目录

iOS自动化测试利器：facebook-wda

- 最新版本：`v2.0`
- 更新时间：`20210717`

简介

总结iOS自动化主流框架facebook-wda的简介，环境搭建以及常见问题，以及开发的心得，包括用weditor调试界面和元素，尤其是iOS的各种坑，以及常用代码段，包括crifan优化后的wda、元素处理、iOS设备操作、app的各个方面，比如app管理、首次登录引导页、微信的安全模式和微信公众号、设置中的更新WiFi代理、AppStore的自动安装iOS的app、弹框处理、调试辅助代码等，并给出部分源码分析，最后附上文档和参考资料。

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/ios_automation_facebook_wda: iOS自动化测试利器：facebook-wda

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

- iOS自动化测试利器：facebook-wda book.crifan.com
- iOS自动化测试利器：facebook-wda crifan.github.io

离线下载阅读

- iOS自动化测试利器：facebook-wda PDF
- iOS自动化测试利器：facebook-wda ePub
- iOS自动化测试利器：facebook-wda Mobi

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

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

简介

之前已在：

移动端自动化测试概览

中提到过，iOS自动化测试的主流框架之一是：`facebook-wda`

此处去详细介绍一下。

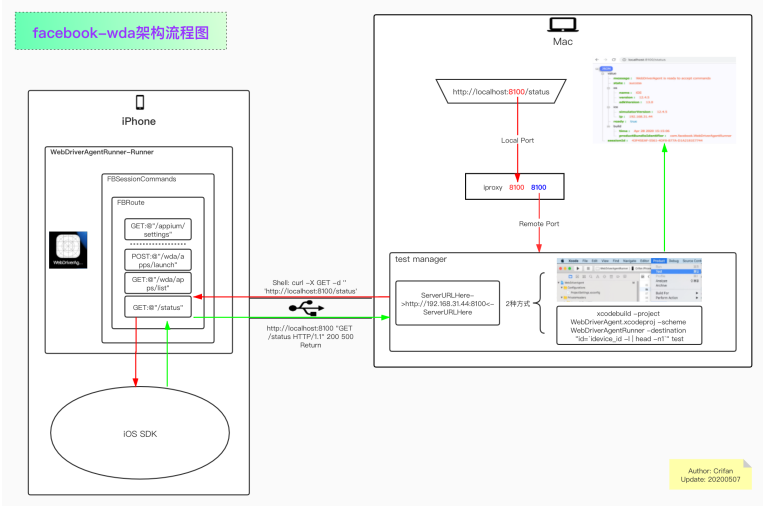
- facebook-wda
  - 主页
    - openatx/facebook-wda: Facebook WebDriverAgent Python Client Library (not official)
  - 作者：`openatx`
  - 语言：`Python`
  - 实现原理
    - 基于 `Appium` 的 `WebDriverAgent`
      - 关于 `WebDriverAgent`
        - 简称：`WDA`
        - 是什么=一句话描述：一个基于 `W3C` 的 `WebDriver` 的 `server`（的具体实现）
        - 底层依赖：`Apple` 的XCUITest（测试框架）
        - 起源和状态
          - 最早：`Facebook` 开发的
            - Facebook的WebDriverAgent
              - 现已**暂停维护**= `archived` = `read-only`
              - 主页
                - facebookarchive/WebDriverAgent: A WebDriver server for iOS that runs inside the Simulator
          - 现在：已停止维护
            - `Appium` 接手继续维护和更新
              - Appium的WebDriverAgent
                - 主页
                  - appium/WebDriverAgent: A WebDriver server for iOS that runs inside the Simulator
      - 关于 `WebDriver`
        - 作者：`W3C`
        - 是什么：一套协议规范

        - 特点：与平台协议无关
        - 目的=作用：远程控制设备
      - 主页
        - https://w3c.github.io/webdriver/
    - 关于：`Apple 的 XCUITest`
      - 是什么：苹果的测试框架
      - 官网文档
        - 之前：
          - XCUITest
        - （貌似）最新
        - XCTest | Apple Developer Documentation
    - 关于：`Appium`
      - Appium: Mobile App Automation

搭建环境

此处介绍如何用 `facebook-wda` 搭建iOS设备的自动化测试环境。

先介绍 `facebook-wda` 的架构流程图：

- 本地图片
  - 

- 在线网页查看
  - facebook-wda架构流程图

开发环境概述

- 开发环境概述
  - `client` = 客户端
    - 你要测试的 `iOS` 设备，比如 `iPhone`
      - 给 `iPhone` 中安装 `WebDriverAgentRunner-Runner`
  - `server` = 服务端 = `test manager` = `WebDriverAgent`的服务
    - 需要在 `Mac` 中启动 `test manager`

首次：初始化

先介绍初始化需要做的事情，其中：

- 初始化 = 第一次 = 首次 = 只需要做一次，以后无需重复做

想要能自动化操作iPhone等iOS设备，需要

（1）先确保Mac环境OK

把相关后续要用到的工具都安装好：

```
brew update
brew uninstall --ignore-dependencies libimobiledevice
brew uninstall --ignore-dependencies usbmuxd
brew install --HEAD usbmuxd
brew unlink usbmuxd
brew link usbmuxd
brew install --HEAD libimobiledevice
```

再去安装wda的库：

```
pip install facebook-wda
```

（2）再去给 `iPhone` 中安装相关内容

先确保iPhone已正常连接：

把iOS设备（iPhone等）插入Mac后，用

```
idevice_id -l
```

确保可以找到iPhone设备。

再去给iPhone中安装：

- 客户端 = APP = `WebDriverAgentRunner-Runner`
  - 用于配合 `Mac` 中的 `server端` 的 `test manager`

安装后的效果：

此处长按变待删除，才能看到app全名是：`WebDriverAgentRunner-Runner`：

给iPhone中安装WebDriverAgentRunner-Runner

核心思路，都是编译和安装app `WebDriverAgentRunner-Runner` 到iPhone中：

- 确保Mac中已安装XCode

下载代码：

```
git clone https://github.com/appium/WebDriverAgent.git
```

切换目录：

```
cd WebDriverAgent
```

可以看到核心的入口文件，即Xcode项目文件：`WebDriverAgent.xcodeproj`

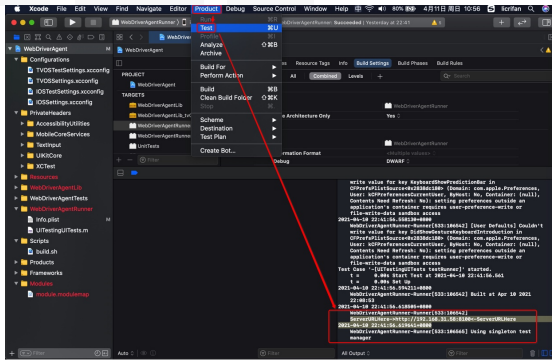
关于如何编译和安装，则有2种方式：

- 通过IDE `XCode` 去编译和安装
  - `Xcode -> Product -> Test`
    - 注：准备工作包括
      - 用 `XCode` 打开 `WebDriverAgent.xcodeproj`
      - 选择 `Target` 的APP是：`WebDriverAgentRunner`
      - 去选择 `Team`，是自己（或别的可用的）`开发者账号`
      - 会触发自动 `Code Signing`
      - 最后才是：`Product -> Test`
- 在 `终端` 运行 `xcodebuild` 命令去编译和安装
  - `Terminal` 中：运行 `xcodebuild` 的 `test`

上述操作步骤，和后续的每次运行 `test manager` 的方式是一样的，所以细节放在后面介绍。

之后：每次调试之前

启动test manager服务

- `server` = `服务端` = `test manager` = `WebDriverAgent的服务`
  - 需要在 `Mac` 中启动 `test manager`
    - 2种方式
      - XCode
        - `Xcode` -> `Product` -> `Test`
        - 
      - 终端
        - `Terminal` 中：运行 `xcodebuild` 的 `test`
          - 直接一步：

            ```
            xcodebuild -project WebDriverAgent.x
            ```

          - 或分2步
            - 先获取iOS设备的UDID：

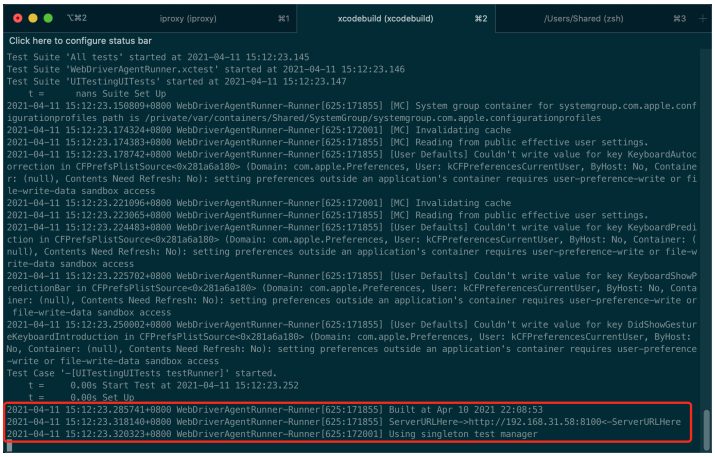
              ```
              CUR_UDID=$(idevice_id -l | head
              ```

            - 再运行

              ```
              xcodebuild -project WebDriverAge
              ```

          - 注：
            - 要在 `WebDriverAgent` 的目录中运行上述命令
            - `idevice_id -l` 作用是列出当前连接到Mac中的所有iOS的设备（的UDID）
              - 详见：idevice_id
            - `head -n1` 作用是获取第一个（iOS设备的UDID）

第一次：确保wda服务运行正常

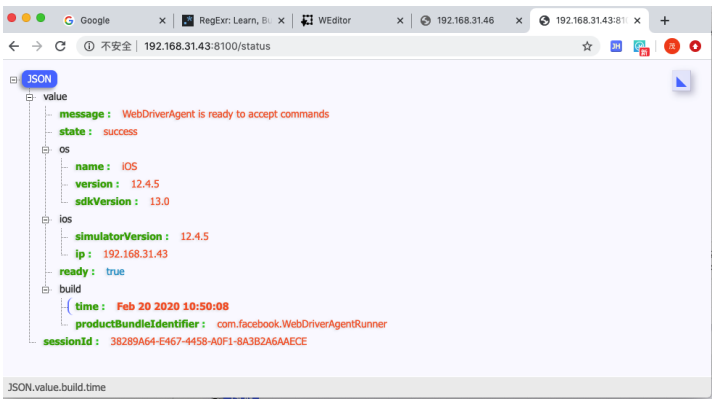
- 先确保输出正常的信息，包含：`ServerURLHere`和`Using singleton test manager`
  - 图
    - 

  - 文字

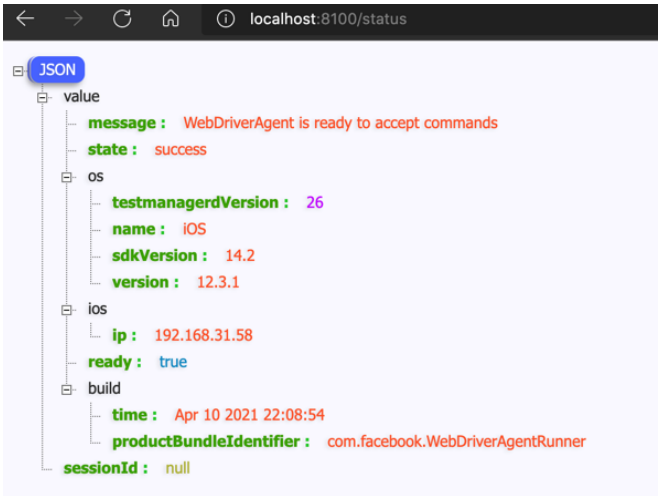
```
o o o
Test Case '-[UITestingUITests testRunner]' started.
    t =     0.01s Start Test at 2020-02-20 10:50:59
    t =     0.01s Set Up
2020-02-20 10:50:59.968359+0800 WebDriverAgentRunne
2020-02-20 10:51:00.119667+0800 WebDriverAgentRunne
2020-02-20 10:51:00.123946+0800 WebDriverAgentRunne
```

即表示正常启动了`test manager`=`WDA的server`了

如何确认`test manager`服务已正常运行

- 用浏览器等访问status端口
  - （用浏览器）打开`test manager`中显示的地址
    - http://192.168.31.43:8100
    - 再加上`status`后，就是
    - http://192.168.31.43:8100/status
  - 可以返回json状态信息
    - 


```
{
  "value": {
      "message": "WebDriverAgent is ready to ac
      "state": "success",
      "os": {
          "name": "iOS",
          "version": "12.4.5",
          "sdkVersion": "13.0"
      },
      "ios": {
          "simulatorVersion": "12.4.5",
          "ip": "192.168.31.43"
      },
      "ready": true,
      "build": {
          "time": "Feb 20 2020 10:50:08",
          "productBundleIdentifier": "com.faceb
      }
  },
  "sessionId": "38289A64-E467-4458-A0F1-8A3B2A6
}
```

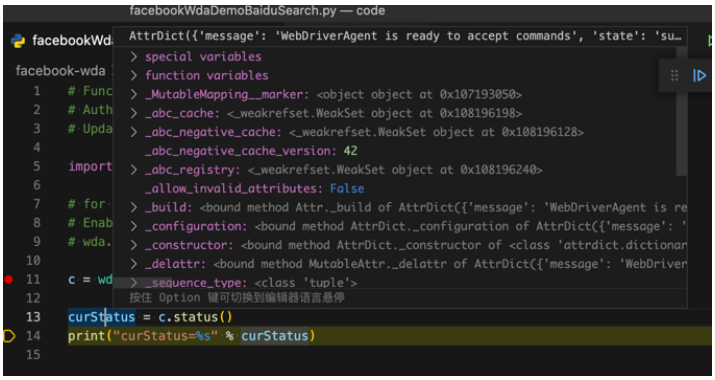

- 如果已用（ `iproxy 8100 8100` 实现了）端口转发，则可以直接用 `localhost`
  - http://localhost:8100/status
    - 

- 用代码测试

```
import wda

# for debug
# Enable debug will see http Request and Response
# wda.DEBUG = True

c = wda.Client('http://localhost:8100')

curStatus = c.status()
print("curStatus=%s" % curStatus)
```

- 确保能输出信息
  - 比如

    ```
    curStatus=AttrDict({'message': 'WebDriverAgent
    ```

  - 

说明服务启动正常，环境搭建正常了

搭建环境期间常见问题和心得

下面整理一些搭建环境期间的常见问题和心得总结：

当前被测iOS设备详情

在启动 `test manager` 期间会输出当前被测设备的详细信息

举例：

(1) `iOS 12.4.5` 的 `iPhone6`

```
2020-05-07 09:20:31.198 xcodebuild[2440:2434041] [MT] IDETe
        deviceSerialNumber:         DNPND9S1G5MR
        identifier:                 ed94089f3e34d5538065a69
        deviceClass:                iPhone
        deviceName:                 Crifan iPhone6
        deviceIdentifier:           ed94089f3e34d5538065a69
        productVersion:             12.4.5
        buildVersion:               16G161
        deviceSoftwareVersion:      12.4.5 (16G161)
        deviceArchitecture:         arm64
        deviceTotalCapacity:        60058931200
        deviceAvailableCapacity:    38391648256
        deviceIsTransient:          NO
        ignored:                    NO
        deviceIsBusy:               NO
        deviceIsPaired:             YES
        deviceIsActivated:          YES
        deviceActivationState:      Activated
        isPasscodeLocked:           NO
        deviceType:                 <DVTDeviceType:0x7f8456
        supportedDeviceFamilies:    (
    1
)
        applications:               (null)
        provisioningProfiles:       (null)
        hasInternalSupport:         NO
        hasWritableSystem:          NO
        isSupportedOS:              YES
        bootArgs:                   (null)
        nextBootArgs:               (null)
        connected:                  YES
        isWirelessEnabled:          NO
        connectionType:             direct
        hostname:                   (null)
        bonjourServiceName:         d4:f4:6f:0a:30:80@fe80::
        activeProxiedDevice:        (null)
        } (12.4.5 (16G161))
```

USB端口转发

为了测试更方便，最好安装和启动端口转发

具体方式是，用 `iproxy` 或 `mobiledevice` 实现，把访问Mac本地的端口，转发到USB连接着的iOS设备中

命令：

对于只连接单个iOS设备，比如某个iPhone的话，只需要：

```
iproxy 8100 8100
```

或：

```
mobiledevice tunnel 8100 8100
```

更多解释和用法，详见：

端口转发 · 苹果相关开发总结

如何确认 `test manager` 服务已正常运行

可以去访问运行了 `test manager` 最后所输出的地址：

`http://192.168.31.43:8100`

加上 `status` 后是：

`http://192.168.31.43:8100/status`

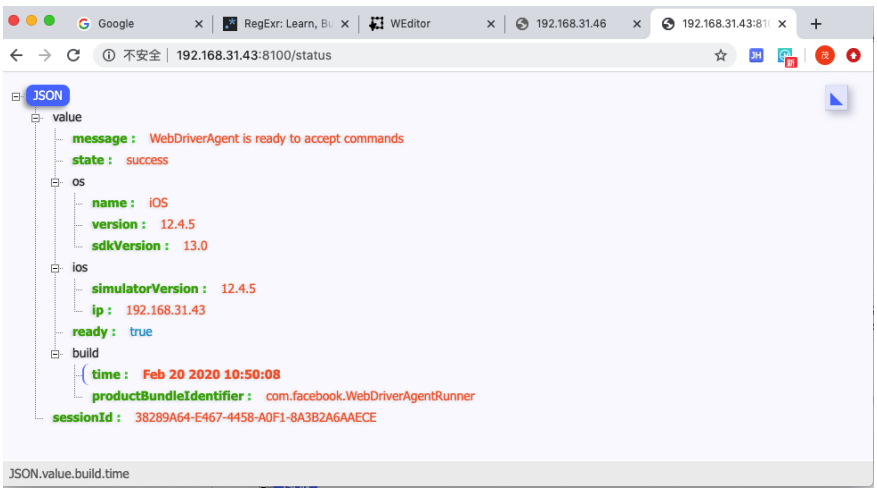
如果已端口转发则可以把IP换localhost

如果用了端口转发，则可以把IP换成localhost：

`http://localhost:8100/status`

会输出当前状态信息：

```
{
    "value": {
        "message": "WebDriverAgent is ready to accept comma
        "state": "success",
        "os": {
            "name": "iOS",
            "version": "12.4.5",
            "sdkVersion": "13.0"
        },
        "ios": {
            "simulatorVersion": "12.4.5",
            "ip": "192.168.31.43"
        },
        "ready": true,
        "build": {
            "time": "Feb 20 2020 10:50:08",
            "productBundleIdentifier": "com.facebook.WebDr
        }
    },
    "sessionId": "38289A64-E467-4458-A0F1-8A3B2A6AAECE"
}
```

