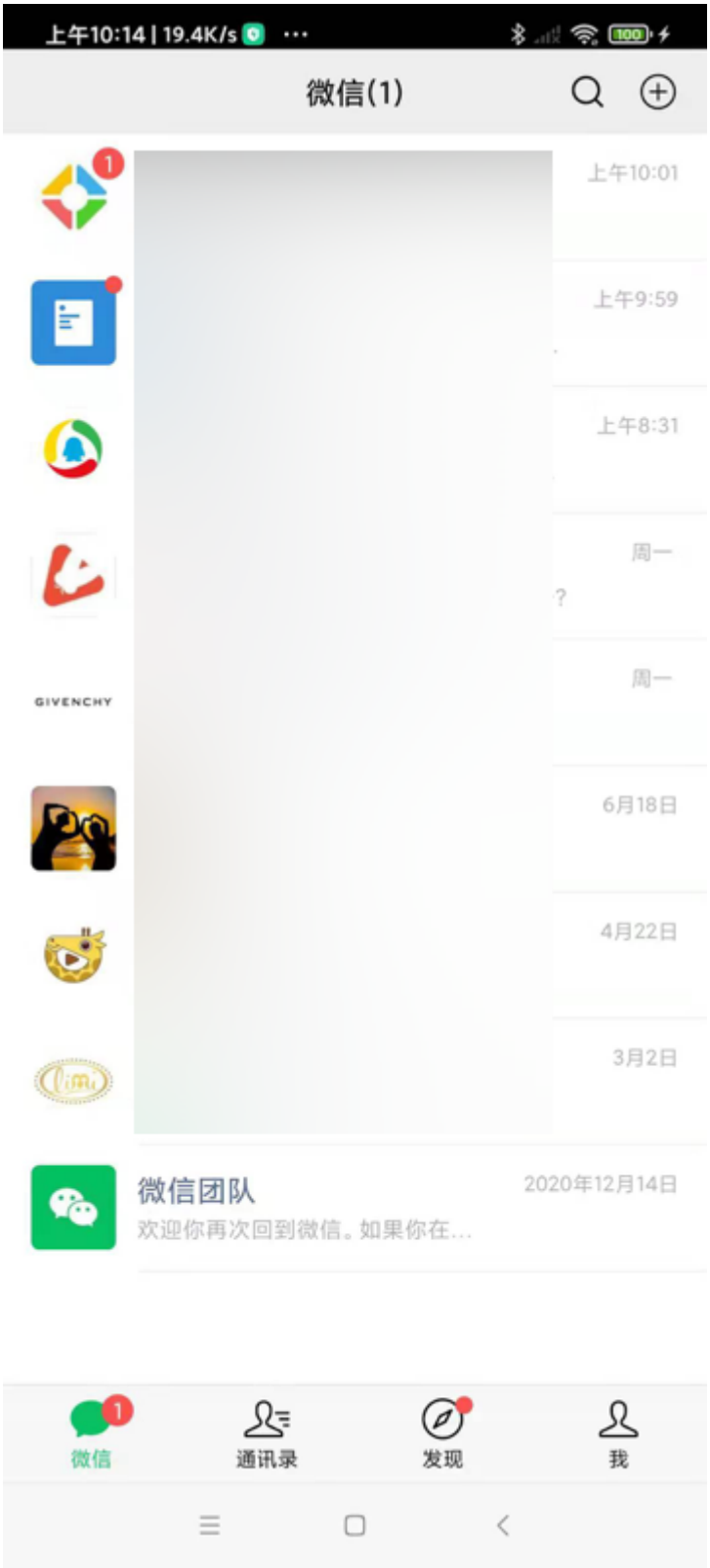
        - `platformName`：`Android`
            - 表示安卓平台
        - `udid`：安卓设备的序列号，设备唯一编号
            - 可以通过`adb devices`找到
                - 比如：

```
# adb devices
List of devices attached
orga4pmzee4ts47t    device
```