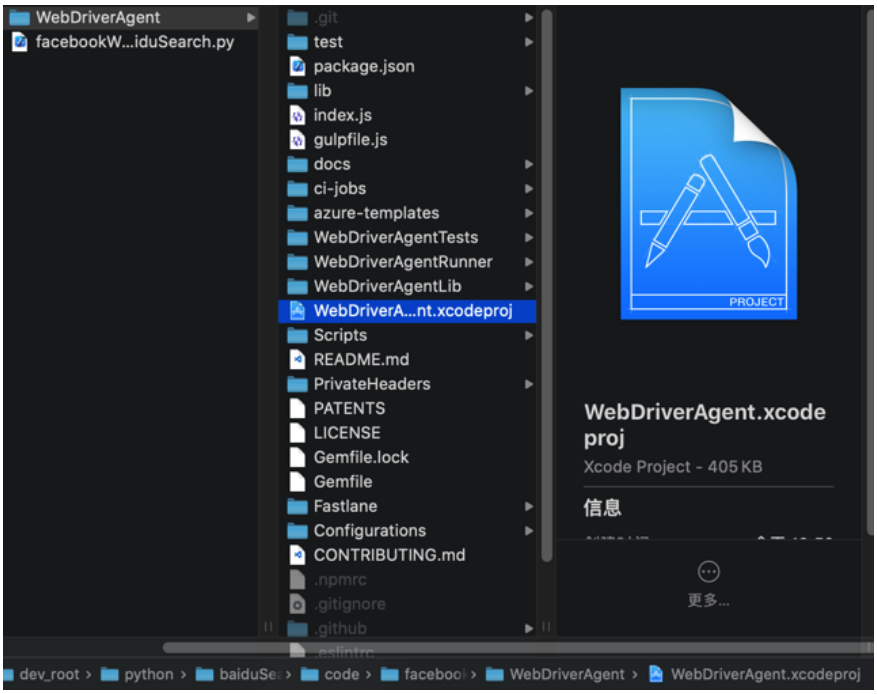
如何用XCode编译 WebDriverAgent.xcodeproj

对于下载到 `WebDriverAgent` 的源码中的 `WebDriverAgent.xcodeproj` ，第一次编译最好去用XCode编译。

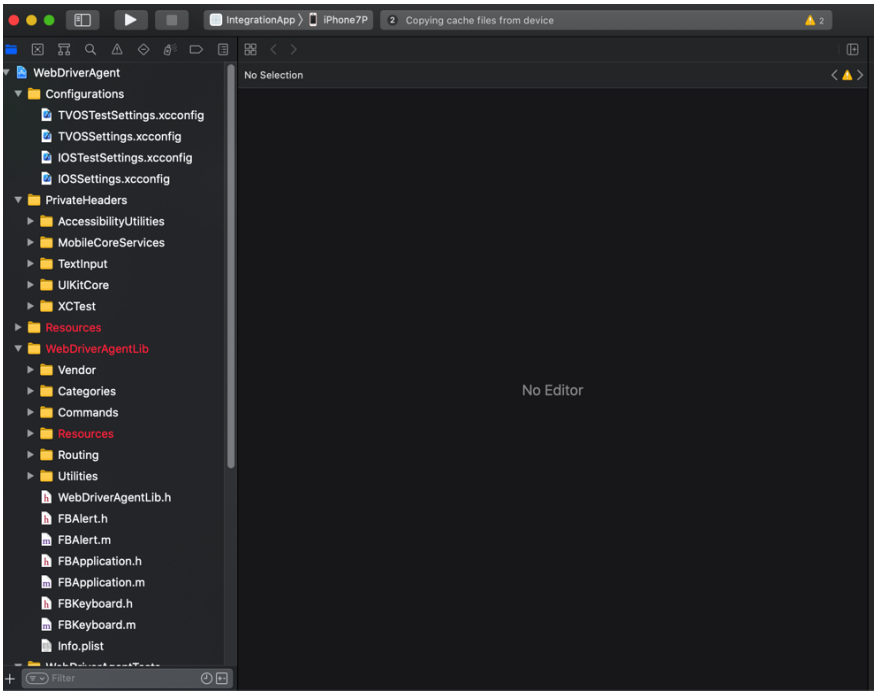
因为往往涉及到配置 `Team` 和 `自动签名` 等事宜。

下面就来介绍，如何用 `XCode` 去配置和编译 `WebDriverAgent.xcodeproj`

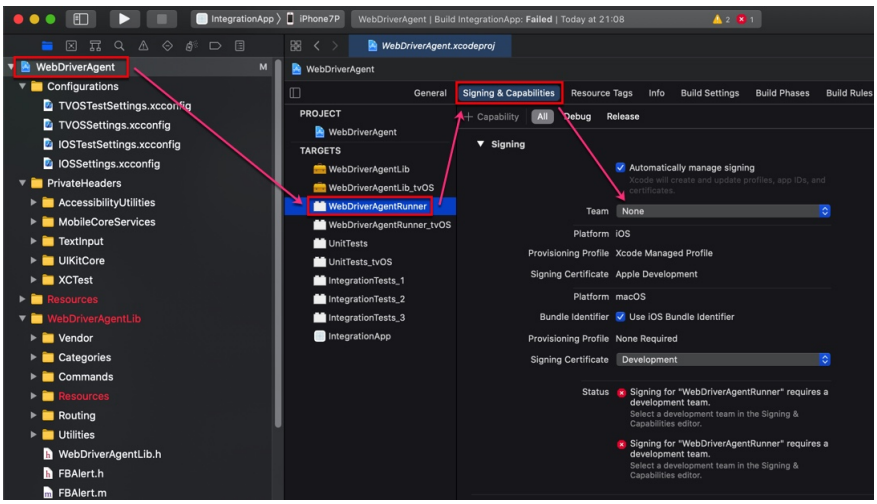
双击 `WebDriverAgent.xcodeproj`



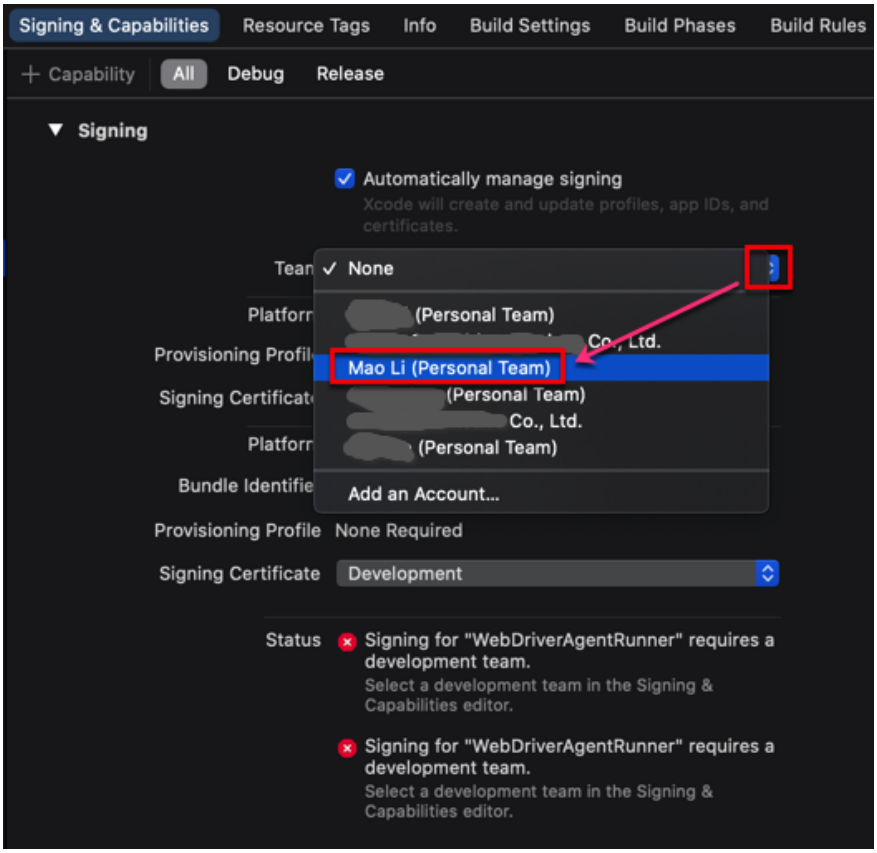
会自动用XCode打开：



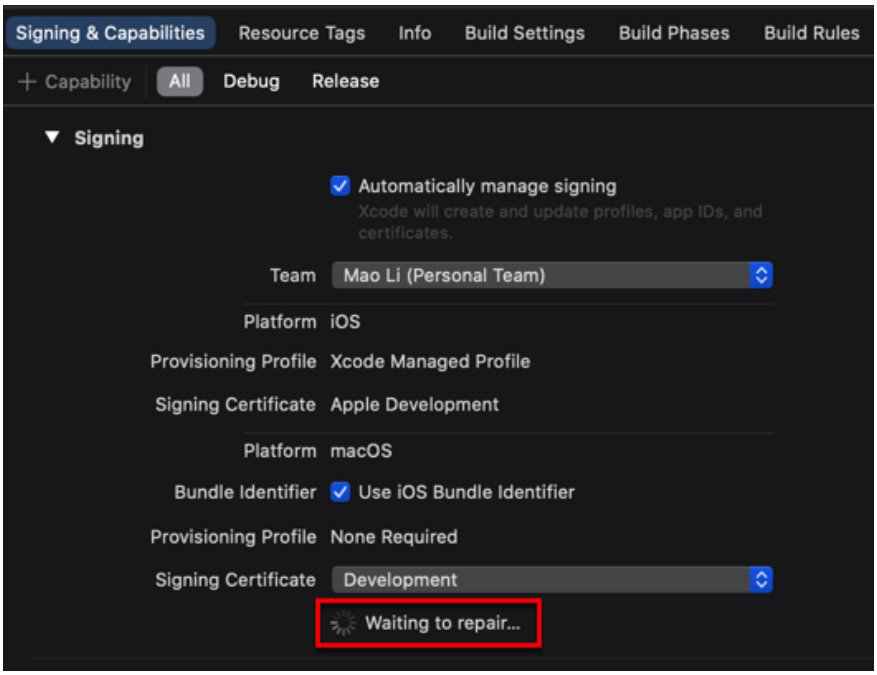
点击左上角的项目，进入项目属性，点击 `TARGETS` 中的 `WebDriverAgentRunner`，切换到 `Signing & Capabilities`：



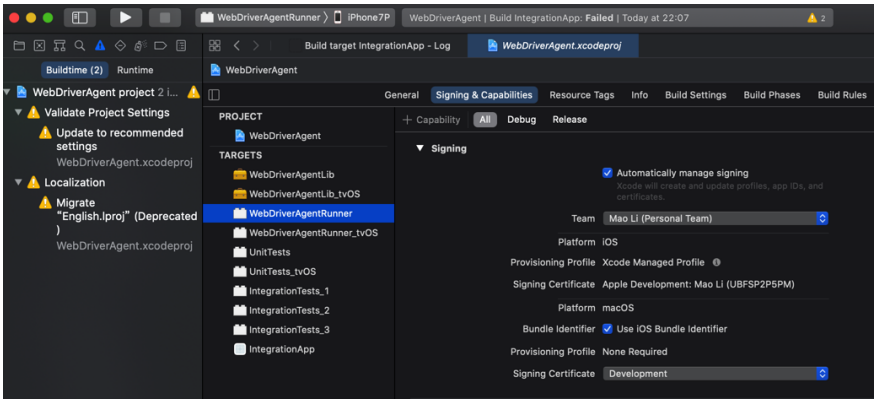
默认 `Team` 是 `None`，需要去选择一个自己的苹果账号：



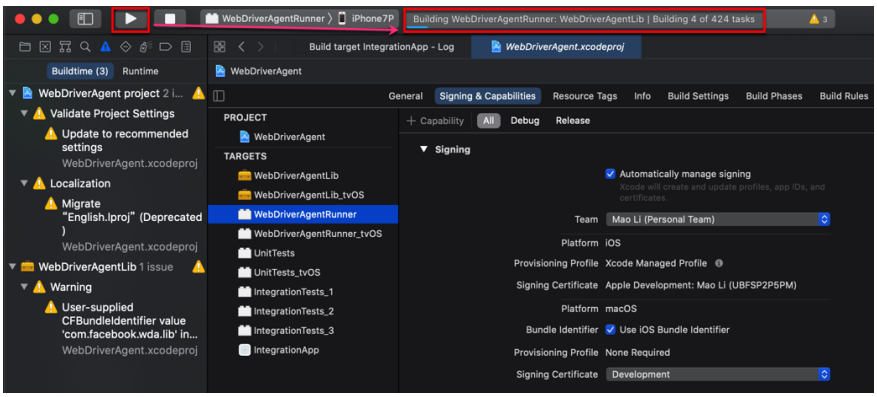
然后会触发自动修复，显示 `Waiting to repair` ：



看到没有其他警告或错误，就表示自动创建签名和Profile等工作正常了：



接着即可去编译了：点击左上角 `▶` 按钮，即可触发编译，显示 `Building ...`

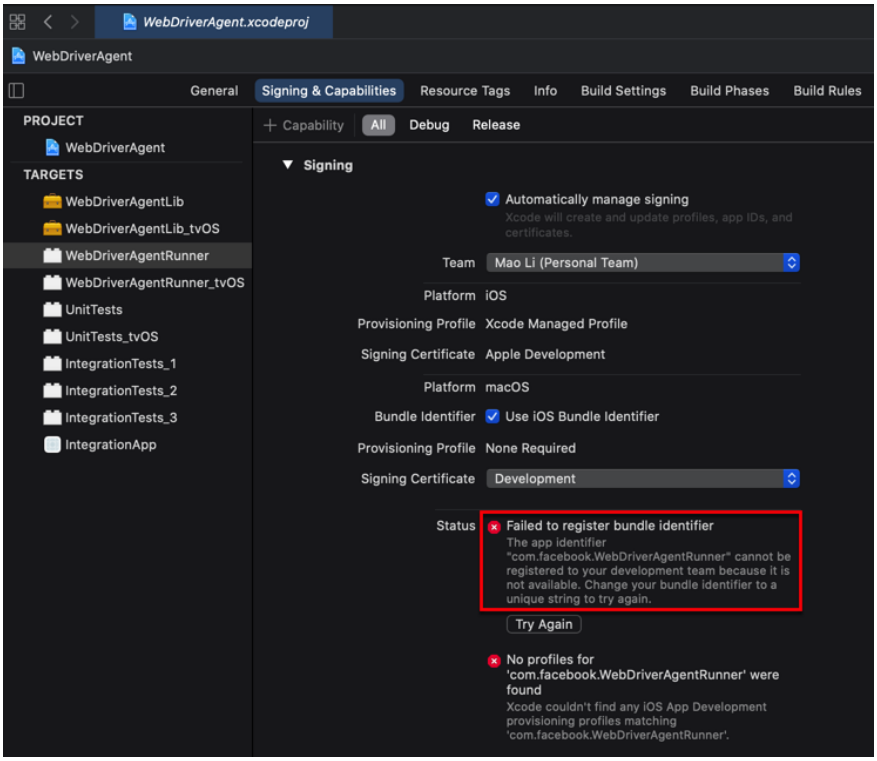


之后即可正常的 `Product -> Test` 去测试，启动服务，供后续使用了。

Failed to register bundle identifier

如果 `Signing & Capablities` 的自动修复后报错：

```
Failed to register bundle identifier
The app identifier "com.facebook.WebDriverAgentRunner" cann
```

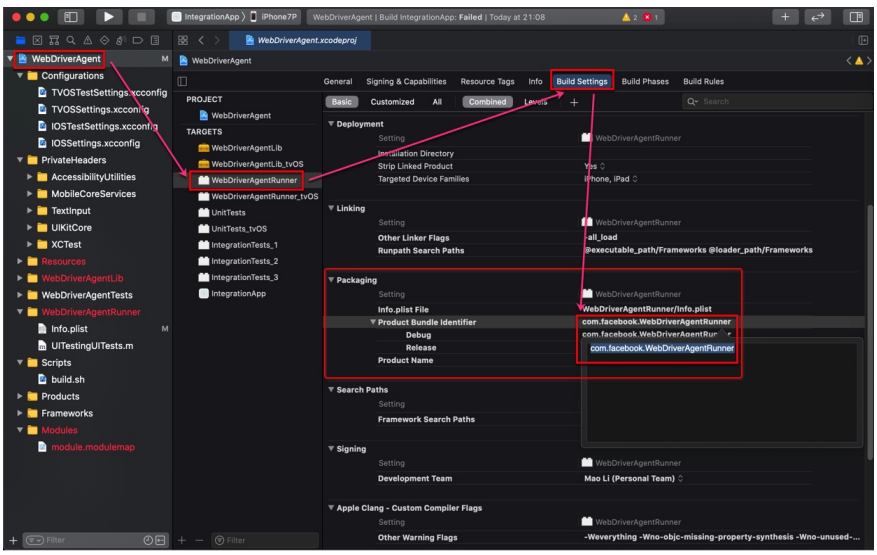


原因：（很可能是）默认的ID：`com.facebook.WebDriverAgentRunner`已存在，重复了，导致无法继续。

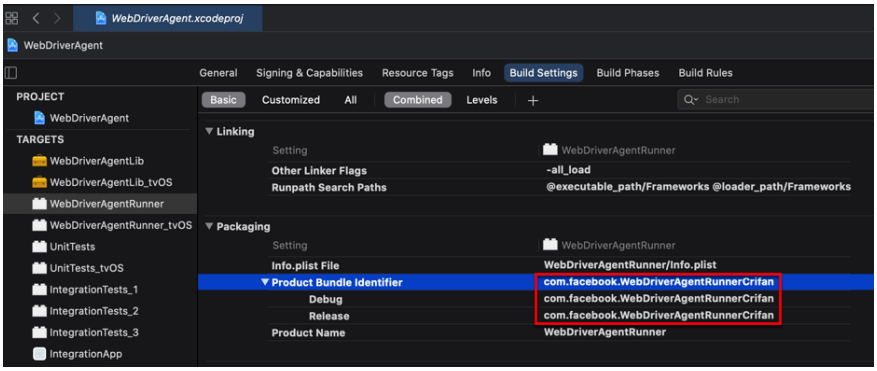
解决办法：修改为其他（独一无二的）值

操作步骤：

`WebDriverAgentRunner`的属性 -> `Build Settings` -> `Packaging` -> `Product Bundle Identifier`



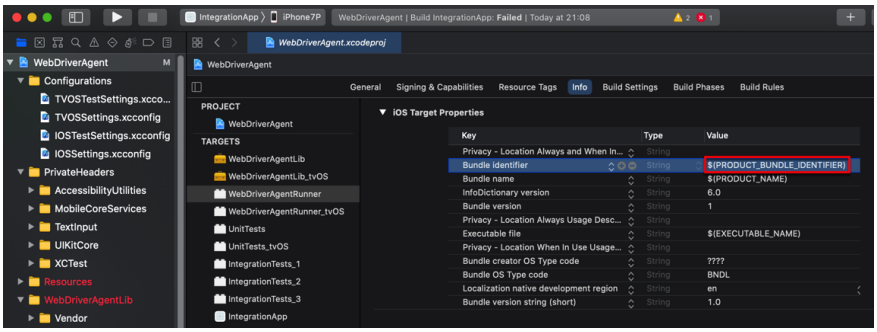
把值从默认的：`com.facebook.WebDriverAgentRunner`改为别的，确保不重复的值，比如我此处改为：`com.facebook.WebDriverAgentRunnerCrifan`



别处调用到此处的Product Bundle Identifier

后来注意到一个细节，别处会调用到此处的 `Product Bundle Identifier` 中的值

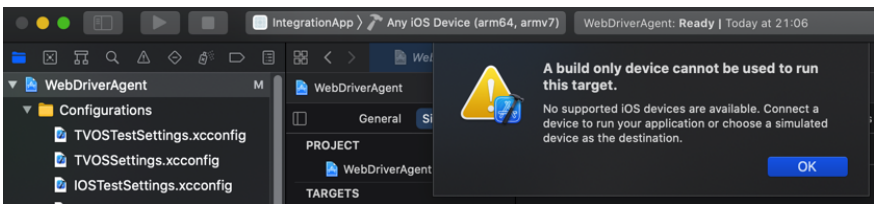
比如： `Info` -> `Key` -> `Bundle Identifier` : `$(PRODUCT_BUNDLE_IDENTIFIER)`



XCode报错：A build only device cannot be used to run this target

现象：编译期间报错

```
A build only device cannot be used to run this target.
No supported iOS devices are available. Connect a device to
```



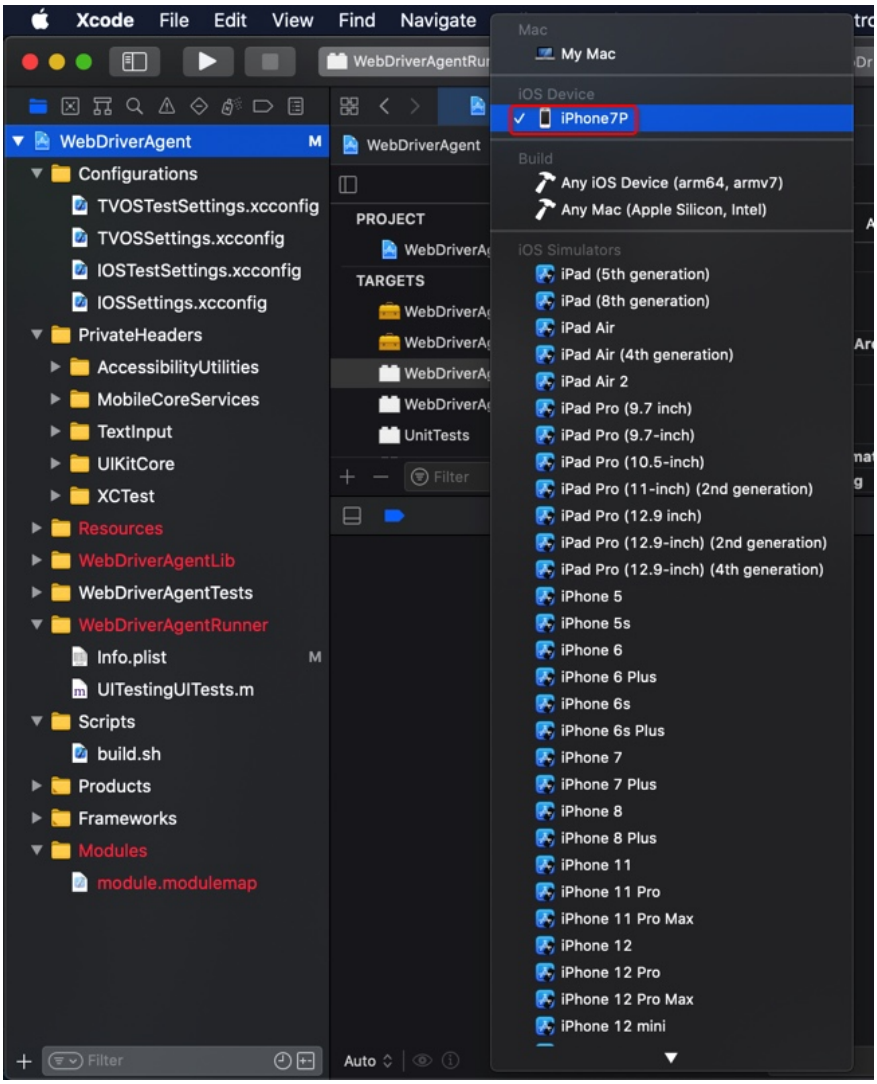
原因：XCode中没有选择正确的目标设备

解决办法：插入iPhone，且选择对应的iPhone等iOS真机设备。

具体步骤：把此处的 `iPhone7P` 插入 `Mac`


esc
tab
Q
W
A
Z
21:38
10
日历
时钟
天气
备忘录
提醒事项
App Store
音乐
信息
电话
计算器
文件
相机
地图

然后XCode中选择对应的目标设备，为 `iPhone7P`



注：可以借助于 `idevice_id` 去列出当前已连接的iOS设备的ID：

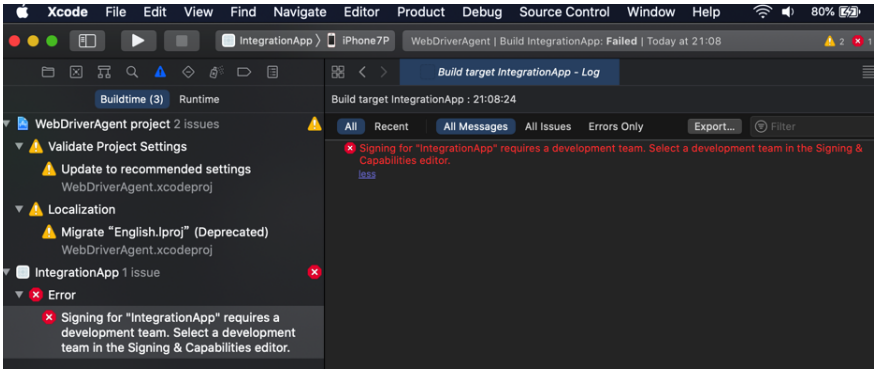
```
□ idevice_id -l
3dc13714e21415898e8e2c2863d96990a4d69c97
```

说明iOS设备的确已连接

XCode报错：Signing for requires a development team. Select a development team in the Signing & Capabilities editor

XCode去编译项目，报错：

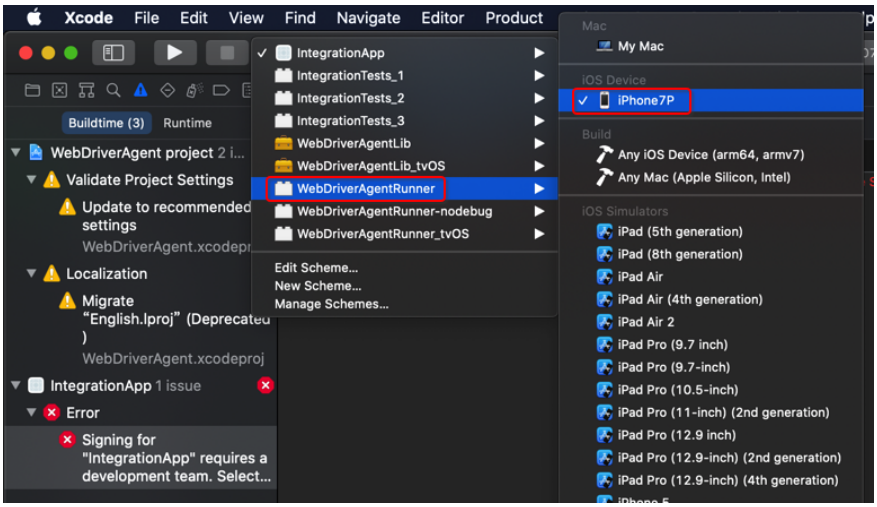
```
Signing for "IntegrationApp" requires a development team. S

Showing All Messages
```



出错原因：把本身要编译的app搞错了，不是这个 `IntegrationApp` ，应该是 `WebDriverAgentRunner`

解决办法：把要编译的app换成 `WebDriverAgentRunner`

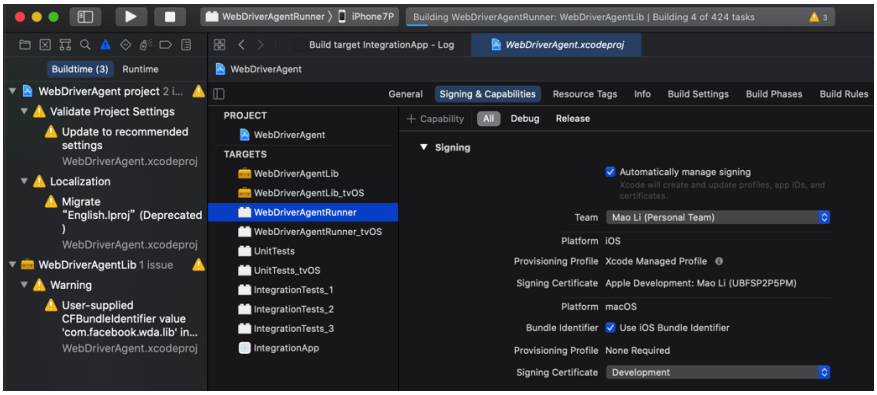
具体步骤：



即可。

另外，需要去给 `WebDriverAgentRunner` 加上 `code signing`

-》 此处是通过把 `Team` 从 `None` 改为 `自己的值` ，然后自动修复

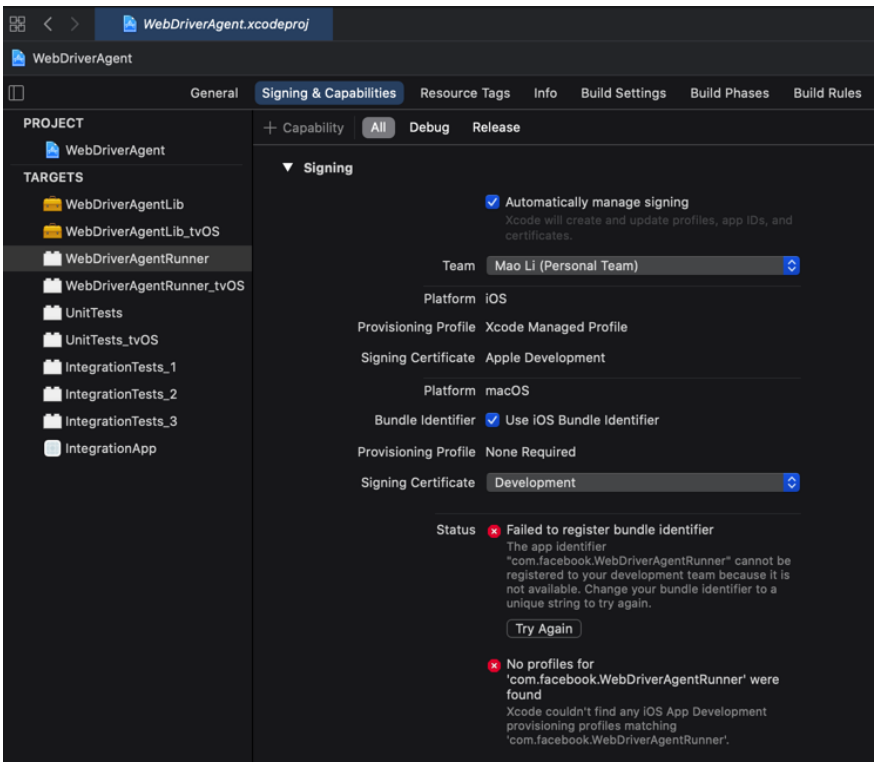


一般即可修复成功，最终加上code signing。

XCode报错：Failed to register bundle identifier com.facebook.WebDriverAgentRunner

XCode中尝试编译WebDriverAgentRunner，当选择了Team后，自动Singing，结果报错：

```
Failed to register bundle identifier
The app identifier "com.facebook.WebDriverAgentRunner" cann
```



原因：估计是id重复了

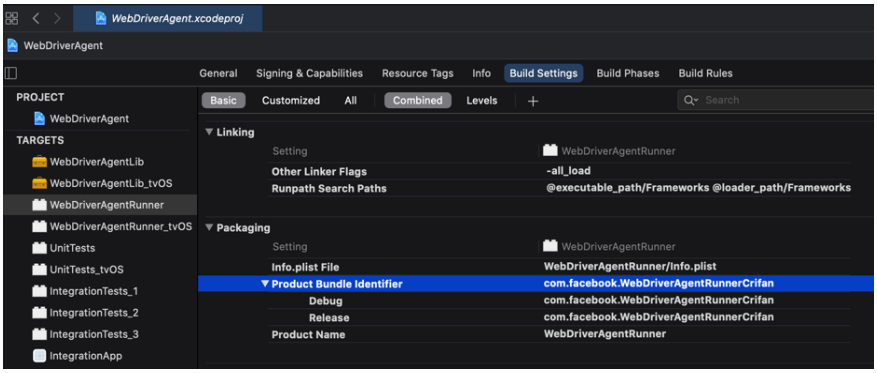
解决办法：去更换id

具体步骤：

把 `com.facebook.WebDriverAgentRunner` 为别的值，比如：`com.facebook.WebDriverAgentRunner_Crifan`

最终是，对于 `WebDriverAgentRunner 的 bundle Identifier` 值，最后是在：

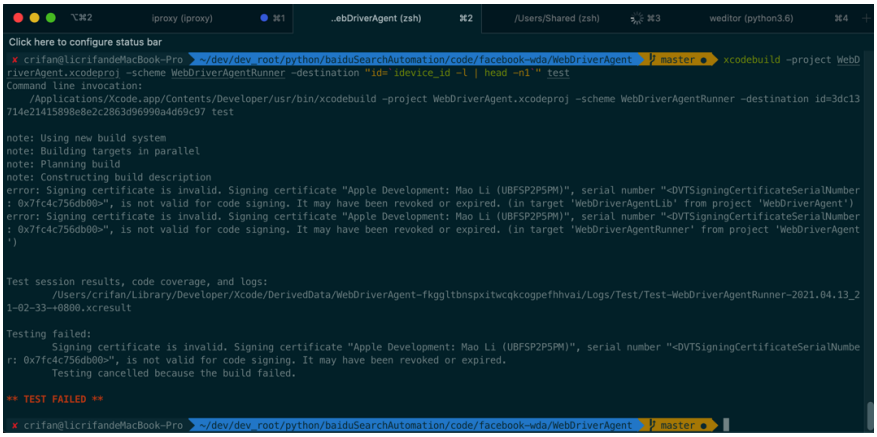
`WebDriverAgentRunner 的 属性 -》 Build Settings -》 Packaging -》 Product Bundle Identifier` 中去修改的



xcodebuild报错：Signing certificate is invalid

xcodebuild编译报错：

```
□ xcodebuild -project WebDriverAgent.xcodeproj -scheme Webl
Command line invocation:
    /Applications/Xcode.app/Contents/Developer/usr/bin/xcod

note: Using new build system
note: Building targets in parallel
note: Planning build
note: Constructing build description
error: Signing certificate is invalid. Signing certificate
error: Signing certificate is invalid. Signing certificate

Test session results, code coverage, and logs:
    /Users/crifan/Library/Developer/Xcode/DerivedData/WebDr

Testing failed:
    Signing certificate is invalid. Signing certificate "Ap
    Testing cancelled because the build failed.

** TEST FAILED **
```



原因：自己的Apple苹果（开发者）账号过期了。不可用，没法给代码code sign了。

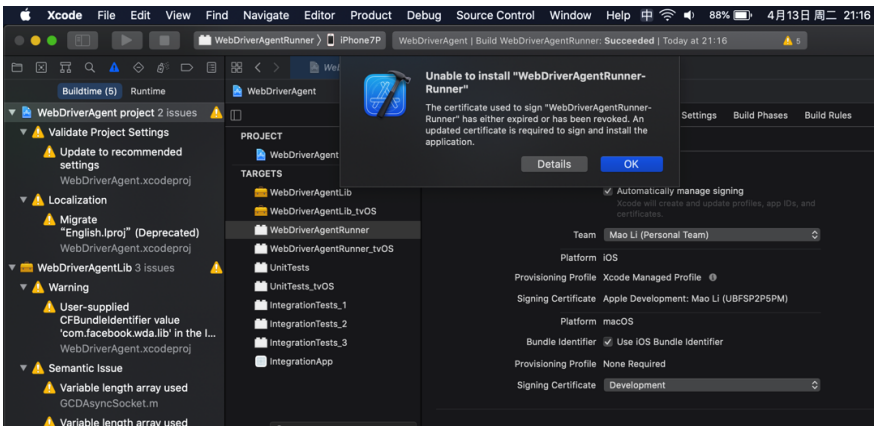
解决办法：花钱，给苹果开发者账号续费。价格：99美元/年。

XCode报错：The certificate used to sign has either expired or has been revoked

其他类似的问题：

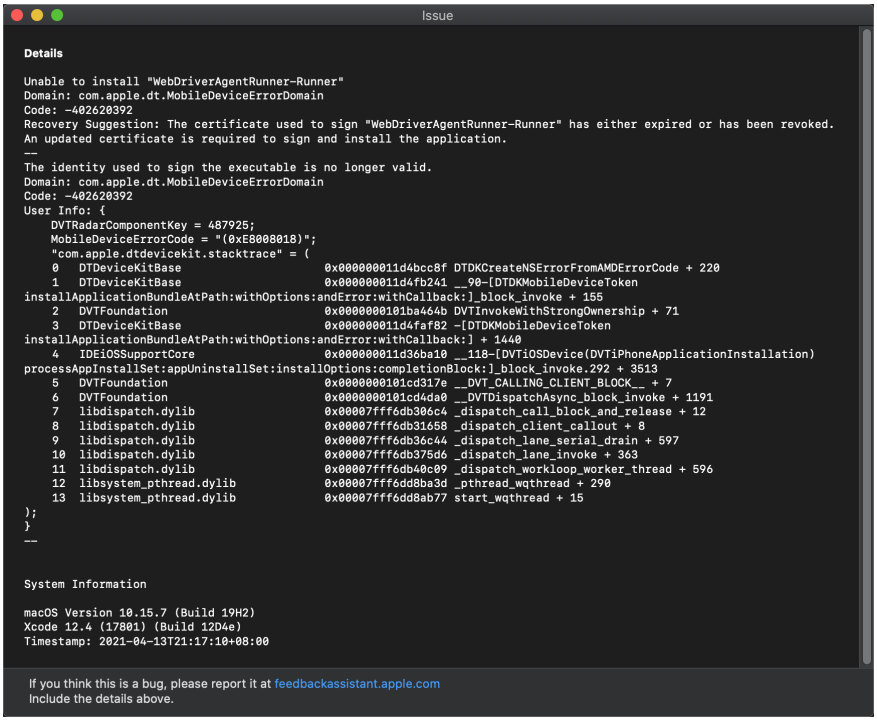
XCode中，报错

```
Unable to install "WebDriverAgentRunner-Runner"
The certificate used to sign "WebDriverAgentRunner-Runner"
```



点击 `Details` 还可以看到详情：

```
Details

Unable to install "WebDriverAgentRunner-Runner"
Domain: com.apple.dt.MobileDeviceErrorDomain
Code: -402620392
Recovery Suggestion: The certificate used to sign "WebDrive
--
The identity used to sign the executable is no longer valic
Domain: com.apple.dt.MobileDeviceErrorDomain
Code: -402620392
User Info: {
    DVTRadarComponentKey = 487925;
    MobileDeviceErrorCode = "(0xE8008018)";
    "com.apple.dtdevicekit.stacktrace" = (
     0   DTDeviceKitBase                    0x000000011d4b
     1   DTDeviceKitBase                    0x000000011d4f
     2   DVTFoundation                      0x0000000101ba
     3   DTDeviceKitBase                    0x000000011d4f
     4   IDEiOSSupportCore                  0x000000011d36
     5   DVTFoundation                      0x0000000101cc
     6   DVTFoundation                      0x0000000101cc
     7   libdispatch.dylib                  0x00007fff6db3
     8   libdispatch.dylib                  0x00007fff6db3
     9   libdispatch.dylib                  0x00007fff6db3
     10  libdispatch.dylib                  0x00007fff6db3
     11  libdispatch.dylib                  0x00007fff6db4
     12  libsystem_pthread.dylib            0x00007fff6dd8
     13  libsystem_pthread.dylib            0x00007fff6dd8
);
}
--

System Information

macOS Version 10.15.7 (Build 19H2)
Xcode 12.4 (17801) (Build 12D4e)
Timestamp: 2021-04-13T21:17:10+08:00
```



原因：苹果开发者账号过期了，没续费。导致证书不可用。

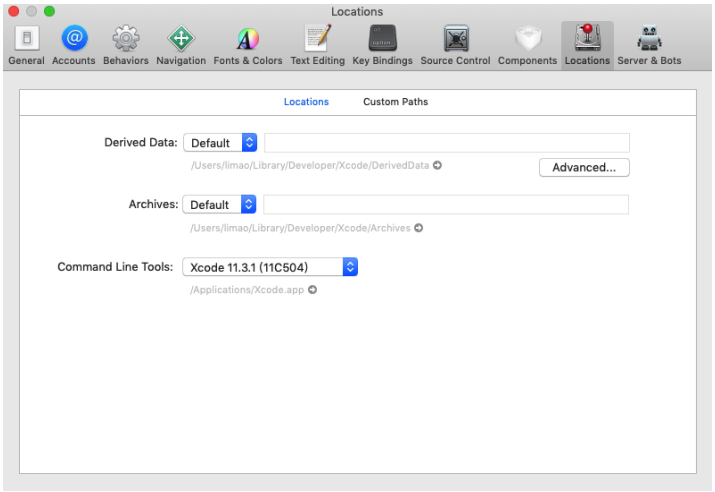
解决办法：同上，续费即可。

xcodebuild报错：xcode-select error tool xcodebuild requires Xcode

如果运行xcodebuld报错：

```
xcode-select: error: tool 'xcodebuild' requires Xcode, but
```

- **原因**：没有安装XCode 或 虽然已安装XCode，但是没启用XCode的命令行
- **解决办法**：去安装并开启XCode的命令行
- **步骤**：
  - 文字
    - `Xcode` -> `设置` -> `Locations` -> `Command Line Tools`，默认是**空**，下拉选择 `Xcode 11.3.1(11C504)`
  - 截图



安装后，即可查看版本信息：

```
~ □ xcodebuild -version
Xcode 11.3.1
Build version 11C504
```

xcodebuild报错：xcodebuild error missing value for key

如果没有iOS设备（如iPhone）插入到Mac中，则运行：

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDr
```

会报错：

```
□ ~/dev/xxx/crawler/appAutoCrawler/AppCrawler/iOSAutomatio
xcodebuild: error: missing value for key 'id' of option 'D
```

开发心得

wda如何同时测试多个设备

- **问**：如何用wda同时测试多个设备?
- **答**：使用不同端口转发

具体做法举例：