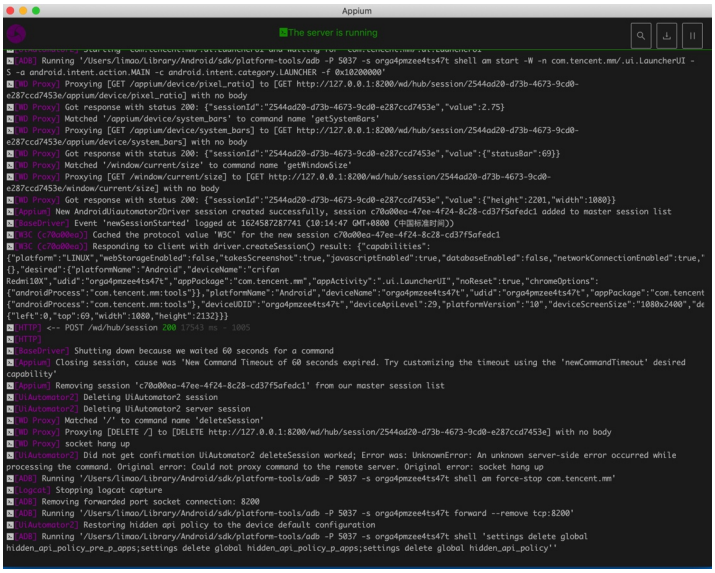
                - 中的`orga4pmzee4ts47t`
        - `deviceName`：随便填个值（当然最好是见名知意用户看得懂的值）
            - 比如
                - `"deviceName": "crifan Redmi10X",`
    - 详见
        - Desired Capabilities - Appium

效果：

- 正常打开了微信app



- Appium Desktop客户端中输出对应的log
  - 图


Appium
The server is running
[ADB] Running '/Users/limao/Library/Android/sdk/platform-tools/adb -P 5037 -s orga4pmzee4ts47t shell am start -W -n com.tencent.mm/.ui.LauncherUI -S -a android.intent.action.MAIN -c android.intent.category.LAUNCHER -f 0x10200000'
[WD Proxy] Proxying [GET /appium/device/pixel_ratio] to [GET http://127.0.0.1:8200/wd/hub/session/2544ad20-d73b-4673-9cd0-e287ccd7453e/appium/device/pixel_ratio] with no body
[WD Proxy] Got response with status 200: {"sessionId":"2544ad20-d73b-4673-9cd0-e287ccd7453e","value":2.75}
[WD Proxy] Matched '/appium/device/system_bars' to command name 'getSystemBars'
[WD Proxy] Proxying [GET /appium/device/system_bars] to [GET http://127.0.0.1:8200/wd/hub/session/2544ad20-d73b-4673-9cd0-e287ccd7453e/appium/device/system_bars] with no body
[WD Proxy] Got response with status 200: {"sessionId":"2544ad20-d73b-4673-9cd0-e287ccd7453e","value":{"statusBar":69}}
[WD Proxy] Matched '/window/current/size' to command name 'getWindowSize'
[WD Proxy] Proxying [GET /window/current/size] to [GET http://127.0.0.1:8200/wd/hub/session/2544ad20-d73b-4673-9cd0-e287ccd7453e/window/current/size] with no body
[WD Proxy] Got response with status 200: {"sessionId":"2544ad20-d73b-4673-9cd0-e287ccd7453e","value":{"height":2201,"width":1080}}
[Appium] New AndroidUiautomator2Driver session created successfully, session c70a00ea-47ee-4f24-8c28-cd37f5afedc1 added to master session list
[BaseDriver] Event 'newSessionStarted' logged at 1624587287741 (10:14:47 GMT+0800 (中国标准时间))
[MJSC (c70a00ea)] Cached the protocol value 'W3C' for the new session c70a00ea-47ee-4f24-8c28-cd37f5afedc1
[MJSC (c70a00ea)] Responding to client with driver.createSession() result: {"capabilities":{"platform":"LINUX","webStorageEnabled":false,"takesScreenshot":true,"javascriptEnabled":true,"databaseEnabled":false,"networkConnectionEnabled":true,"},"desired":{"platformName":"Android","deviceName":"crifan Redmi10X","udid":"orga4pmzee4ts47t","appPackage":"com.tencent.mm","appActivity":".ui.LauncherUI","noReset":true,"chromeOptions":{"androidProcess":"com.tencent.mm:tools"}},"platformName":"Android","deviceName":"orga4pmzee4ts47t","udid":"orga4pmzee4ts47t","appPackage":"com.tencent{"androidProcess":"com.tencent.mm:tools"},"deviceUDID":"orga4pmzee4ts47t","deviceApiLevel":29,"platformVersion":"10","deviceScreenSize":"1080x2400","de{"left":0,"top":69,"width":1080,"height":2132}}}
[HTTP] <-- POST /wd/hub/session 200 17513 ms - 1005
[HTTP]
[BaseDriver] Shutting down because we waited 60 seconds for a command
[Appium] Closing session, cause was 'New Command Timeout of 60 seconds expired. Try customizing the timeout using the 'newCommandTimeout' desired capability'
[Appium] Removing session 'c70a00ea-47ee-4f24-8c28-cd37f5afedc1' from our master session list
[UiAutomator2] Deleting UiAutomator2 session
[UiAutomator2] Deleting UiAutomator2 server session
[WD Proxy] Matched '/' to command name 'deleteSession'
[WD Proxy] Proxying [DELETE /] to [DELETE http://127.0.0.1:8200/wd/hub/session/2544ad20-d73b-4673-9cd0-e287ccd7453e] with no body
[WD Proxy] socket hang up
[UiAutomator2] Did not get confirmation UiAutomator2 deleteSession worked; Error was: UnknownError: An unknown server-side error occurred while processing the command. Original error: Could not proxy command to the remote server. Original error: socket hang up
[ADB] Running '/Users/limao/Library/Android/sdk/platform-tools/adb -P 5037 -s orga4pmzee4ts47t shell am force-stop com.tencent.mm'
[logcat] Stopping logcat capture
[ADB] Removing forwarded port socket connection: 8200
[ADB] Running '/Users/limao/Library/Android/sdk/platform-tools/adb -P 5037 -s orga4pmzee4ts47t forward --remove tcp:8200'
[UiAutomator2] Restoring hidden api policy to the device default configuration
[ADB] Running '/Users/limao/Library/Android/sdk/platform-tools/adb -P 5037 -s orga4pmzee4ts47t shell 'settings delete global hidden_api_policy_pre_p_apps;settings delete global hidden_api_policy_p_apps;settings delete global hidden_api_policy''