```
iproxy 8100 8100
iproxy 8101 8101
iproxy 8102 8102
```

代码中，本地连接不同端口：

```
gWdaClient0 = wda.Client('http://localhost:8100')
gWdaClient1 = wda.Client('http://localhost:8101')
gWdaClient2 = wda.Client('http://localhost:8102')
```

即可。

感慨：对于apple的态度

见到别人有提到：

> Apple公司因其无与伦比的设计，让无数果粉为之迷恋
>
> 但作为iOS测试人员，也因为iOS系统封闭和不开放库苦不堪言，羡慕死Android测试

对此深有体会，不能再同意更多：

- 消费者：对于apple产品觉得很好看，很喜欢
- 测试、自动化人员：苦不堪言
  - 原因：apple生态封闭，不开放
    - 虽然提供了XCTest，但是很不好用
    - iOS，Mac等内部的库是不开放的
      - 没法直接用来做测试和自动化
      - -》Facebook的WebDriverAgent（后由Appium维护）已经做到了
        - 用工具从 私有的库中dump出头文件和api接口
        - 但是实际用起来，仍然是各种bug和不兼容
          - 包括但不限于（后续会介绍到的）各种坑
            - 获取不到源码

- 只能获取部分源码
- 获取源码会导致test manager崩溃（需要重装WebDriverAgentRunner）
- 无法完美支持元素visible属性
- 获取到的源码很混乱
  - 比如
    - 包含了前一页（甚至几页）的xml源码

weditor

`weditor` 是 `facebook-wda` 的作者，为了方便调试移动端设备，除了 `uiautomator2` 所支持的 `Android` 设备之外，也支持iOS设备。

- `weditor`
  - 安装
    - `pip install -U weditor`
  - GitHub
    - alibaba/web-editor: web editor for atx
      - https://github.com/alibaba/web-editor

使用weditor调试iOS设备

启动weditor：

```
weditor
```

- 注：旧的运行方式是

  ```
  python -m weditor
  ```

然后会自动（调用浏览器）打开地址：

http://localhost:17310/

选择目标设备类型是：`iOS`

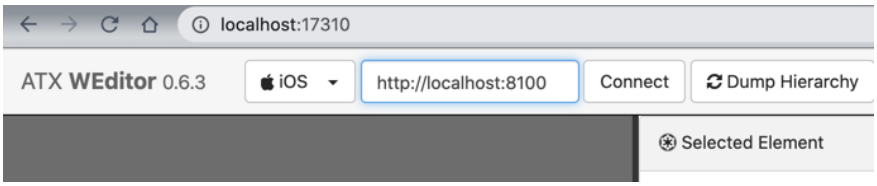
然后输入wda服务的地址，比如：

http://192.168.31.58:8100

- 注：如果之前已用 `iproxy 8100 8100` 端口转发，则可以直接输入 `localhost` 的地址

http://localhost:8100

再点击 `Connect`



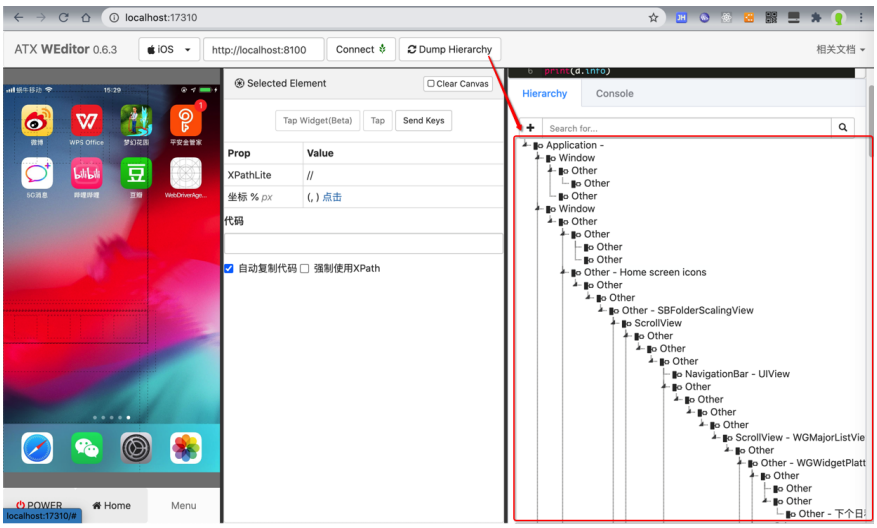
显示绿色（的红绿灯 标识），表示连上了，且能看到iOS设备的画面了：

注：想要看到iOS设备的实时画面，需要点击一次Dump Hierarchy，才能看到实时页面。

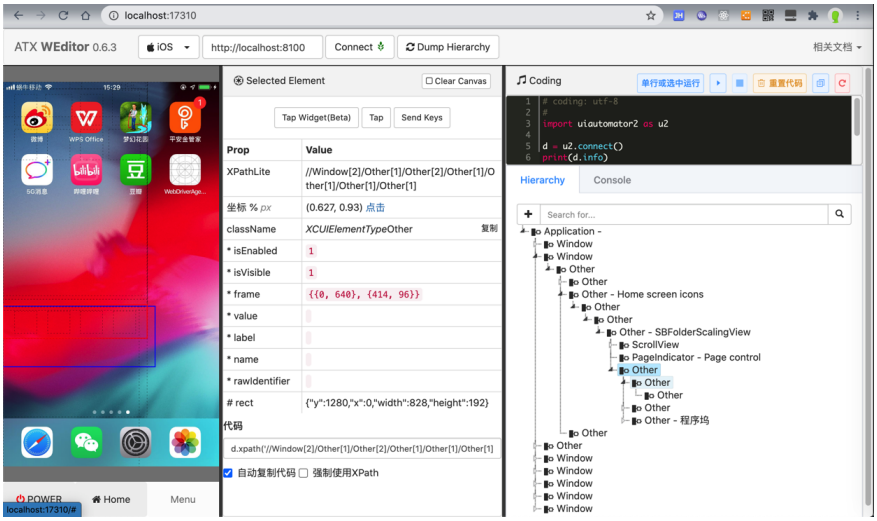
然后就是，对于weditor的使用了：

比如：

点击 `Dump Hierarchy` ，可以显示出页面元素结构：



以及点击元素，查看到属性：



iOS的各种坑

在用 `facebook-wda` + `WebDriverAgent` 去自动化测试iOS设备期间，遇到各种**坑**。

其中多数坑，都是苹果官方的API不稳定或有bug导致的，少数是 `WebDriverAgent` 或 `facebook-wda` 的。

wda找到元素，点击元素，竟然偶尔会无效

对于页面：

代码去找到并点击 个人所得税：

```
    isIntoDetailOk = CommonUtils.multipleRetry(
        {
            "functionCallback": self.appStoreSearchResultIn
            "functionParaDict": {
                "appName": appName,
            }
        },
        maxRetryNum = 10,
        sleepInterval = 0.5,
    )

def appStoreSearchResultIntoDetail(self, appName):
    """for AppStore search result list page
        try find first match result
            then click into detail page

    Args:
        appName (str): app name
    Returns:
        bool, dict
            bool: is into detail page or not
    Raises:
    """
    isIntoDetailOk = False
    """
        搜索结果列表页 京东 重新下载:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeCollectionView type="XC
                        <XCUIElementTypeCell type="XCUIElem
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIElem
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                    </XCUIElementTypeCollectionView>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        搜索结果列表页 美团 获取:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeCollectionView type="XC
                        <XCUIElementTypeCell type="XCUIElem
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIElem
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                    </XCUIElementTypeCollectionView>
```

```
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    parentCollectionViewClassChain = "/XCUIElementTypeColle
    firstMatchCellQuery = {"type":"XCUIElementTypeCell", "
    firstMatchCellQuery["parent_class_chains"] = [ parentC
    foundAndClicked = self.findAndClickElement(query=firstM
    isIntoDetailOk = foundAndClicked
    return isIntoDetailOk
```

始终都是正常的：可以找到并点击元素，然后会进入app下载的详情页

但是目前调试期间，先后遇到2次了

只是对于特殊的app名字：`个人所得税`，出现了虽然代码中能找到元素，并点击了元素：

```
[200609 15:45:29][DevicesMethods.py 851] True to Clicked e
```

但是实际上：

竟然点击没生效

-》页面没有进入后续的详情页

-》但是同样代码，重新测试，却又正常，可以点击元素进入详情页了：

很是诡异。

根本原因：至今未知。

暂时在列表页前后加上等待时间：

```
# Special: try add some wait time to avoid some special cas
# for 个人所得税 search result page, found and click 个人所得
time.sleep(0.5)
isIntoDetailOk = CommonUtils.multipleRetry(
    {
        "functionCallback": self.appStoreSearchResultIntoDe
        "functionParaDict": {
            "appName": appName,
        }
    },
    maxRetryNum = 10,
    sleepInterval = 0.5,
)
if not isIntoDetailOk:
    respInfo = "Fail to into app detail page for %s" % appN
    return isInstallOk, respInfo
# Special: try add some wait time to avoid some special cas
# for 个人所得税 search result page, found and click 个人所得
time.sleep(0.2)
```

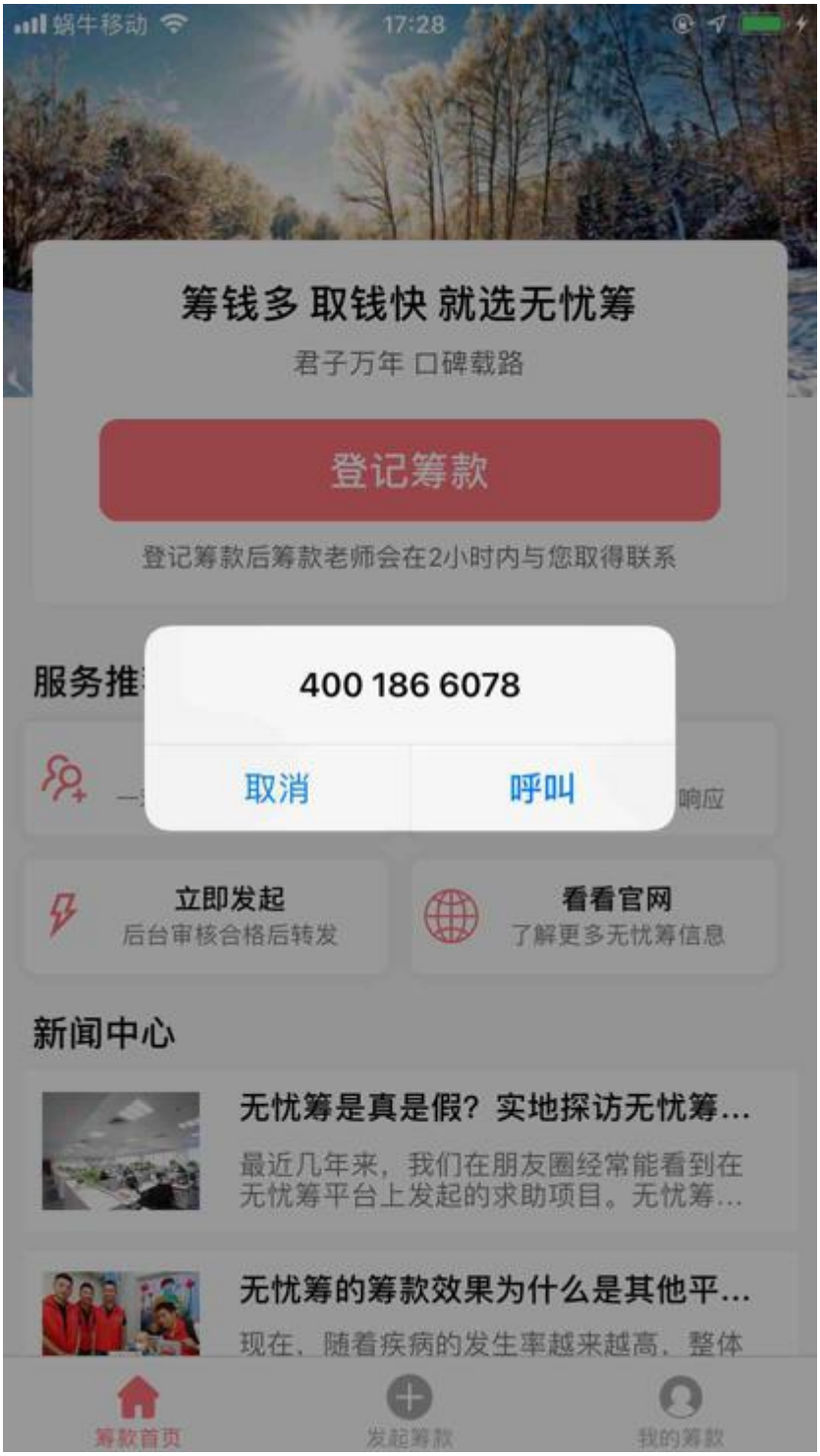
希望，或许能稳定些，或许能规避此问题？

详见：

【未解决】facebook-wda点击个人所得税元素无效：没有进入AppStore详情页

偶尔会遇到 通过坐标值点击元素 无效 实际上误点击别的位置

对于页面：



代码已找到了 取消 按钮，然后去点击 其中间坐标位置

```
clickCenterPosition(curSession, cancelSoup.attrs)

def clickCenterPosition(curSession, elementAttrDict):
    x = int(elementAttrDict["x"])
    y = int(elementAttrDict["y"])
    width = int(elementAttrDict["width"])
    height = int(elementAttrDict["height"])
    centerX = x + int(width / 2)
    centerY = y + int(height / 2)
    curSession.click(centerX, centerY)
    logging.info("Clicked [%s, %s]", centerX, centerY)
```

之前此点击元素中间位置的代码工作都是正常的

唯独这此，点击 取消按钮 后，实际上是点击了：呼叫 按钮的位置，导致进入 呼叫 界面：

最后无奈，只能绕过这个bug，换用别的方式去点击元素：

用wda的query去查找元素，通过元素点击本身

```
        # parentOtherSoup = callSoup.parent
        # if parentOtherSoup:
        #     parentParentOtherSoup = parentOtherSoup.paren
        #     if parentParentOtherSoup:
        #         cancelSoup = parentParentOtherSoup.find(
        #             "XCUIElementTypeButton",
        #             attrs={"enabled":"true", "visible":"t
        #         )
        #         if cancelSoup:
        #             clickCenterPosition(curSession, cance
        #             foundAndProcessedPopup = True

        # above click position not work for 取消 !!!
        # change to find 取消 then click element

        cancelButtonQuery = {"type":"XCUIElementTypeButton"
        foundAndClicked = findAndClickElement(curSession, c
        foundAndProcessedPopup = foundAndClicked
```

才可以：点击取消 让弹框消失。

详见：

【已解决】自动抓包iOS的app无忧筹：弹框呼叫拨打电话

【后记1】

又在：

【未解决】自动抓包iOS的app京东金融：弹框想给您发送通知允许

遇到同样的问题：

bs4中搜索到了 允许 按钮，去点击 通过点击允许按钮的中间坐标值，结果实际上却是点击了：另外一个按钮 不允许 。。。

然后无奈，只能想办法用wda的query去查询元素 允许，再通过元素点击估计就可以了。

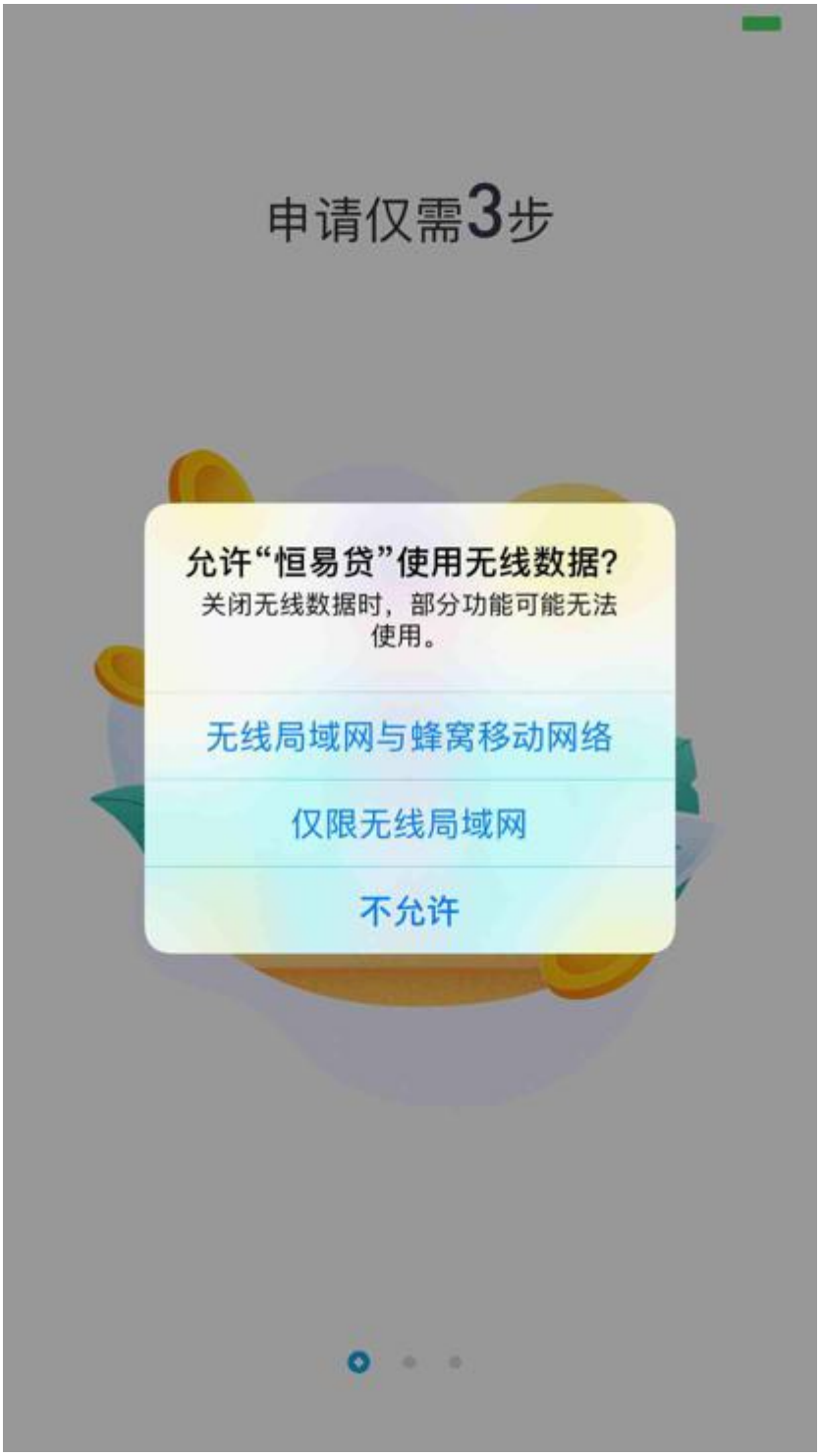
【后记2】

又在：

【未解决】自动抓包iOS的app恒易贷：弹框使用无线数据无线局域网与蜂窝移动网络

遇到同样问题：

对于页面：


申请仅需3步
允许“恒易贷”使用无线数据?
关闭无线数据时，部分功能可能无法使用。
无线局域网与蜂窝移动网络
仅限无线局域网
不允许

都已经用代码：

```
    wifiCellularChainList = [
        {
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    wifiCellularSoup = utils.bsChainFind(soup, wifiCellular
    if wifiCellularSoup:
        clickCenterPosition(curSession, wifiCellularSoup.at
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

查到并点击了 无线局域网与蜂窝移动网络 按钮的中间位置，但是实际上点击的是：不允许

导致后来app无法访问网络，再次启动app后，也提示请开启网络权限。

只能去改为，wda的元素查找，找到元素后，根据元素去click点击：

```
        wifiCellularSoup = CommonUtils.bsChainFind(soup, wi
        if wifiCellularSoup:
            # self.clickElementCenterPosition(wifiCellularS
            # foundAndProcessedPopup = True

            # found 无线局域网与蜂窝移动网络 but actually click
            # change to wda query element then click by ele
            curName = wifiCellularSoup.attrs["name"] # 好
            wifiCellularButtonQuery = {"type":"XCUIElement"
            foundAndClicked = self.findAndClickElement(wifi
            foundAndProcessedPopup = foundAndClicked
        return foundAndProcessedPopup
```

才可以。

【后记3】

由于经常遇到此问题，所以后来专门去提取逻辑到独立函数中，详见：

元素处理 · iOS自动化测试利器：facebook-wda

坑：元素查找条件 都写的最完整，不能再详细了，但是却会出现 可以查询到 找到 多个元素

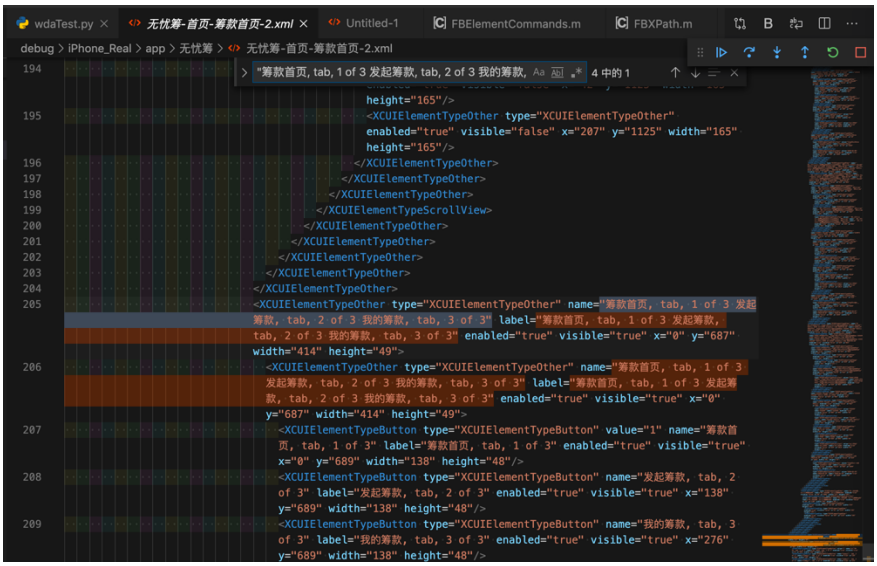
比如页面：

左下角的 3个tab页的父级元素，对应 locator，调试出现警告：

```
[200515 14:23:27][ParsePage.py 1019] Found 2 same node fro
```

提示上述locator可以找到2个元素，然后去xml源码中看看，果然是的：

```
    <XCUIElementTypeOther type="XCUIElementTypeOther" name=
    <XCUIElementTypeOther type="XCUIElementTypeOther" name=
。。。
```



就是：底部3个按钮主菜单 的parent 和 parent的parent

-》坑就是：

如果通过上述（最详尽的）条件去定位元素，则理论上是会出现多个的

-》无法完美精准定位查询到某个想要的元素。

详见：

【未解决】自动抓包iOS的app：无忧筹点击首页的筹款首页后无法返回

wda获取到了switch的值，但是是错的

对于页面：

对应xml

```
<XCUIElementTypeCell type="XCUIElementTypeCell" value="0" e
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
    <XCUIElementTypeStaticText type="XCUIElementTypeStatic
    <XCUIElementTypeSwitch type="XCUIElementTypeSwitch" va
</XCUIElementTypeCell>
```

已经可以用代码：

```
newAuthenticateValue = newManualProxyValue["authenticate"]
authSwitchQuery = {"type":"XCUIElementTypeSwitch", "name":
authSwitchQuery["parent_class_chains"] = [ parentCellClass(
foundAuth, respInfo = self.findElement(authSwitchQuery, tir
```

找到 鉴定 对应的switch

但是获取其value值：

```
authSwitchElement = respInfo
curAuthValue = authSwitchElement.value # '0'
```

竟然是：`'0'`

而不是真正实际的值：`'1'`

规避办法：

最后无奈只能改用别的方式（bs的find，获取到xml源码）去获取值

虽然速度慢点，但是至少值是准的：

```
curAuthValueStr = ""
# curAuthValue = authSwitchElement.value # '0'
# curAuthValueStr = str(curAuthValue)
# Special: sometime wda element value is WRONG, actual is
# so change to bs find then get value from page source xml
curPageXml = self.get_page_source()
soup = CommonUtils.xmlToSoup(curPageXml)
authSwitchChainList = [
    {
        "tag": "XCUIElementTypeTable",
        "attrs": self.FullScreenAttrDict
    },
    {
        "tag": "XCUIElementTypeCell",
        "attrs": {"enabled":"true", "visible":"true", "x":"
    },
    {
        "tag": "XCUIElementTypeSwitch",
        "attrs": {"enabled":"true", "visible":"true", "name
    },
]
authSwitchSoup = CommonUtils.bsChainFind(soup, authSwitchCh
if authSwitchSoup:
    curAuthValue =  authSwitchSoup.attrs.get("value", None)
    if curAuthValue:
        curAuthValueStr = str(curAuthValue)
```

详见：

【已解决】facebook-wda获取鉴定的value值是错误的

wda找到元素，但是无法用clear_text清除值value值

界面中：

用代码已经找到 服务器 元素了：

```
newServerValue = newManualProxyValue["server"]
serverFieldQuery = {"type":"XCUIElementTypeTextField", "nar
serverFieldQuery["parent_class_chains"] = [ parentCellClass
isFound, respInfo = self.findElement(query=serverFieldQuery
logging.debug("isFound=%s, respInfo=%s", isFound, respInfo)
if isFound:
    curElement = respInfo
```

但是去清除当前的值

```
curElement.clear_text()
```

却不起效果

最后无奈用 `set_text()` 传入多个 `\b` ，通过一个个删除字符的方式实现了删除输入的值的效果

```
def iOSClearText(self, curElement):
    """iOS clear current element's text value
        Note: clear_text not working, so need use other wo

    Args:
        curElement (Element): wda element
    Returns:
    Raises:
    """
    # curElement.click()
    # curElement.clear_text()
    # curElement.tap_hold(2.0) # then try select All -> De
    backspaceChar = '\b'
    maxDeleteNum = 50
    curElement.set_text(maxDeleteNum * backspaceChar)
    return
```

调用：

```
curElement = respInfo
if isNeedClear:
    # before set new value, clear current value
    self.iOSClearText(curElement)
curElement.set_text(text)
```

间接实现clear text的效果。

详见：

【已解决】facebook-wda中元素clear清除文本值无效

无法获取元素value值

类似于页面：

AppStore中，正在下载app

xml源码是：

```
<XCUIElementTypeButton type="XCUIElementTypeButton" value='
```

但是通过

文件：`/Users/limao/.pyenv/versions/3.8.0/Python.framework/Versions/3.8/lib/python3.8/site-packages/wda/__init__.py`

```
@property
def value(self):
    # curValue = self._prop('attribute/value')
    curValue = self._prop('attribute/wdValue')
    if DEBUG:
        print("curValue=%s" % curValue)
    return curValue
```

对应着代码：

文件：`refer/WebDriverAgent/WebDriverAgentLib/Commands/FBElementCommands.m`

```
    [[FBRoute GET:@"/element/:uuid/attribute/:name"] respor

+ (id<FBResponsePayload>)handleGetAttribute:(FBRouteRequest
{
  FBElementCache *elementCache = request.session.elementCac
  XCUIElement *element = [elementCache elementForUUID:reque
  if (nil == element) {
    return FBResponseWithStatus([FBCommandStatus staleEleme

  }
  id attributeValue = [element fb_valueForWDAttributeName:n
  attributeValue = attributeValue ?: [NSNull null];
  return FBResponseWithObject(attributeValue);
}
```

却获取不到value值，始终是null：

```
20200609 01:49:34 connectionpool.py:428  DEBUG   http://loc
20200609 01:49:34 __init__.py:178  DEBUG   Return (213ms):
      "value" : {
        "element-6066-11e4-a52e-4f735466cecf" : "32000000-0
        "attribute\/visible" : true,
        "attribute\/name" : "正在下載",
        "attribute\/value" : null,
        "attribute\/accessible" : true,
        "text" : "正在下載",
        "label" : "正在下載",
        "rect" : {
          "y" : 308,
          "x" : 154,
          "width" : 74,
          "height" : 30
        },
        "type" : "XCUIElementTypeButton",
        "name" : "XCUIElementTypeButton",
        "ELEMENT" : "32000000-0000-0000-4122-000000000000"
      },
      "sessionId" : "710DB6C7-3669-4677-B479-C006692CC3F6"
    }

20200609 01:49:48 connectionpool.py:428  DEBUG   http://loc
20200609 01:49:48 __init__.py:178  DEBUG   Return (204ms):
      "value" : null,
      "sessionId" : "710DB6C7-3669-4677-B479-C006692CC3F6"
    }
```

详见：

【未解决】研究facebook-wda和WebDriverAgent中attribute/value始终是null无法获取有效值

偶尔页面中有内容刷新，动画进行中，则无法方便的获取到页面源码

详见：

【未解决】WebDriverAgent获取iPhone页面源码报错：Code 5 Error kAXErrorIPCTimeout getting snapshot for element

页面源码中，个别元素的最大x值超出屏幕

比如页面：

中的店铺热卖

相关部分xml是：

```
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="t
    <XCUIElementTypeButton type="XCUIElementTypeButton" nam
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
</XCUIElementTypeCell>
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="t
    <XCUIElementTypeButton type="XCUIElementTypeButton" nam
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
</XCUIElementTypeCell>
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="t
    <XCUIElementTypeButton type="XCUIElementTypeButton" nam
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
</XCUIElementTypeCell>
```

店铺热卖 的：x1=x0+width=264+112=376 大于 屏幕宽度375

导致原先代码逻辑判断出错：以为元素不在bottom底部区域，而被过滤掉，找不到菜单

最后是额外加了特殊处理，才可以保留和找到此菜单

详见：

【已解决】自动抓包iOS公众号：niuer-tmall中丢失部分主菜单

元素query时不完全支持visible属性

最新的结果也支持：有时候支持，有时候不支持

比如对于页面：

中，xml是：

```
<XCUIElementTypeButton type="XCUIElementTypeButton" name="
```

query查询条件中，加了visible：

```
{'enabled': 'true', 'height': '49', 'label': '牛尔天猫', 'na
```

就找不到，log是：

```
[200430 17:00:39][__init__.py 164] Shell: curl -X POST -d
[200430 17:00:39][connectionpool.py 221] Starting new HTTP
[200430 17:00:40][connectionpool.py 428] http://localhost:8
[200430 17:00:40][__init__.py 178] Return (175ms): {
      "value" : {
        "error" : "no such element",
        "message" : "unable to find an element using 'class
。。。
```

去掉visible：

```
{'enabled': 'true', 'height': '49', 'label': '牛尔天猫', 'na
```

就能找到。

详见：

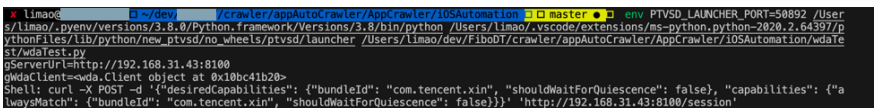
【已解决】Python中facebook-wda和WebDriverAgent中是否可以支持displayed以及是否能替换visible

【已解决】自动抓包iOS公众号：niuer-tmall定位主菜单失败

【已解决】合并最新版WebDriverAgent后测试是否支持元素的visible属性的query查询

偶尔代码无法运行，要看看服务端test manager是否正常

调试时发现偶尔卡死：



始终无法继续运行了。

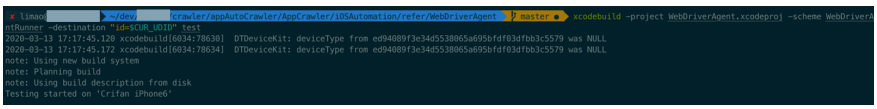
以为代码改动出了问题。

后来发现是：服务端挂了：

```
Testing failed:
    WebDriverAgentRunner:
        testRunner encountered an error (Encountered a prob
** TEST FAILED **
```

所以去重启服务：

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDr
```



直到看到：

```
2020-03-13 17:17:57.615841+0800 WebDriverAgentRunner-Runner
2020-03-13 17:17:57.659059+0800 WebDriverAgentRunner-Runner
```

即可。

获取源码xml

iOS中最大的坑，就是**获取页面源码** `xml` 期间，遇到的各种问题。

坑：即使查询条件和xml中内容正确匹配，也查询不到

对于页面：

xml是：

```
<XCUIElementTypeStaticText type="XCUIElementTypeStaticText'
```

去用：

```
{'value': '可申请(元) 200,000', 'name': '可申请(元) 200,000',
```

以及 去掉y的：

```
{'value': '可申请(元) 200,000', 'name': '可申请(元) 200,000',
```

都查不到元素。

不过，去掉value，name，label后：

```
{'enabled': 'true', 'x': '85', 'y': '226', 'width': '244',
```

是可以查询到元素的，所以很是诡异。

其原因，自己推测是此处的（value等）值有问题

但是具体的值是不是我猜测的

`可申请(元) 200000`

则无需，也懒得再去试了。

更重要的是，对于：

`可申请(元) 200,000`

页面上的内容的显示，是肉眼可见的分成了2部分

```
可申请(元)
200,000
```

且显示的样式都不同

-> 所以十分怀疑是：

iOS内部的元素和代码，其实本身就是这2部分是分开的

只不过是输出xml时，混在了一起

-> 导致通过value（name，label）才找不到元素的

-> 去掉value等值后，只用x、y等坐标值，就能找到：说明是对应着页面上的其中某一个元素

要么是 可申请(元)，或者是200,000

总之是：

iOS内部页面内容，和输出xml代码之间，一直做的很垃圾。

或者说故意做的很垃圾，让你很难自动化测试iOS。

详见：

【不去解决】自动抓包iOS的app恒易贷：找不到元素可申请元200000

坑：界面上按钮有文字，但是源码中没有文字

界面上：

本来希望去：写规则去查找button，且name是立即进入

结果源码中

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
        <XCUIElementTypeScrollView type="XCUIElementTypeScr
            <XCUIElementTypeImage type="XCUIElementTypeImag
        </XCUIElementTypeScrollView>
        <XCUIElementTypeButton type="XCUIElementTypeButton"
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
```

没有我们希望的文字：立即进入

注：目测看起来，这个 立即进入 的button的文字 不是属于button图片本身，而是普通文字，只不过xml源码中，的确找不到

这样就影响了后续代码逻辑的判断，无法准确判断当前页面的按钮，是否是最后一页了。

详见：

【未解决】自动抓包iOS的app：左滑引导页进入首页

不爽的点：页面类似，但xml源码差异很大

对于页面：

但是对应xml：

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
    <XCUIElementTypeTable type="XCUIElementTypeTable" name=
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
        <XCUIElementTypeImage type="XCUIElementTypeImage" 
        <XCUIElementTypeStaticText type="XCUIElementTypeSta
        <XCUIElementTypeButton type="XCUIElementTypeButton"
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
```

很明显，页面中的 刷新试试 明显是一个按钮，是没问题的

-> 后续就容易写规则去匹配和处理

但是后来遇到和上面很类似的页面：

可见页面上 `再刷新下` 也是一个按钮

但发现xml却是：

```
<XCUIElementTypeOther type="XCUIElementTypeOther" name="系统
    <XCUIElementTypeOther type="XCUIElementTypeOther" name=
        <XCUIElementTypeOther type="XCUIElementTypeOther" n
            <XCUIElementTypeImage type="XCUIElementTypeImag
            <XCUIElementTypeStaticText type="XCUIElementTyp
        </XCUIElementTypeOther>
        <XCUIElementTypeOther type="XCUIElementTypeOther" n
            <XCUIElementTypeOther type="XCUIElementTypeOthe
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
```

再刷新下 却是一个 XCUIElementTypeOther，而不是 XCUIElementTypeButton

-> 后续代码去处理和写匹配逻辑，就显得很不顺，让人很不爽。

-> 如果也是和前面一样的XCUIElementTypeButton，就容易统一成一个逻辑去处理，更加通用，效率更高。

-> 现在没法统一，效率很低，逻辑上显得很冗余

总体结论：

页面上的元素，和xml源码内容，很多时候，对不上，甚至完全对不上，驴唇不对马嘴的感觉。

详见：

【未解决】自动抓包iOS的app京东金融：网络不稳定刷新试试

【未解决】自动抓包iOS的app京东金融：系统正在开小差再刷新下

坑：有些页面 获取到的源码实际上是空的 没有包含页面元素的源码

比如页面：

希望获取源码中包含弹框部分的内容

但是实际上获取到的是：

```
<?xml version="1.0" encoding="UTF-8"?>
<XCUIElementTypeApplication type="XCUIElementTypeApplicatio
  <XCUIElementTypeWindow type="XCUIElementTypeWindow" enab
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
      <XCUIElementTypeOther type="XCUIElementTypeOther" ena
        <XCUIElementTypeOther type="XCUIElementTypeOther" e
          <XCUIElementTypeOther type="XCUIElementTypeOther"
        </XCUIElementTypeOther>
      </XCUIElementTypeOther>
    </XCUIElementTypeOther>
  </XCUIElementTypeWindow>
  <XCUIElementTypeWindow type="XCUIElementTypeWindow" enab
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
  </XCUIElementTypeWindow>
  <XCUIElementTypeWindow type="XCUIElementTypeWindow" enab
  <XCUIElementTypeWindow type="XCUIElementTypeWindow" enab
    <XCUIElementTypeStatusBar type="XCUIElementTypeStatusBa
  </XCUIElementTypeWindow>
  <XCUIElementTypeWindow type="XCUIElementTypeWindow" enab
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
  </XCUIElementTypeWindow>
</XCUIElementTypeApplication>
```

即：

中间主体内容是空的

没有包含我们希望看到的 弹框部分

详见：

【未解决】自动抓包iOS的app益路同行：弹框退出游戏

坑：页面中图片明显可见，但是xml源码中visible=false表示不可见

页面中的中间部分的2个图片：

此处xml源码竟然是：

```
                    <XCUIElementTypeCell type="XCUIElement
                      <XCUIElementTypeImage type="XCUIEleme
                      <XCUIElementTypeStaticText type="XCU
                      <XCUIElementTypeProgressIndicator typ
                      <XCUIElementTypeStaticText type="XCU
                      <XCUIElementTypeStaticText type="XCU
                      <XCUIElementTypeStaticText type="XCU
                      <XCUIElementTypeStaticText type="XCU
                    </XCUIElementTypeCell>
```

其中

```
<XCUIElementTypeImage type="XCUIElementTypeImage" enabled="
```

即：

只有一个Image节点，（当前可能本身就是一张图，但是从app中看起来不像，还是像2张图）并且还是visible=false，即不可见！

你妹的，那还怎么解析出有效节点，根本没法提取有效节点，和后续抓取。

详见：

【未解决】自动抓包工具抓包iOS的app：善友筹

坑：app内部某一层的页面中的xml源码，竟然还保留（之前的几层）父级的元素

比如

某个二级页面：

康爱公社-二级页面-百万医保补充互助社.jpg

其中，正常的符合预期的是，页面xml源码中，有页面中的元素，比如顶部的第二排的 互助公约 资助公示 本期分摊 联系客服 等

但是点击了 资助公示 后，进入 三级页面：

康爱公社-三级页面-资助公示-弹框提醒.jpg

竟然其中xml源码中，还有 前一页的页面元素：