- （拷贝出的）文字

```
[debug] [35m[WD Proxy][39m Proxying [POST /session]
[debug] [35m[WD Proxy][39m Got response with status
[info] [35m[WD Proxy][39m Determined the downstream
[debug] [35m[WD Proxy][39m Proxying [GET /appium/de
[debug] [35m[WD Proxy][39m Got response with status
[debug] [35m[ADB][39m Running '/Users/limao/Library
[info] [35m[AndroidDriver][39m Screen already unloc
[info] [35m[UiAutomator2][39m Starting 'com.tencent
[debug] [35m[ADB][39m Running '/Users/limao/Library
[debug] [35m[WD Proxy][39m Got response with status
[debug] [35m[WD Proxy][39m Matched '/appium/device/
[debug] [35m[WD Proxy][39m Proxying [GET /appium/de
[debug] [35m[WD Proxy][39m Got response with status
[debug] [35m[WD Proxy][39m Matched '/window/current
[debug] [35m[WD Proxy][39m Proxying [GET /window/cu
[debug] [35m[WD Proxy][39m Got response with status
[info] [35m[Appium][39m New AndroidUiautomator2Driv
[debug] [35m[BaseDriver][39m Event 'newSessionStart
[debug] [35m[W3C (c70a00ea)][39m Cached the protoco
[debug] [35m[W3C (c70a00ea)][39m Responding to clie
[info] [35m[HTTP][39m [37m<-- POST /wd/hub/session
[info] [35m[HTTP][39m [90m[39m[warn] [35m[BaseDrive
[warn] [35m[Appium][39m Closing session, cause was
[info] [35m[Appium][39m Removing session 'c70a00ea-
[debug] [35m[UiAutomator2][39m Deleting UiAutomator
[debug] [35m[UiAutomator2][39m Deleting UiAutomator
[debug] [35m[WD Proxy][39m Matched '/' to command n
[debug] [35m[WD Proxy][39m Proxying [DELETE /] to [
[info] [35m[WD Proxy][39m socket hang up
[warn] [35m[UiAutomator2][39m Did not get confirmat
[debug] [35m[ADB][39m Running '/Users/limao/Library
[debug] [35m[Logcat][39m Stopping logcat capture
[debug] [35m[ADB][39m Removing forwarded port socke
[debug] [35m[ADB][39m Running '/Users/limao/Library
[info] [35m[UiAutomator2][39m Restoring hidden api
[debug] [35m[ADB][39m Running '/Users/limao/Library
```

安卓手机：点击允许安装app

第一次运行Appium的代码，会触发Appium去给安卓手机安装必要的app：

- `Appium Settings`

- `ios.appium.uiautomator2.server`

- `io.appium.uiautomator2.server.test`

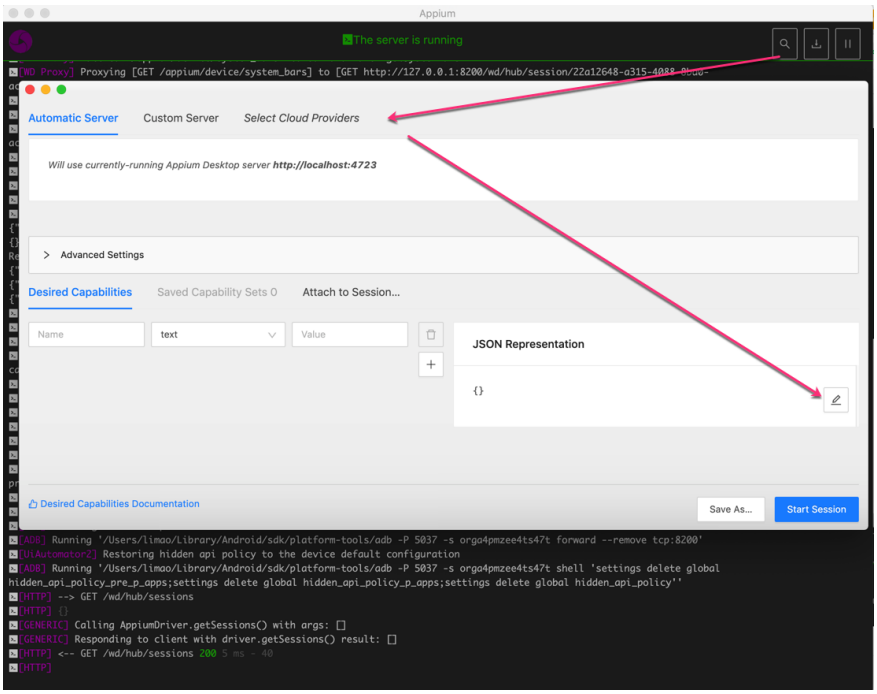
注意：及时点击 `继续安装` ，允许安装。

调试界面

调试安卓界面

Mac中用Appium的界面调试工具去调试安卓手机端的界面

在Appium的桌面客户端中，点击 放大镜🔍，启动参数配置界面：

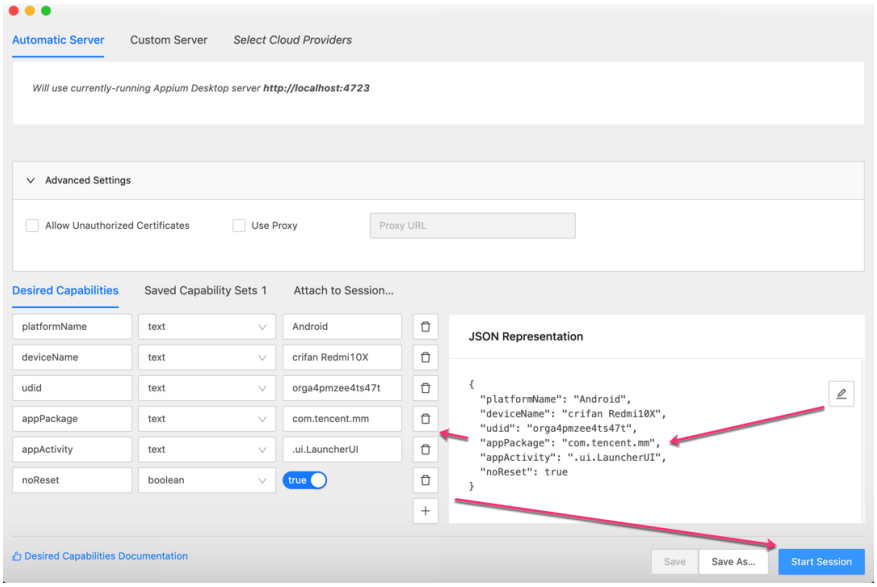


点击 `JSON Representation` 的编辑按钮，输入 `Capability` 的 `json` 配置

比如此处是：

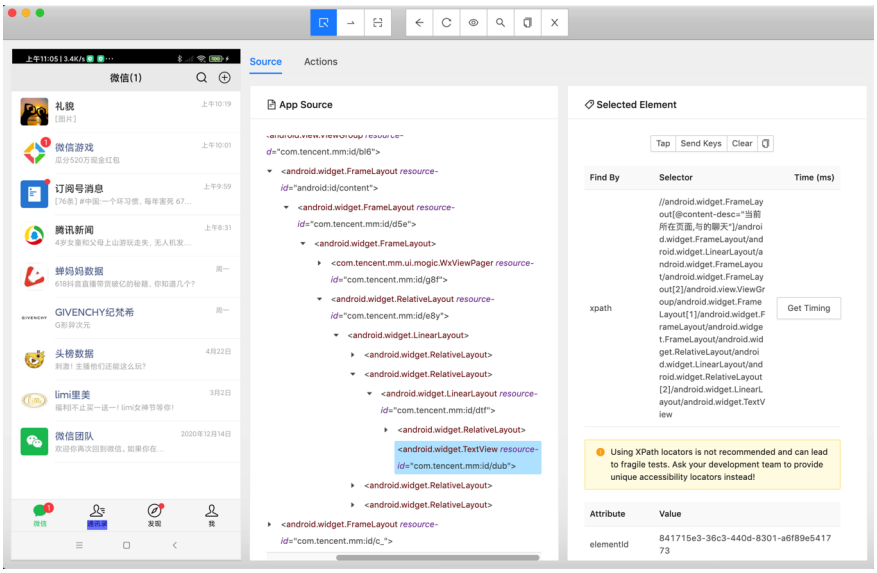
```
{
    "platformName": "Android",
    "deviceName": "crifan Redmi10X",
    "udid": "orga4pmzee4ts47t",
    "appPackage": "com.tencent.mm",
    "appActivity": ".ui.LauncherUI",
    "noReset": true