```
<XCUIElementTypeOther type="XCUIElementTypeOther" enab
    <XCUIElementTypeStaticText type="XCUIElementTypeSta
</XCUIElementTypeOther>
```

其中可见，不仅存在之前页面的元素的xml，且竟然是visible=true，即：

表示当前页面可见。但是实际上不可见，不可能看到，前面几级页面的内容。

-》导致后续的基于xml源码判断元素的逻辑，就不可用了。完全混乱了。

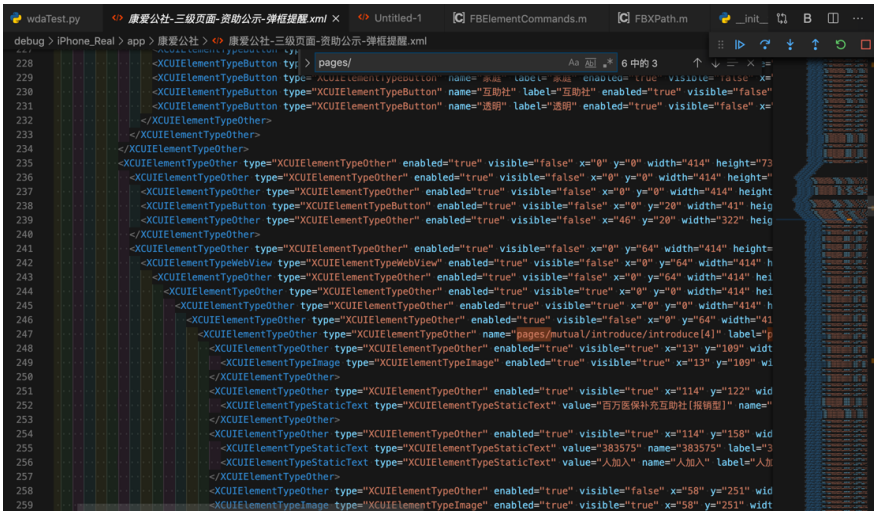
即：在第三级页面，也能找到第二级，甚至第一级页面的元素，以为是在第二级或第一级页面呢，无需返回，即可找到并点击相关元素，而实际上页面上，并不是第二级或第一级页面，屏幕上并没有这些元素。

使得后续页面跳转，完全失效。无法继续正常逻辑。

仔细去看xml源码中发现，有个特点：

会存在 `pages/xxx/xxx` 之类的元素：

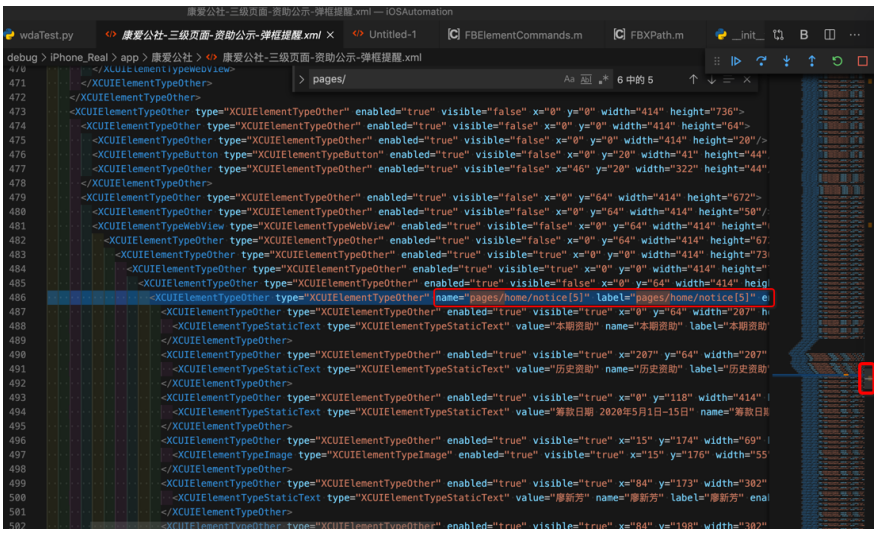
```
<XCUIElementTypeOther type="XCUIElementTypeOther" name="pag
。。。
```



且不止一个：

```
<XCUIElementTypeOther type="XCUIElementTypeOther" name="pag
```

。。。



其中有几个 `page/xxx`

-> 存在 当前页面 实际上 包含了 几个（前后一共几级的）页面的xml源码

详见：

【无法解决】iOS抓包app康爱公社：第三级页面中也能点击到第一级页面中的元素导致页面无法返回

【规避解决】iOS抓包app康爱公社：第三级别RestPage互助公约子页面无法返回

获取页面经常出现各种问题

比如：

【未解决】WebDriverAgent获取iPhone页面源码报错：Code 5 Error kAXErrorIPCTimeout getting snapshot for element

目前的结论是：

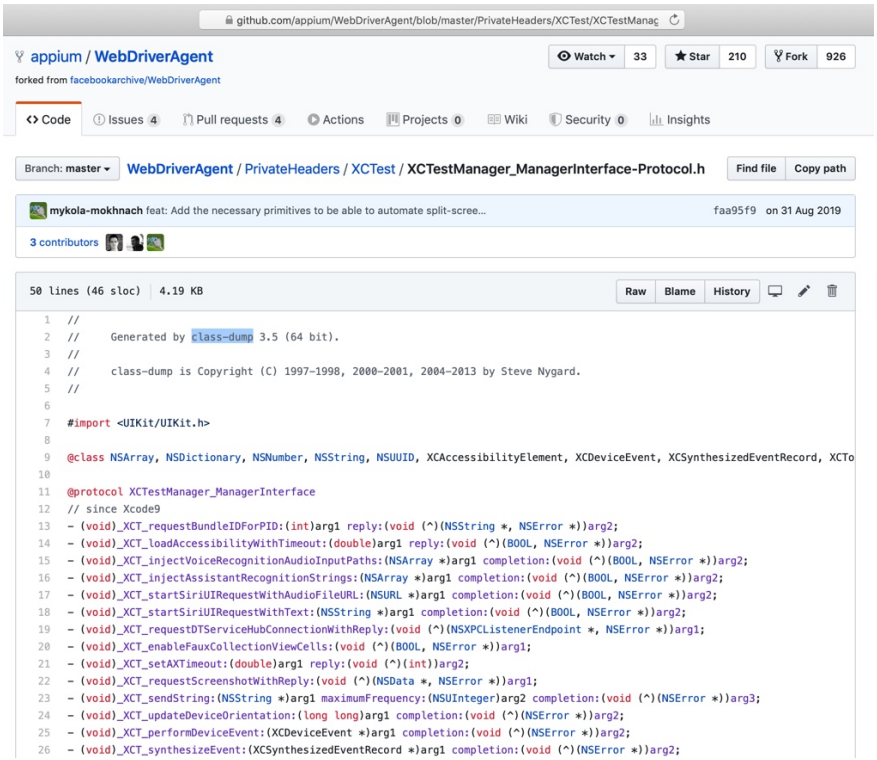
Apple方面，对于iOS设备的自动化测试，本身就是：从底层不太愿意支持

所以很多API接口，尤其是元素的是否可见的visible属性，就很难获取到

Appium的实现方式都是，想办法用第三方工具class-dump，RuntimeBrowser等从库中导出头文件，才看到有哪些API：

比如：

WebDriverAgent/XCTestManager_ManagerInterface-Protocol.h at master · appium/WebDriverAgent



然后利用这些私有的API，去实现想要的功能。

所以往往是：兼容性很差

尤其是iOS 升级到了13后，兼容性极其差，尤其是snapshot，即获取页面源码方面，会出现各种各样的问题，报各种各样的错

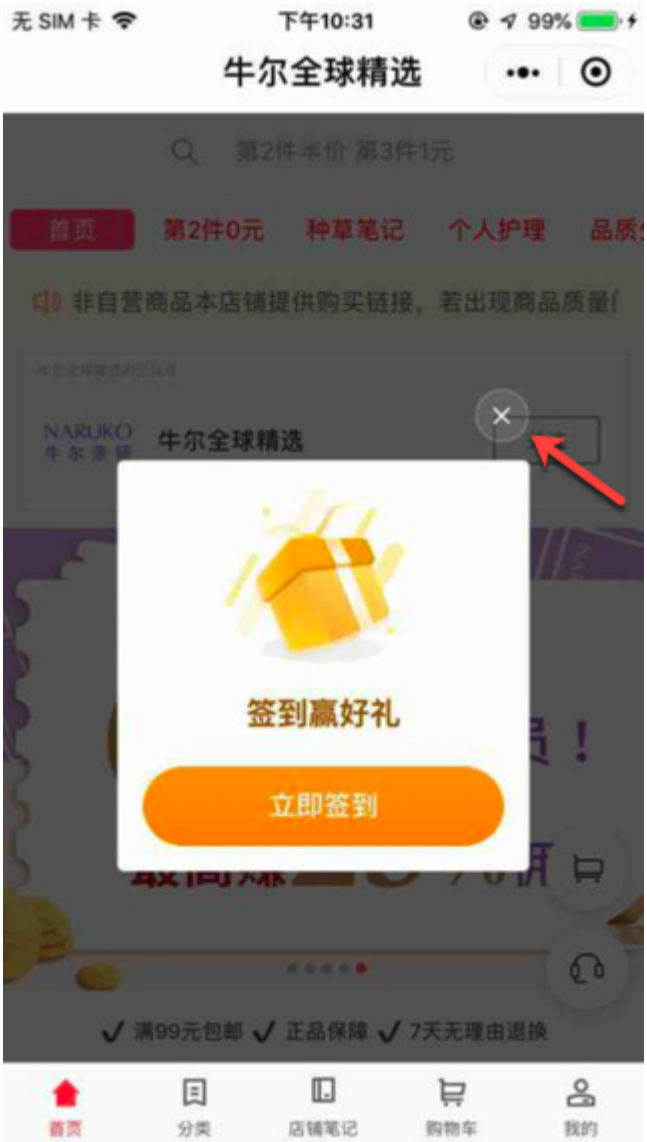
- Code 5 Error kAXErrorIPCTimeout getting snapshot for element
- Code=5 "Error -25216 getting snapshot for element
- Code=5 "Error kAXErrorServerNotFound getting snapshot for element

目前看来：

- 问题最多的：`iOS 13`
- 稍微好好一点的是：`iOS 12` 或 `iOS 11`

有些页面元素根本就无法获取到xml源码

比如：



相关部分源码是：

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
  <XCUIElementTypeStaticText type="XCUIElementTypeStaticTex
</XCUIElementTypeOther>
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
  <XCUIElementTypeStaticText type="XCUIElementTypeStaticTex
</XCUIElementTypeOther>
```

即：只有 签到赢好礼 和 立即签到

根本就找不到 弹框右上角的 x关闭 按钮的xml源码

-》导致无法定位按钮元素，无法点击关闭弹框

详见：

【未解决】自动抓包iOS公众号：小程序中可关闭弹框签到赢好礼

无法获取完整页面源码

比如：

现象是：

- 微信公众号搜索页 搜 unesunes后：
  - iOS 11的iPhone6P：获取不到完整源码
  - iOS 13的iPhone8P：能获取到源码
  - iOS 12的iPhone6：能获取到源码

其中iOS 11的iPhone6P获取源码期间，test manager能看到错误log

```
    t =  4793.27s Find: Identity Binding
2020-04-29 10:26:17.325280+0800 WebDriverAgentRunner-Runner
2020-04-29 10:26:17.325547+0800 WebDriverAgentRunner-Runner
2020-04-29 10:26:17.325677+0800 WebDriverAgentRunner-Runner
```

按道理，应该是换iOS 13的iPhone8P或iOS 12的iPhone6，但是之前又都是由于有各种问题，才换这个iOS 11的iPhone6P的：

- `iOS 13` 的 `iPhone8P`：
  - 优理氏的某个小程序的客服聊天页：获取源码，不仅是失败，而是会导致test manager崩溃
    - 不仅是崩溃，重启后也无效
      - 需要重新卸载后重新安装WebDriverAgentRunner-Runner才行
        - 但是依旧是获取源码导致崩溃，陷入死循环
- `iOS 12` 的 `iPhone6`：
  - 获取某些页面速度很慢
    - 竟然比（本身相对比较卡顿，不流畅的）iOS 11的iPhone6，还慢
  - 很久之前，更新WebDriverAgent代码之前
    - 动卡空间，点击 关注（还是进入）公众号
      - 会导致微信崩溃
    - 最新：合并最新WebDriverAgent代码后，目前暂时不崩溃了
      - 但是还是获取很多页面速度比较慢

总之是：

现在虽然有3个iPhone，不同硬件尺寸，不同iOS版本，竟然没有一个顺利运行的，各有各的问题

最终：

暂时只能换iPhone而规避解决：

换 iOS 13的iPhone8P 或 iOS 12的iPhone6，都可以：获取到完全的页面源码

详见：

【规避解决】自动抓包iOS公众号：获取微信公众号unesunes搜索结果页面源码失败

获取页面源码：偶尔会导致微信崩溃

最早的iPhone 6 + iOS 12.4.5，获取 动卡空间关注页 点击 关注（还是进入） 公众号后，结果微信崩溃

后来不崩溃了

可能的原因：更新了WebDriverAgent代码，内部解决或规避了之前的某个（WebDriverAgent或iOS或Xcode的API本身的）bug?

详见：

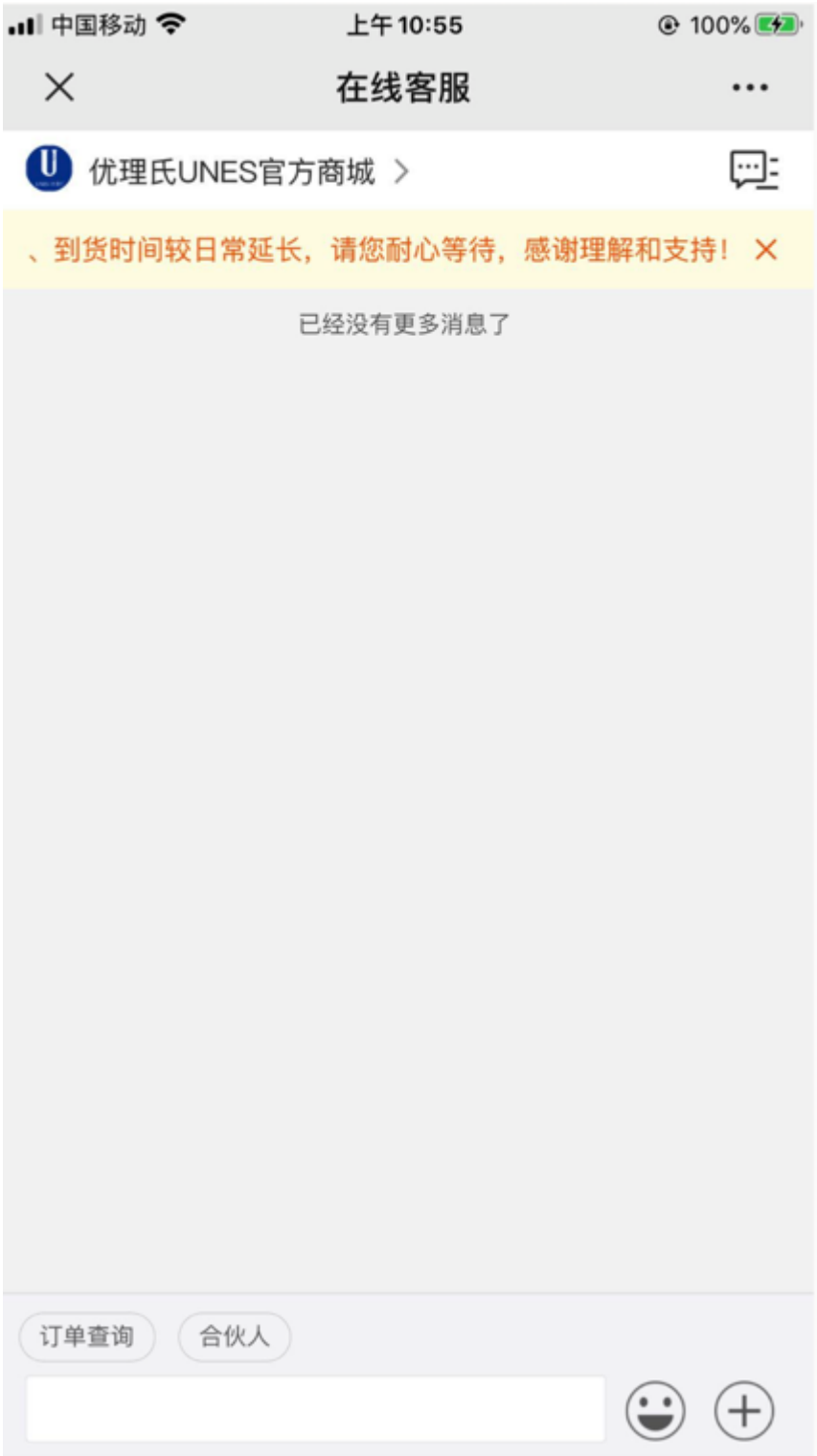
【记录】更新WebDriverAgent后测试iOS 12的iPhone6抓包微信公众号

获取页面源码：不仅无法获取源码，还会导致WebDriverAgent崩溃，要卸载后重新安装

WebDriverAgentRunner-Runner后才能重新使用

对于iOS 13.3.1的iPhone8P

去获取页面：



的源码时，会导致WebDriverAgent崩溃：

```
2020-04-28 10:49:13.283259+0800 WebDriverAgentRunner-Runner

Ignored Exception: Exception while decoding argument 0 (#1
<NSInvocation: 0x280fa4900>
return value: {v} void
target: {@?} 0x0 (block)
argument 1: {@} 0x0
argument 2: {@} 0x0

Exception: decodeObjectForKey: too many nested collections
(
    0   CoreFoundation                      0x00000001ba05
    1   libobjc.A.dylib                     0x00000001b9d7
    2   Foundation                          0x00000001ba54
    3   Foundation                          0x00000001ba54
    4   Foundation                          0x00000001ba31
    5   XCTAutomationSupport                0x000000010503
    6   Foundation                          0x00000001ba54
    7   Foundation                          0x00000001ba54
    8   Foundation                          0x00000001ba31
    9   XCTAutomationSupport                0x000000010501
    10  Foundation                          0x00000001ba54
    11  Foundation                          0x00000001ba54
    12  Foundation                          0x00000001ba55
    13  Foundation                          0x00000001ba32
    14  Foundation                          0x00000001ba33
    15  Foundation                          0x00000001ba54
    16  Foundation                          0x00000001ba54
    17  Foundation                          0x00000001ba31
    18  XCTAutomationSupport                0x000000010501
    19  Foundation                          0x00000001ba54
    20  Foundation                          0x00000001ba54
    21  Foundation                          0x00000001ba55
    22  Foundation                          0x00000001ba32
    23  Foundation                          0x00000001ba33
    24  Foundation                          0x00000001ba54
    25  Foundation                          0x00000001ba54
    26  Foundation                          0x00000001ba31
```

且重启WebDriverAgent也没用。

只能去iPhone中卸载掉之前的WebDriverAgentRunner-Runner后

重新安装WebDriverAgentRunner-Runner，才能继续使用。

但是本身上述页面，获取源码就导致崩溃，则无法解决。

经调试找到根本原因是：

`refer/WebDriverAgent/WebDriverAgentLib/Utilities/FBXCodeCompatibility.m`

中的

```
- (XCElementSnapshot *)fb_cachedSnapshot
{
  static dispatch_once_t onceToken;
  static BOOL isUniqueMatchingSnapshotAvailable;
  dispatch_once(&onceToken, ^{
    isUniqueMatchingSnapshotAvailable = [self respondsToSe
  });
。。。
```

的uniqueMatchingSnapshotWithError

-》属于Apple自己的API的bug，无法解决

详见：

【规避解决】XCode实时调试NSXPCConnection的 _XCT_fetchSnapshotForElement:attributes:parameters:reply错误

【规避解决】WebDriverAgent获取页面源码报错：xpc.exceptions NSXPCConnection com.apple.testmanagerd _XCT_fetchSnapshotForElement

获取源码速度非常慢

之前是：

【已解决】wda用source()获取页面源码xml速度极其慢

算是：从30秒左右，不知道做了啥，变成了，优化成了，10秒多

期间：

合并了最新的WebDriverAgent的代码：

- 【已解决】把旧版WebDriverAgent自己优化改动合并到最新版代码中
- 【已解决】验证最新WebDriverAgent代码功能上是否正常
- 【已解决】XCode编译最新版WebDriverAgent

也并没有解决 获取源码速度慢的问题

后来是：

- 【已解决】用XCode实时调试WebDriverAgent希望找到并解决获取页面源码慢的原因
- 【已解决】尝试解决facebook-wda和WebDriverAgent的获取源码很慢的原因

- 【未解决】WebDriverAgent和wda获取源码提速：尝试shouldLoadSnapshotWithAttributes参数
- 【未解决】调节Appium的Capability的参数去提高facebook-wda和WebDriverAgent获取源码的速度
- 【未解决】WebDriverAgent获取源码慢尝试调节参数：shouldUseTestManagerForVisibilityDetection
- 【已解决】Xcode调试WebDriverAgent研究fb_waitUntilSnapshotIsStable含义希望提高获取源码速度
- 【已解决】WebDriverAgent报错：Internal error Error Domain com.apple.dt.xctest.automation-support.error Code 5 Error kAXErrorServerNotFound getting snapshot for element
- 【已解决】WebDriverAgent中fb_waitUntilSnapshotIsStable的作用和含义即为何加上