}
```

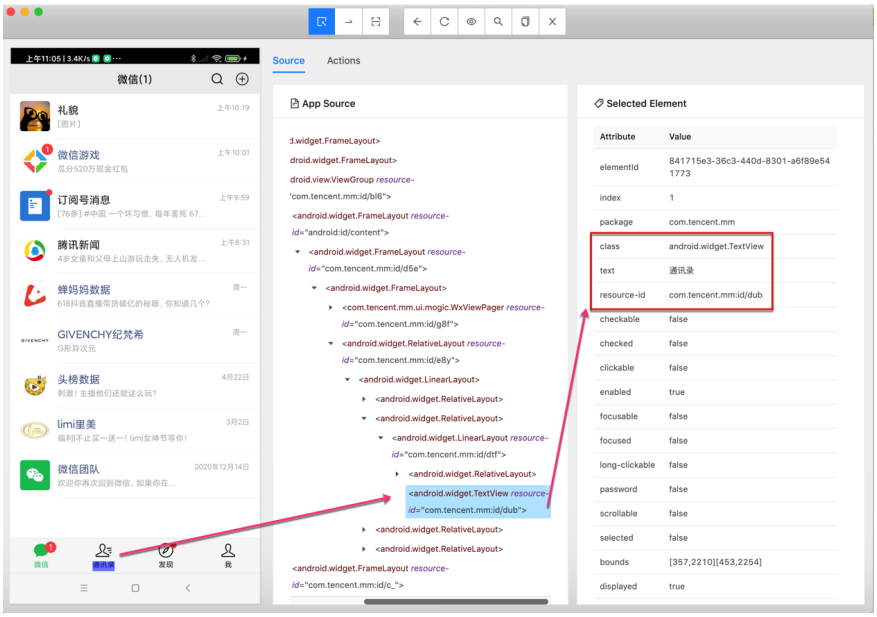
点击Save，自动保存和解析出相关参数：



点击 `Start Session` ，（稍等片刻）即可启动调试界面

然后点击手机页面中的元素，右边即可看到属性：





也可以拷贝出属性值：

```
Attribute     Value
elementId 841715e3-36c3-440d-8301-a6f89e541773
index 1
package com.tencent.mm
class android.widget.TextView
text 通讯录
resource-id com.tencent.mm:id/dub
checkable false
checked false
clickable false
enabled true
focusable false
focused false
long-clickable false
password false
scrollable false
selected false
bounds [357,2210][453,2254]
displayed true
```

其中，可以用于定位的，一般是用：

- `class`：`android.widget.TextView`
- `text`：`通讯录`
- `resource-id`：`com.tencent.mm:id/dub`

写成代码，可以是：

```
driver.find_element_by_xpath("//android.widget.TextView[@re
```

也可以是：

```
driver.find_element_by_xpath("//android.widget.TextView[@te
```

即可用上述定位元素的代码，去调试你的业务逻辑了。

比如此处可以接着点击 `通讯录` ：

```
contactElement = driver.find_element_by_xpath("//android.wi
contactElement.click()
```

进入通讯录的tab页面了：

心得：

- appium 和 Appium的GUI界面调试工具：没有uiautomator2的weditor好用
  - appium（不论是Python的代码，还是 界面调试工具）连接了安卓手机后（比如打开了微信app），会导致用户无法正常手动操作

(微信)

- 而weditor，就不会影响用户手动操作

结论：

- appium不好用。不建议继续用appium
  - 建议换用：uiautomator2 + weditor

常见问题和心得

此处总结Appium开发期间常遇到的一些问题和一些使用方面的心得和经验：

Original error: Could not find a connected Android device

此处appium启动报错

```
Original error: Could not find a connected Android device
```

意思是，连不上，找不到设备。

解决办法：

- 硬件上
  - 确保安卓设备正常通过USB数据线连接到了Mac中
    - `adb devices`
      - 可以查看到设备（的udid值）
- 软件上
  - 参数配置，最核心的是三个
    - `platformName`：Android
      - 安卓平台
    - `udid`：安卓设备的序列号，设备唯一编号
      - 可以通过 `adb devices` 找到
    - deviceName：随便填个值（当然最好是见名知意用户看得懂的值）
      - 比如
        - `"deviceName": "crifan Redmi10X"`

具体参数解释，详见官网

Desired Capabilities - Appium

或：

appium/caps.md at master · appium/appium (github.com)

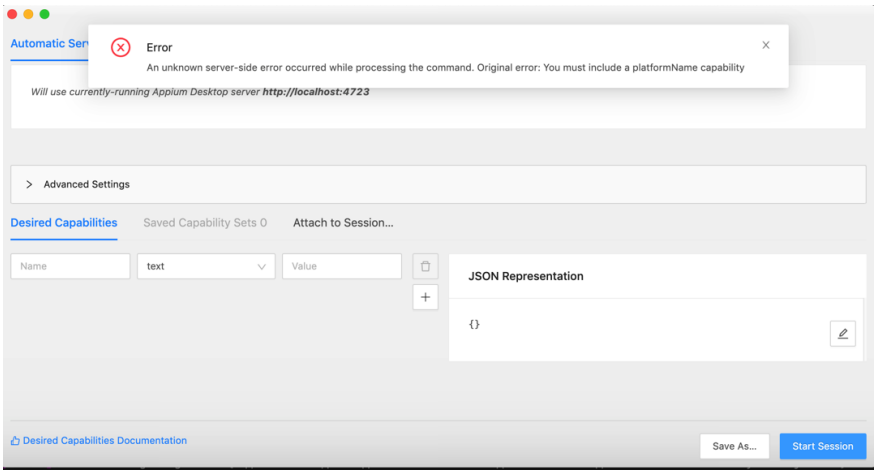
Original error You must include a platformName capability

背景：已启动了appium的server端.

已用Python代码，加上Capability参数，启动了client端，且成功打开了当前的app：微信

然后希望用appium的GUI图形界面 `Appium Desktop` 中点击 `放大镜🔎` 后，出现的调试界面，其中点击 `Start Session` ，但是报错：

```
An unknown server-side error occurred while processing the
Original error: You must include a platformName capability
```