最终是：

- 【已解决】WebDriverAgent获取源码慢尝试调节参数：FB_ANIMATION_TIMEOUT

解决了：

从10秒多，优化成，1~5秒左右，对于个别页面元素多时，才需要10秒+

常用代码段

在折腾 `facebook-wda` 期间，把各种常用的功能封装成了函数，现整理如下供参考：

get_cmd_lines 执行命令返回结果

```
def get_cmd_lines(cmd, text=False):
    # 执行cmd命令，将结果保存为列表
    resultStr = ""
    resultStrList = []
    try:
        consoleOutputByte = subprocess.check_output(cmd, sh
        try:
            resultStr = consoleOutputByte.decode("utf-8")
        except UnicodeDecodeError:
            # TODO: use chardet auto detect encoding
            # consoleOutputStr = consoleOutputByte.decode('
            resultStr = consoleOutputByte.decode("gb18030")

        if not text:
            resultStrList = resultStr.splitlines()
    except Exception as err:
        print("err=%s when run cmd=%s" % (err, cmd))

    if text:
        return resultStr
    else:
        return resultStrList
```

multipleRetry 多次尝试执行某个函数直到成功

```
def multipleRetry(functionInfoDict, maxRetryNum=5, sleepInt
    """
    do something, retry mutiple time if fail

    Args:
        functionInfoDict (dict): function info dict contai
        maxRetryNum (int): max retry number
        sleepInterval (float): sleep time of each interval
        isShowErrWhenFail (bool): show error when fail if t
    Returns:
        bool
    Raises:
    """
    doSuccess = False
    functionCallback = functionInfoDict["functionCallback"]
    functionParaDict = functionInfoDict.get("functionParaDi

    curRetryNum = maxRetryNum
    while curRetryNum > 0:
        if functionParaDict:
            doSuccess = functionCallback(**functionParaDict
        else:
            doSuccess = functionCallback()

        if doSuccess:
            break

        time.sleep(sleepInterval)
        curRetryNum -= 1

    if not doSuccess:
        if isShowErrWhenFail:
            functionName = str(functionCallback)
            # '<bound method DevicesMethods.switchToAppSto
            logging.error("Still fail after %d retry for %s
    return doSuccess
```

注：最新版本代码以及具体解释，详见：

通用逻辑 · Python常用代码段

isPageHasNaviBar_iOS 是否包含导航栏

```
def isPageHasNaviBar_iOS(self, page):
    """Check whether current page has XCUIElementTypeNaviga
    hasNaviBar = False
    naviBarName = ""
    """
        has:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>

            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>

            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>

            某个公众号：动卡空间
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>
        not:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>
    """
    soup = CommonUtils.xmlToSoup(page)
    foundNaviBar = soup.find(
        'XCUIElementTypeNavigationBar',
        attrs={"type": "XCUIElementTypeNavigationBar", "ena
    )
    if foundNaviBar:
        # maybeFakeNaviBarName = foundNaviBar.attrs["name"]
        maybeFakeNaviBarName = foundNaviBar.attrs.get("name
        typeOtherNameP = re.compile("%s,?" % maybeFakeNaviB
        foundTypeOther = foundNaviBar.find("XCUIElementType
        if foundTypeOther:
            hasNaviBar = True
            naviBarName = maybeFakeNaviBarName
```

```
    return hasNaviBar, naviBarName
```

调用：

```
OfflinePageNaviBarNameList = [
    "微信",
    "通讯录",
    "公众号",
]
hasNaviBar, naviBarName = self.isPageHasNaviBar_iOS(page)
```

获取页面源码getCurPageSource_iOS

```
def getCurPageSource_iOS(self, sourceFormat="xml"):
    """Get iOS current page source of xml/json

    Args:
        sourceFormat (str): source format: xml/json
    Returns:
        str
    Raises:
    """
    logging.info("start get iOS page source")
    pageSource = ""
    beforeGetTime = time.time()
    if sourceFormat == "xml":
        curPageXml = self.wdaClient.source() # format XML
        pageSource = curPageXml
    elif sourceFormat == "json":
        curPageJson = self.wdaClient.source(accessible=True

        pageSource = curPageJson
    else:
        logging.error("Unsupported source format: %s", sou
    afterGetTime = time.time()
    getSourceTime = afterGetTime - beforeGetTime
    logging.info("Cost %.2f seconds to get iOS page source"
    return pageSource
```

调用：

```
def getCurPageSource(self):
    if self.isAndroid:
        return self.getCurPageSource_Android()
    elif self.isiOS:
        return self.getCurPageSource_iOS()
```

search_iOS 点击弹出的键盘中的搜索按钮触发搜索

```
def search_iOS(self, wait=1):
    # 触发点击搜索按钮
    foundAndClickedDoSearch = False
    # <XCUIElementTypeButton type="XCUIElementTypeButton" r
    # <XCUIElementTypeButton type="XCUIElementTypeButton" r
    # searchButtonQuery = {"name": "Search"}
    searchButtonQuery = {"name": "Search", "type": "XCUIEle
    # Note: occasionally not found Search, change to find r
    MaxRetryNumber = 5
    curRetryNumber = MaxRetryNumber
    while curRetryNumber > 0:
        foundAndClickedDoSearch = self.findAndClickElement(
        if foundAndClickedDoSearch:
            break

        curRetryNumber -= 1

    if not foundAndClickedDoSearch:
        logging.error("Not found and/or clicked for %s", se
    return foundAndClickedDoSearch
```

调用举例：

```
isSearchOk = self.search_iOS(wait=0.2)
```

wait_element_setText_iOS 给元素输入文字，设置值

```
def wait_element_setText_iOS(self, query, text, isNeedClear
    """iOS set element text

    Args:
        query (dict): wda element query
        text (str): new text to set
        isNeedClear (bool): before set new text, is need c
    Returns:
    Raises:
    """
    isInputOk = False
    isFound, respInfo = self.findElement(query=query)
    logging.debug("isFound=%s, respInfo=%s", isFound, respI
    if isFound:
        curElement = respInfo
        if isNeedClear:
            # before set new value, clear current value
            self.iOSClearText(curElement)
        curElement.set_text(text)
        logging.info("has input text: %s", text)
        isInputOk = True
    return isInputOk
```

相关函数：

```
def iOSClearText(self, curElement):
    """iOS clear current element's text value
        Note: clear_text not working, so need use other wo

    Args:
        curElement (Element): wda element
    Returns:
    Raises:
    """
    # curElement.click()
    # curElement.clear_text()
    # curElement.tap_hold(2.0) # then try select All -> De
    backspaceChar = '\b'
    maxDeleteNum = 50
    curElement.set_text(maxDeleteNum * backspaceChar)
    return
```

调用举例：

（1）

对于xml

```
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="t
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
    <XCUIElementTypeTextField type="XCUIElementTypeTextFie
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
</XCUIElementTypeCell>
```

调用：

```
newUrlValue = newAutoProxyValue["url"]
parentCellClassChain = "/XCUIElementTypeCell[`rect.x = 0 AN
urlFieldQuery = {"type":"XCUIElementTypeTextField", "name"
urlFieldQuery["parent_class_chains"] = [ parentCellClassCha
# foundUrl, respInfo = self.findElement(urlFieldQuery, time
# if not foundUrl:
#     return False
isInputUrlOk = self.wait_element_setText_iOS(urlFieldQuery,
```

(2)

```
newAuthUserValue = newManualProxyValue["authUser"]
userFieldQuery = {"type":"XCUIElementTypeTextField", "name"
userFieldQuery["parent_class_chains"] = [ parentCellClassCh
isInputUserOk = self.wait_element_setText_iOS(userFieldQuer
```

wda

此处整理和 `facebook-wda` 本身相关的，常用的代码。

init_device_driver_iOS wda基本的初始化

```
def init_device_driver_iOS(self):
    # init facebook-wda
    # wda.HTTP_TIMEOUT = self.config["WdaHttpTimeout"]
    wda.HTTP_TIMEOUT = self.config["wda"]["http"]["timeout"
    # wda.DEBUG = self.config["WdaDebug"]
    wda.DEBUG = self.config["wda"]["debug"]

    # self.wdaClient = wda.Client('http://localhost:8100')
    # wdaServerUrl = 'http://%s:%s' % (self.config["WdaServ
    wdaServerUrl = 'http://%s:%s' % (self.config["wda"]["se
    logging.info("wdaServerUrl=%s", wdaServerUrl) # 'http:/
    self.wdaClient = wda.Client(wdaServerUrl)
    logging.info("self.wdaClient=%s", self.wdaClient)

    self.curSession = self.wdaClient

    # init settings
    # newSettings = {
    #     "snapshotTimeout": self.config["WdaSnapshotTimeou
    #     "screenshotQuality": self.config["WdaScreenQuali
    #     "mjpecfgScalingFactor": self.config["WdaScalingFa
    #     "includeNonModalElements": self.config["WdaInclud
    #     "shouldUseCompactResponses": self.config["WdaShou
    #     "elementResponseAttributes": self.config["WdaEler
    # }
    newSettings = {
        "snapshotTimeout": self.config["wda"]["settings"]["
        "screenshotQuality": self.config["wda"]["settings"]
        "mjpecfgScalingFactor": self.config["wda"]["setting
        "includeNonModalElements": self.config["wda"]["set
        "shouldUseCompactResponses": self.config["wda"]["se
        "elementResponseAttributes": self.config["wda"]["se

    }
    respNewSettings = self.curSession.set_settings(newSett
    logging.debug("respNewSettings=%s", respNewSettings)
```

processWdaResponse 处理(post等返回的response）返回数据

```
def processWdaResponse(self, wdaResponse):
    """Process common wda (http post) response

    Args:
    Returns:
        bool, ?/dict:
            true, response value
            false, error info dict
    Raises:
    """
    isRespOk = False
    respInfo = None
    logging.debug("wdaResponse=%s", wdaResponse)
    respStatus = wdaResponse.status
    respValue = wdaResponse.value
    respSessionId = wdaResponse.sessionId
    logging.debug("respStatus=%s, respValue=%s, respSessio
    if respStatus == 0:
        isRespOk = True
        respInfo = respValue
    else:
        isRespOk = False
        errInfo = {
            "status": respStatus,
            "value": respValue,
            "sessionId": respSessionId,
        }
        respInfo = errInfo
    return isRespOk, respInfo
```

调用举例：

(1)

```
curAppState = self.wdaClient.app_state(appBundleId)
isGetOk, respInfo = self.processWdaResponse(curAppState)
```

(2)

```
terminateResp = self.wdaClient.app_terminate(appBundleId)
logging.debug("terminateResp=%s", terminateResp)
# respStatus = resp.status
# respValue = resp.value
# respSessionId = resp.sessionId
# logging.info("respStatus=%s, respValue=%s, respSessionId'
# if respStatus == 0:
#     isTerminalOk = True
#     respInfo = None
# else:
#     errInfo = {
#         "status": respStatus,
#         "value": respValue,
#     }
#     respInfo = errInfo
isTerminalOk, respInfo = self.processWdaResponse(terminateF
```

(3)

```
launchResp = self.wdaClient.app_launch(appBundleId)
isLaunchOk, respInfo = self.processWdaResponse(launchResp)
```

crifan版wda

此处调试期间，已经基于 `v0.7.2` 的原版openatx的facebook-wda做了不少改动，以便于支持很多细节功能，和修复了一些bug。

有需要的，可以下载下面源码，然后替换已安装好的 `wda`

注：已安装好的wda的位置，此处是 `/Users/limao/.pyenv/versions/3.8.0/Python.framework/Versions/3.8/lib/python3.8/site-packages/wda/__init__.py` ，供参考。

crifan版wda源码

- 在线下载：
  - crifan版wda `__init__.py`
- 直接贴出源码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Updated: 20200609 17:01
# Editor: Crifan Li

from __future__ import print_function, unicode_literals

import base64
import copy
import functools
import io
import json
import os
import re
import threading
import time
from collections import defaultdict, namedtuple

import requests
import retry
import six
from six.moves import urllib
try:
    from functools import cached_property # Python3.8+
except ImportError:
    from cached_property import cached_property

from . import xcui_element_types

import logging
from enum import Enum

urlparse = urllib.parse.urlparse
_urljoin = urllib.parse.urljoin

if six.PY3:
    from functools import reduce

DEBUG = False
# to change to logging -> later can use colorful logging
print = logging.debug

HTTP_TIMEOUT = 60.0  # unit second

# RETRY_INTERVAL = 0.01
RETRY_INTERVAL = 0.1

LANDSCAPE = 'LANDSCAPE'
PORTRAIT = 'PORTRAIT'
LANDSCAPE_RIGHT = 'UIA_DEVICE_ORIENTATION_LANDSCAPERIGHT'
```

```
PORTRAIT_UPSIDEDOWN = 'UIA_DEVICE_ORIENTATION_PORTRAIT_UPSI

alert_callback = None

JSONDecodeError = json.decoder.JSONDecodeError if hasattr(
    json.decoder, "JSONDecodeError") else ValueError

##############################################################
# Definition
##############################################################

# https://developer.apple.com/documentation/uikit/uidevice/
class BatteryState(Enum):
    Unknown = 0
    Unplugged = 1
    Charging = 2
    Full = 3

# https://developer.apple.com/documentation/xctest/xctimage
class ScreenshotQuality(Enum):
    Original = 0 # lossless PNG image
    Medium = 1 # high quality lossy JPEG image
    Low = 2 # highly compressed lossy JPEG image

# https://developer.apple.com/documentation/xctest/xcuiapp
class ApplicationState(Enum):
    Unknown = 0
    NotRunning = 1
    RunningBackgroundSuspended = 2
    RunningBackground = 3
    RunningForeground = 4


class WDAError(Exception):
    """ base wda error """


class WDARequestError(WDAError):
    def __init__(self, status, value):
        self.status = status
        self.value = value

    def __str__(self):
        return 'WDARequestError(status=%d, value=%s)' % (se
                                                         se


class WDAEmptyResponseError(WDAError):
    """ response body is empty """
```

```
class WDAElementNotFoundError(WDAError):
    """ element not found """


class WDAElementNotDisappearError(WDAError):
    """ element not disappera """


##################################################################
# Utils Functions
##################################################################

def convert(dictionary):
    """
    Convert dict to namedtuple
    """
    return namedtuple('GenericDict', list(dictionary.keys()


def urljoin(*urls):
    """
    The default urlparse.urljoin behavior look strange
    Standard urlparse.urljoin('http://a.com/foo', '/bar')
    Expect: http://a.com/foo/bar
    Actually: http://a.com/bar

    This function fix that.
    """
    return reduce(_urljoin, [u.strip('/') + '/' for u in ur
                  '').rstrip('/')


def roundint(i):
    return int(round(i, 0))


def namedlock(name):
    """
    Returns:
        threading.Lock
    """
    if not hasattr(namedlock, 'locks'):
        namedlock.locks = defaultdict(threading.Lock)
    return namedlock.locks[name]


def httpdo(url, method="GET", data=None):
    """
    thread safe http request
    """
    p = urlparse(url)
```

```
    with namedlock(p.scheme + "://" + p.netloc):
        return _unsafe_httpdo(url, method, data)


def _unsafe_httpdo(url, method='GET', data=None):
    """
    Do HTTP Request
    """
    start = time.time()
    if DEBUG:
        body = json.dumps(data) if data else ''
        print("Shell: curl -X {method} -d '{body}' '{url}'"
            method=method.upper(), body=body or '', url=ur

    try:
        response = requests.request(method,
                                    url,
                                    json=data,
                                    timeout=HTTP_TIMEOUT)
    except (requests.exceptions.ConnectionError,
            requests.exceptions.ReadTimeout) as e:
        raise

    if DEBUG:
        ms = (time.time() - start) * 1000
        print('Return ({:.0f}ms): {}'.format(ms, response.t

    try:
        retjson = response.json()
        retjson['status'] = retjson.get('status', 0)
        r = convert(retjson)
        if r.status != 0:
            raise WDARequestError(r.status, r.value)
        if isinstance(r.value, dict) and r.value.get("erro
            raise WDARequestError(100, r.value['error']) #
        return r
    except JSONDecodeError:
        if response.text == "":
            raise WDAEmptyResponseError(method, url, data)
        raise WDAError(method, url, response.text)


##########################################################
# Main
##########################################################


class HTTPClient(object):
    def __init__(self, address, alert_callback=None):
        """
        Args:
```

```
            address (string): url address eg: http://localh
            alert_callback (func): function to call when a
        """
        self.address = address
        self.alert_callback = alert_callback

    def new_client(self, path):
        return HTTPClient(
            self.address.rstrip('/') + '/' + path.lstrip('/
            self.alert_callback)

    def fetch(self, method, url, data=None):
        return self._fetch_no_alert(method, url, data)
        # return httpdo(urljoin(self.address, url), method,

    def _fetch_no_alert(self, method, url, data=None, depth
        target_url = urljoin(self.address, url)
        try:
            return httpdo(target_url, method, data)
        except WDARequestError as err:
            if depth >= 10:
                raise
            if err.status != 26:  # alert status code
                raise
            if not callable(self.alert_callback):
                raise
            self.alert_callback()
            return self._fetch_no_alert(method, url, data,

    def __getattr__(self, key):
        """ Handle GET,POST,DELETE, etc ... """
        return functools.partial(self.fetch, key)


class Rect(object):
    def __init__(self, x, y, width, height):
        self.x = x
        self.y = y
        self.width = width
        self.height = height

    def __str__(self):
        return 'Rect(x={x}, y={y}, width={w}, height={h})'.
            x=self.x, y=self.y, w=self.width, h=self.height

    def __repr__(self):
        return str(self)

    # @property
    @cached_property
    def center(self):
```

```
            if DEBUG:
                print("calc center position")
            return namedtuple('Point', ['x', 'y'])(self.x + se
                                                   self.y + se

        @property
        def origin(self):
            return namedtuple('Point', ['x', 'y'])(self.x, self

        @property
        def left(self):
            return self.x

        @property
        def top(self):
            return self.y

        @property
        def right(self):
            return self.x + self.width

        @property
        def bottom(self):
            return self.y + self.height

        @property
        def x0(self):
            return self.x

        @property
        def y0(self):
            return self.y

        @property
        def x1(self):
            return self.x + self.width

        @property
        def y1(self):
            return self.y + self.height

        @property
        def centerX(self):
            return self.center[0]
            # return self.center["x"]

        @property
        def centerY(self):
            return self.center[1]
            # return self.center["y"]
```

```
    @property
    def isZero(self):
        isXZero = self.x == 0
        isYZero = self.y == 0
        isWidthZero = self.width == 0
        isHeightZero = self.height == 0
        isAllZero = isXZero and isYZero and isWidthZero and
        return isAllZero


class Client(object):
    def __init__(self, url=None, _session_id=None):
        """
        Args:
            target (string): the device url

        If target is empty, device url will set to env-var
        """
        if not url:
            url = os.environ.get('DEVICE_URL', 'http://loca
        assert re.match(r"^https?://", url), "Invalid URL:

        self.http = HTTPClient(url)

        # Session variable
        self.__session_id = _session_id
        self.__timeout = 30.0
        self.__target = None

    def wait_ready(self, timeout=120):
        """
        wait until WDA back to normal

        Returns:
            bool (if wda works)
        """
        deadline = time.time() + timeout
        while time.time() < deadline:
            try:
                self.status()
                return True
            except:
                time.sleep(2)
        return False

    @retry.retry(exceptions=WDAEmptyResponseError, tries=3,
    def status(self):
        res = self.http.get('status')
        sid = res.sessionId
        res.value['sessionId'] = sid
        return res.value
```

```
    def home(self):
        """Press home button"""
        return self.http.post('/wda/homescreen')

    def healthcheck(self):
        """Hit healthcheck"""
        return self.http.get('/wda/healthcheck')

    def locked(self):
        """ returns locked status, true or false """
        return self.http.get("/wda/locked").value

    def lock(self):
        return self.http.post('/wda/lock')

    def unlock(self):
        """ unlock screen, double press home """
        return self.http.post('/wda/unlock')

    def app_current(self):
        """
        Returns:
            dict, eg:
            {
                "running" : true,
                "state" : 4,
                "generation" : 0,
                "processArguments" : {
                    "env" : {},
                    "args" : []
                },
                "title" : "",
                "bundleId" : "com.tencent.xin",
                "label" : "微信",
                "path" : "",
                "name" : "",
                "pid" : 1357
            }
        """
        return self.http.get("/wda/activeAppInfo").value

    def source(self, format='xml', accessible=False):
        """
        Args:
            format (str): only 'xml' and 'json' source type
            accessible (bool): when set to true, format is
        """
        if accessible:
            return self.http.get('/wda/accessibleSource').v
        return self.http.get('source?format=' + format).va
```

```
    def screenshot(self, png_filename=None, format='pillow
        """
        Screenshot with PNG format

        Args:
            png_filename(string): optional, save file name
            format(string): return format, "raw" or "pillow