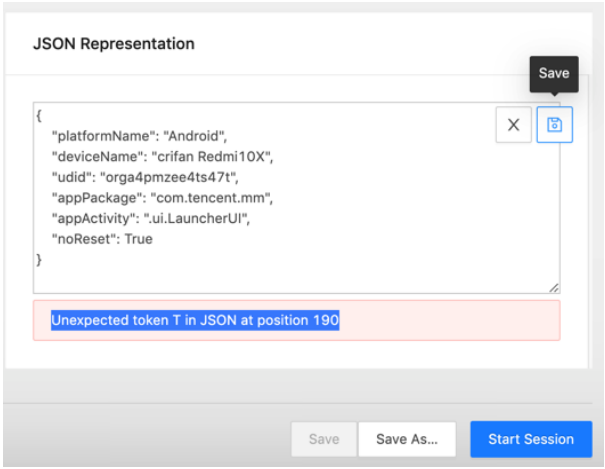


原因：没有填入 `platformName` 参数

解决办法：填入参数即可

但要额外说明的是：

- 此处appium的GUI调试界面，要填写的Capability参数，和Python代码中Capability参数，没关系
  - 就不会，像我本以为的，appium的server内部，自动填入python代码传递的参数了
- GUI中填写Capability参数时，可以直接一次性写好JSON字符串，放到 JSON Representation中，点击Save，即可自动解析到左边参数列表中
  - 注意
    - 如果填写的json有错误，会有错误提示的
      - 注：GUI窗口要足够大，确保能看到JSON Representation底部的JSON错误提示



        - 我开始就是，窗口太小，没看到底部错误提示，而搞不清为何还是报错
    - 说明
        - `Save` 后，记得 `Save as` 一下，可以直接加载，避免每次充分输入参数。
            - 正常 `Save` 后，第二个Tab：`Saved Capability Set` 后面会有个数提示，比如 `1` ， `2` 。点击 `Saved Capability Set` ，可以看到参数详情。

附上，此处的参数：

```
{
    "platformName": "Android",
    "deviceName": "crifan Redmi10X",
    "udid": "orga4pmzee4ts47t",
    "appPackage": "com.tencent.mm",
    "appActivity": ".ui.LauncherUI",
    "noReset": true
}
```

其中：

- json中，不支持Python中的**True**，要写成小写的**true**

Appium Inspector调试连接已存在的session会话

经过后续确认，Appium也是支持调试连接已有session的。

背景是：此处已用Python连接了Appium的Server，已经正在操控安卓手机了，比如点击进去了新的页面。

此时，希望搞清楚新页面中元素细节，以便于写代码定位元素，实现点击元素等后续操作。

希望此处可以用Appium去连接和调试，且同时不要冲突了当前的Python连接。

后续得知，此种需求就叫做：Appium的调试工具Appium Inspector，连接已存在的会话

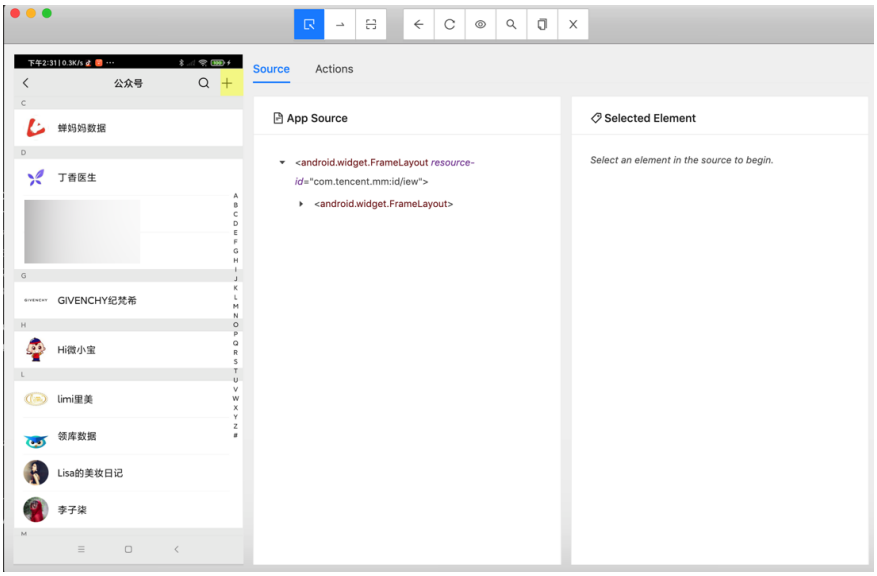
操作方法：切换到 `Attach to Sesssion`

![appium_inspector_attach_session]

```
If you have an already-running session of the above server
Select the Session ID in the dropdown below.
```

然后Appium Inspector会自动检测出当前已有的session，点击切换到你要连接的session，再点击 `Attach to Sesssion`

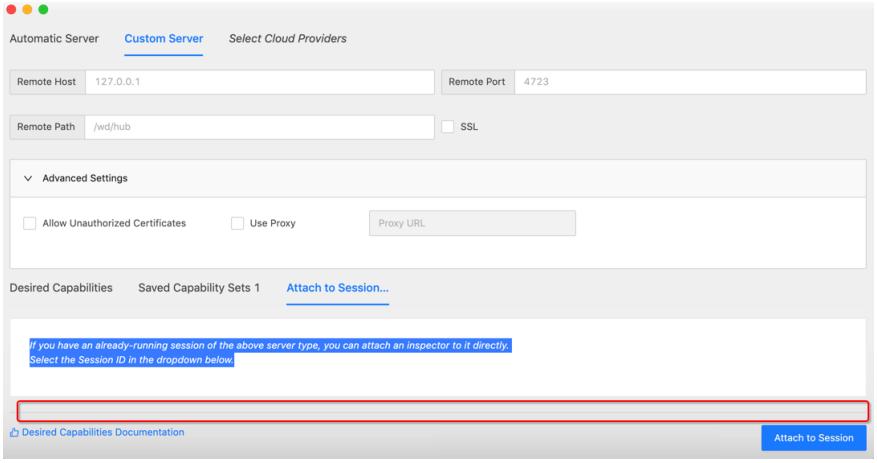
即可，不影响现有手机中的画面，可以开始调试找元素属性了：



注意：

确保当前Inspector的窗口高度足够高，不要出现我之前遇到的：

Inspector窗口高度太低，比如：



就会导致，上面的 已检测出的session的列表，看不到了。

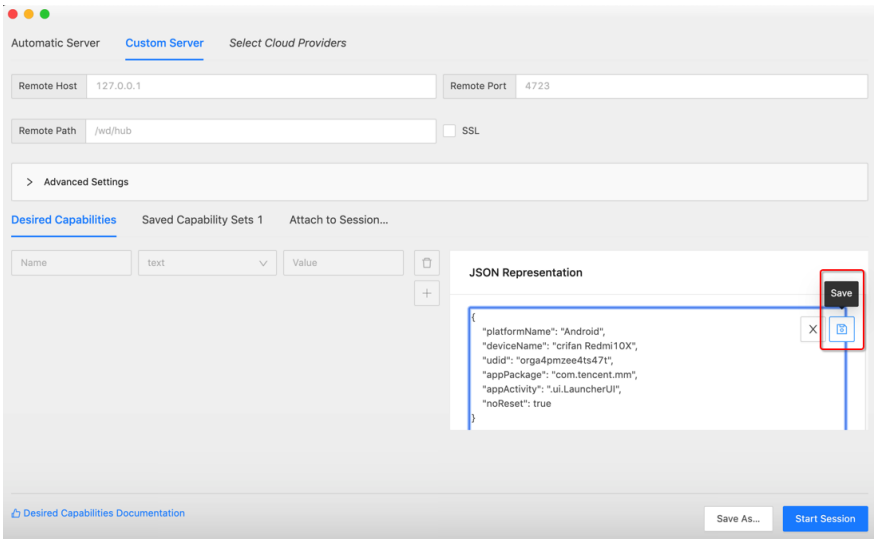
从而误以为，Appium Inspector不支持连接已有session的功能呢。

Appium保存参数配置

Appium Inspector支持，保存参数，避免每次都要很麻烦的输入参数才能连接设备。

步骤是：

Appium Inspector，在输入和Save了Capability的参数后，点击 `Save` ：



后，再切换到 `Saved Capability Set` ，且有个后缀数字，此处是 `1` ，表示已保存了1个配置：

appium_saved_capability_set_1

如此，即可保存配置，以后每次无需再次输入参数，即可，点击选中当前要用的配置，再点击 `Start Session` 即可连接设备。

附录

下面列出相关参考资料。

参考资料

- 【已解决】Mac中安装appium的Python库
- 【已解决】Mac中初始化搭建appium的Python的运行环境
- 【已解决】Mac中下载安装和启动appium的server服务
- 【已解决】Mac中如何找到Android手机的appium的参数deviceName
- 【已解决】Appium报错：WebDriverException Original error Could not find a connected Android device
- 【已解决】Appium启动调试界面报错：Original error You must include a platformName capability
- 【已解决】Mac中用appium的GUI图形界面调试页面元素
- 【已解决】Appium界面调试工具Desktop Inspector如何连接已存在的session会话
-
- 简介 - Appium
- Introduction - Appium
- Supported Platforms - Appium
- XCUITest Real Devices (iOS) - Appium
- Real devices ios - appium
- appium-xcuitest-driver/real-device-config.md at master · appium/appium-xcuitest-driver
- Desired Capabilities - Appium
-