        Returns:
            PIL.Image or raw png data

        Raises:
            WDARequestError
        """
        value = self.http.get('screenshot').value
        raw_value = base64.b64decode(value)
        png_header = b"\x89PNG\r\n\x1a\n"
        if not raw_value.startswith(png_header) and png_fi
            raise WDARequestError(-1, "screenshot png forma

        if png_filename:
            with open(png_filename, 'wb') as f:
                f.write(raw_value)

        if format == 'raw':
            return raw_value
        elif format == 'pillow':
            from PIL import Image
            buff = io.BytesIO(raw_value)
            return Image.open(buff)
        else:
            raise ValueError("unknown format")

    def session(self,
                bundle_id=None,
                arguments=None,
                environment=None,
                alert_action=None):
        """
        Args:
            - bundle_id (str): the app bundle id
            - arguments (list): ['-u', 'https://www.google.
            - enviroment (dict): {"KEY": "VAL"}
            - alert_action (str): "accept" or "dismiss"

        WDA Return json like

        {
            "value": {
                "sessionId": "69E6FDBA-8D59-4349-B7DE-A9CA4
```

```
                "capabilities": {
                    "device": "iphone",
                    "browserName": "部落冲突",
                    "sdkVersion": "9.3.2",
                    "CFBundleIdentifier": "com.supercell.ma
                }
            },
            "sessionId": "69E6FDBA-8D59-4349-B7DE-A9CA41A97
            "status": 0
        }

        To create a new session, send json data like

        {
            "desiredCapabilities": {
                "bundleId": "your-bundle-id",
                "app": "your-app-path"
                "shouldUseCompactResponses": (bool),
                "shouldUseTestManagerForVisibilityDetectio
                "maxTypingFrequency": (integer),
                "arguments": (list(str)),
                "environment": (dict: str->str)
            },
        }
        """
        if bundle_id is None:
            return self

        if arguments and type(arguments) is not list:
            raise TypeError('arguments must be a list')

        if environment and type(environment) is not dict:
            raise TypeError('environment must be a dict')

        capabilities = {
            'bundleId': bundle_id,
            'arguments': arguments,
            'environment': environment,
            'shouldWaitForQuiescence': False,
            # In the latest appium/WebDriverAgent, set shou
            # 'useJSONSource': True,
            # 'simpleIsVisibleCheck': True,
            # 'shouldUseTestManagerForVisibilityDetection'
        }
        # Remove empty value to prevent WDARequestError
        for k in list(capabilities.keys()):
            if capabilities[k] is None:
                capabilities.pop(k)

        if alert_action:
            assert alert_action in ["accept", "dismiss"]
```

```
            capabilities["defaultAlertAction"] = alert_acti

        data = {
            'desiredCapabilities': capabilities,  # For old
            "capabilities": {
                "alwaysMatch": capabilities,  # For recent
            }
        }
        if DEBUG:
            print("data=%s" % data)
        try:
            res = self.http.post('session', data)
            # if DEBUG:
            #     # print("res=%s" % res)
            #     # respJson = res.json()
            #     # print("respJson=%s" % respJson)
            #     respDict = dict(res)
            #     print("respDict=%s" % respDict)
        except WDAEmptyResponseError:
            """ when there is alert, might be got empty res
            use /wda/apps/state may still get sessionId
            """
            res = self.session().app_state(bundle_id)
            if res.value != 4:
                raise
        return Client(self.http.address, _session_id=res.se

    #@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#
    ######  Session methods and properties ######
    #@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#
    def __str__(self):
        # return 'wda.Client (sessionId=%s)' % self._sid
        return 'wda.Client (sessionId=%s)' % self.__sessior

    def __enter__(self):
        """
        Usage example:
            with c.session("com.example.app") as app:
                # do something
        """
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        self.close()

    @property
    def id(self):
        return self._session_id

    @property
    def _session_id(self) -> str:
```

```
        if self.__session_id:
            return self.__session_id
        return self.status()['sessionId']

    @property
    def _session_http(self) -> HTTPClient:
        return self.http.new_client("session/"+self._sessio

    @cached_property
    def scale(self):
        """
        UIKit scale factor

        Refs:
            https://developer.apple.com/library/archive/doc
        There is another way to get scale
            self.http.get("/wda/screen").value returns {"st
        """
        v = max(self.screenshot().size) / max(self.window_s
        if DEBUG:
            print("v=%s" % v)
        return round(v)

    @cached_property
    def bundle_id(self):
        """ the session matched bundle id """
        v = self._session_http.get("/").value
        return v['capabilities'].get('CFBundleIdentifier')

    def implicitly_wait(self, seconds):
        """
        set default element search timeout
        """
        assert isinstance(seconds, (int, float))
        self.__timeout = seconds

    def battery_info(self):
        """
        Returns dict:
            eg: {"level": 1, "state": 2}
            level: 0 ~ 1.0, battery electricity percent
            state: unknown=0, unplugged=1, charging=2, full
                https://developer.apple.com/documentation/u
        """
        return self._session_http.get("/wda/batteryInfo").v

    def matchTouchID(self):
        postResp = self._session_http.post("/wda/touch_id",
            "match": 1,
        })
        respValue = postResp.value
```

```
        if DEBUG:
            print("/wda/touch_id: respValue=%s" % respValue
        return respValue

    def device_info(self):
        """
        Returns dict:
            eg: {'currentLocale': 'zh_CN', 'timeZone': 'As
        """
        return self._session_http.get("/wda/device/info").v

    def screen_info(self):
        """get screen info
            https://developer.apple.com/library/archive/doc

        Returns dict:
            eg:
                {'statusBarSize': {'width': 375, 'height':
                -> updated wda code, can return more info:
                {'statusBarSize': {'width': 375, 'height':
        """
        if DEBUG:
            print("self._session_http=%s" % self._session_h

        return self._session_http.get("/wda/screen").value
        # return self.http.get("/wda/screen").value

    def window_info(self):
        """get window size info

        Returns dict:
            eg: {'width': 375, 'height': 667}
        """
        return self._session_http.get("/window/size").value
        # return self.http.get("/window/size").value

    def get_settings(self):
        resp = self._session_http.get("/appium/settings")
        respSettings = resp.value
        if DEBUG:
            # print("resp=%s" % resp) #  TypeError not all
            print("respSettings=%s" % respSettings)
        return respSettings

    def set_settings(self, newSettings):
        postResp = self._session_http.post("/appium/setting
            "settings": newSettings,
        })
        respNewSettings = postResp.value
        if DEBUG:
            print("respNewSettings=%s" % respNewSettings)
```

```
        return respNewSettings

    def setSnapshotTimeout(self, newSnapshotTimeout):
        """set new snapshotTimeout value for current sessio
        newSettings = {
            "snapshotTimeout": newSnapshotTimeout
        }
        return self.set_settings(newSettings)
    def set_clipboard(self, content, content_type="plainte
        """ set clipboard """
        self._session_http.post(
            "/wda/setPasteboard", {
                "content": base64.b64encode(content.encode
                "contentType": content_type
            })

    def set_alert_callback(self, callback):
        """
        Args:
            callback (func): called when alert popup

        Example of callback:

            def callback(session):
                session.alert.accept()
        """
        if callable(callback):
            self.http.alert_callback = functools.partial(ca
        else:
            self.http.alert_callback = None

    #Not working
    #def get_clipboard(self):
    #   self.http.post("/wda/getPasteboard").value

    # Not working
    #def siri_activate(self, text):
    #    self.http.post("/wda/siri/activate", {"text": text

    def app_launch(self,
                   bundle_id,
                   arguments=[],
                   environment={},
                   wait_for_quiescence=False):
        """
        Args:
            - bundle_id (str): the app bundle id
            - arguments (list): ['-u', 'https://www.google.
            - enviroment (dict): {"KEY": "VAL"}
            - wait_for_quiescence (bool): default False
        """
```

```
        assert isinstance(arguments, (tuple, list))
        assert isinstance(environment, dict)

        return self._session_http.post(
            "/wda/apps/launch", {
                "bundleId": bundle_id,
                "arguments": arguments,
                "environment": environment,
                "shouldWaitForQuiescence": wait_for_quiesce
            })

    def app_activate(self, bundle_id):
        return self._session_http.post("/wda/apps/launch",
            "bundleId": bundle_id,
        })

    def app_terminate(self, bundle_id):
        return self._session_http.post("/wda/apps/terminate
            "bundleId": bundle_id,
        })

    def app_state(self, bundle_id):
        """
        Returns example:
            {
                "value": 4,
                "sessionId": "0363BDC5-4335-47ED-A54E-F7CCB
            }

            value: enum ApplicationState
        """
        return self._session_http.post("/wda/apps/state",
            "bundleId": bundle_id,
        })

    def app_list(self):
        """
        Not working very well, only show springboard

        Returns:
            list of app

        Return example:
            [{'pid': 52, 'bundleId': 'com.apple.springboard
        """
        return self._session_http.get("/wda/apps/list").va

    def open_url(self, url):
        """
        TODO: Never successed using before. Looks like use
        https://github.com/facebook/WebDriverAgent/blob/mas
```

```
        Args:
            url (str): url

        Raises:
            WDARequestError
        """
        return self._session_http.post('url', {'url': url}

    def deactivate(self, duration):
        """Put app into background and than put it back
        Args:
            - duration (float): deactivate time, seconds
        """
        return self._session_http.post('/wda/deactivateApp

    def tap(self, x, y):
        return self._session_http.post('/wda/tap/0', dict(x

    def _percent2pos(self, x, y, window_size=None):
        if DEBUG:
            print("x=%s, y=%s, window_size=%s" % (x, y, wi

        if any(isinstance(v, float) for v in [x, y]):
            w, h = window_size or self.window_size()
            if DEBUG:
                print("type(w)=%s" % type(w))
                print("type(h)=%s" % type(h))
                print("w=%s, h=%s" % (w, h))

            # x = int(x * w) if isinstance(x, float) else x
            # y = int(y * h) if isinstance(y, float) else y
            if isinstance(x, float):
                if (x > 0.0) and (x <= 1.0):
                    convertedX = x * w
                    xInt = int(convertedX)
                else:
                    # consider as real float position value
                    xInt = int(x)
                x = xInt

            if isinstance(y, float):
                if (y > 0.0) and (y <= 1.0):
                    convertedY = y * h
                    yInt = int(convertedY)
                else:
                    # consider as real float position value
                    yInt = int(y)
                y = yInt

            if DEBUG:
                print("type(x)=%s" % type(x))
```

```
                print("type(y)=%s" % type(y))
                print("x=%s, y=%s" % (x, y))

            assert w >= x >= 0
            assert h >= y >= 0
        return (x, y)

    def click(self, x, y):
        """
        x, y can be float(percent) or int
        """
        x, y = self._percent2pos(x, y)
        return self.tap(x, y)

    def double_tap(self, x, y):
        x, y = self._percent2pos(x, y)
        return self._session_http.post('/wda/doubleTap', d

    def tap_hold(self, x, y, duration=1.0):
        """
        Tap and hold for a moment

        Args:
            - x, y(int, float): float(percent) or int(abso
            - duration(float): seconds of hold time

        [[FBRoute POST:@"/wda/touchAndHold"] respondWithTa
        """
        x, y = self._percent2pos(x, y)
        data = {'x': x, 'y': y, 'duration': duration}
        return self._session_http.post('/wda/touchAndHold'

    def swipe(self, x1, y1, x2, y2, duration=0):
        """
        Args:
            x1, y1, x2, y2 (int, float): float(percent), i
            duration (float): start coordinate press durat

        [[FBRoute POST:@"/wda/dragfromtoforduration"] respo
        """
        if any(isinstance(v, float) for v in [x1, y1, x2, y
            size = self.window_size()
            x1, y1 = self._percent2pos(x1, y1, size)
            x2, y2 = self._percent2pos(x2, y2, size)

        data = dict(fromX=x1, fromY=y1, toX=x2, toY=y2, du
        return self._session_http.post('/wda/dragfromtofor

    def swipe_left(self):
        w, h = self.window_size()
        return self.swipe(w, h // 2, 0, h // 2)
```

```
    def swipe_right(self):
        w, h = self.window_size()
        return self.swipe(0, h // 2, w, h // 2)

    def swipe_up(self):
        w, h = self.window_size()
        return self.swipe(w // 2, h, w // 2, 0)

    def swipe_down(self):
        w, h = self.window_size()
        return self.swipe(w // 2, 0, w // 2, h)

    @property
    def orientation(self):
        """
        Return string
        One of <PORTRAIT | LANDSCAPE>
        """
        return self._session_http.get('orientation').value

    @orientation.setter
    def orientation(self, value):
        """
        Args:
            - orientation(string): LANDSCAPE | PORTRAIT | U
                    UIA_DEVICE_ORIENTATION_PORTRAIT_UPSIDEI
        """
        return self._session_http.post('orientation', data=

    def window_size(self):
        """
        Returns:
            namedtuple: eg
                Size(width=320, height=568)
        """
        value = self._session_http.get('/window/size').valu
        w = roundint(value['width'])
        h = roundint(value['height'])
        return namedtuple('Size', ['width', 'height'])(w, h

    def send_keys(self, value):
        """
        send keys, yet I know not, todo function
        """
        if isinstance(value, six.string_types):
            value = list(value)
        return self._session_http.post('/wda/keys', data={

    def keyboard_dismiss(self):
        """
```

```
        Not working for now
        """
        raise RuntimeError("not pass tests, this method is
        self._session_http.post('/wda/keyboard/dismiss')

    def xpath(self, value):
        """
        For weditor, d.xpath(...)
        """
        httpclient = self._session_http.new_client('')
        return Selector(httpclient, self, xpath=value)

    @property
    def alert(self):
        return Alert(self)

    def close(self): # close session
        return self._session_http.delete('/')

    def __call__(self, *args, **kwargs):
        if 'timeout' not in kwargs:
            kwargs['timeout'] = self.__timeout
        return Selector(self._session_http, self, *args, **

    @property
    def alibaba(self):
        """ Only used in alibaba company """
        try:
            import wda_taobao
            return wda_taobao.Alibaba(self)
        except ImportError:
            raise RuntimeError(
                "@alibaba property requires wda_taobao libi

    @property
    def taobao(self):
        try:
            import wda_taobao
            return wda_taobao.Taobao(self)
        except ImportError:
            raise RuntimeError(
                "@taobao property requires wda_taobao libra


Session = Client # for compability

class Alert(object):
    def __init__(self, client):
        self._c = client
        self.http = client.http
```

```
    @property
    def exists(self):
        try:
            self.text
        except WDARequestError as e:
            if e.status != 27:
                raise
            return False
        return True

    @property
    def text(self):
        return self.http.get('/alert/text').value
        # return self._c._session_http.get('/alert/text').v

    def wait(self, timeout=20.0):
        start_time = time.time()
        while time.time() - start_time < timeout:
            if self.exists:
                return True
            time.sleep(0.2)
        return False

    def accept(self):
        return self.http.post('/alert/accept')

    def dismiss(self):
        return self.http.post('/alert/dismiss')

    def buttons(self):
        return self._c._session_http.get('/wda/alert/buttor
        # return self.http.get('/wda/alert/buttons').value

    def click(self, button_name):
        """
        Args:
            - button_name: the name of the button
        """
        # Actually, It has no difference POST to accept or
        return self.http.post('/alert/accept', data={"name'

class Selector(object):
    def __init__(self,
                 httpclient,
                 session,
                 predicate=None,
                 id=None,
                 className=None,
                 type=None,
                 name=None,
                 nameContains=None,
```

```
                nameMatches=None,
                text=None,
                textContains=None,
                textMatches=None,
                value=None,
                valueContains=None,
                valueMatches=None,
                label=None,
                labelContains=None,
                visible=None,
                enabled=None,

                # https://github.com/facebookarchive/WebDr
                #  rect - Element's rectangle as a diction
                rect=None,
                x=None,
                y=None,
                width=None,
                height=None,

                classChain=None,
                xpath=None,
                parent_class_chains=[],
                timeout=10.0,
                index=0):
        '''
        Args:
            predicate (str): predicate string
            id (str): raw identifier
            className (str): attr of className
            type (str): alias of className
            name (str): attr for name
            nameContains (str): attr of name contains
            nameMatches (str): regex string
            text (str): alias of name
            textContains (str): alias of nameContains
            textMatches (str): alias of nameMatches
            value (str): attr of value, not used in most ti
            valueContains (str): attr of value contains
            valueMatches (str): attr of value regex string
            label (str): attr for label
            labelContains (str): attr for label contains
            visible (bool): is visible
            enabled (bool): is enabled
            rect (dict):  Element's rectangle as a dictiona
            x (int/str):  x of rect
            y (int/str):  y of rect
            width (int/str):  width of rect
            height (int/str):  height of rect
            classChain (str): string of ios chain query, eg
            xpath (str): xpath string, a little slow, but v
```

```
            timeout (float): maxium wait element time, defa
            index (int): index of founded elements

        WDA use two key to find elements "using", "value"
        Examples:
        "using" can be on of
            "partial link text", "link text"
            "name", "id", "accessibility id"
            "class name", "class chain", "xpath", "predicat

        predicate string support many keys
            UID,
            accessibilityContainer,
            accessible,
            enabled,
            frame,
            label,
            name,
            rect,
            type,
            value,
            visible,
            wdAccessibilityContainer,
            wdAccessible,
            wdEnabled,
            wdFrame,
            wdLabel,
            wdName,
            wdRect,
            wdType,
            wdUID,
            wdValue,
            wdVisible
        '''
        if DEBUG:
            print("Selector __init__: httpclient=%s, sessio
            print("Selector __init__: className=%s, type=%s
            print("Selector __init__: text=%s, textContains
            print("Selector __init__: label=%s, labelConta
            print("Selector __init__: rect=%s, x=%s, y=%s,
            print("Selector __init__: classChain=%s, xpath=

        self.http = httpclient
        self.session = session

        self.predicate = predicate
        self.id = id
        self.class_name = className or type
        self.name = self._add_escape_character_for_quote_pr
            name or text)
        self.name_part = nameContains or textContains
```

```
        self.name_regex = nameMatches or textMatches
        self.value = value
        self.value_part = valueContains
        self.value_regex = valueMatches
        self.label = label
        self.label_part = labelContains

        # self.enabled = enabled
        # self.visible = visible
        # self.enabled = self._toBool(enabled)
        # self.visible = self._toBool(visible)
        if enabled is not None:
            self.enabled = self._toBool(enabled)
        else:
            self.enabled = None

        if visible is not None:
            self.visible = self._toBool(visible)
        else:
            self.visible = None

        if rect:
            self.rect = rect
        else:
            self.rect = None

            rectDict = {}

            if x:
                self.x = int(x)
                rectDict["x"] = self.x
            if y:
                self.y = int(y)
                rectDict["y"] = self.y
            if width:
                self.width = int(width)
                rectDict["width"] = self.width
            if height:
                self.height = int(height)
                rectDict["height"] = self.height

            if rectDict:
                self.rect = rectDict

        self.index = index

        self.xpath = self._fix_xcui_type(xpath)
        self.class_chain = self._fix_xcui_type(classChain)
        self.timeout = timeout
        # some fixtures
        if self.class_name and not self.class_name.startsw
```

```
                'XCUIElementType'):
            self.class_name = 'XCUIElementType' + self.clas
        if self.name_regex:
            if not self.name_regex.startswith(
                    '^') and not self.name_regex.startswitl
                self.name_regex = '.*' + self.name_regex
            if not self.name_regex.endswith(
                    '$') and not self.name_regex.endswith(
                self.name_regex = self.name_regex + '.*'
            if DEBUG:
                print("after update: self.name_regex=%s" %
        self.parent_class_chains = parent_class_chains

    def _toBool(self, inputValue):
        respBool = False
        if isinstance(inputValue, bool):
            respBool = inputValue
        elif isinstance(inputValue, str):
            lowerStr = inputValue.lower()
            TrueStrList = ["true", "yes", "1"]
            if lowerStr in TrueStrList:
                respBool = True
        elif isinstance(inputValue, int):
            if inputValue >= 1:
                respBool = True

        if DEBUG:
            print("inputValue=%s -> respBool=%s" % (inputVa

        return respBool

    def _fix_xcui_type(self, s):
        if s is None:
            return
        re_element = '|'.join(xcui_element_types.ELEMENTS)
        return re.sub(r'/(' + re_element + ')', '/XCUIEleme

    def _add_escape_character_for_quote_prime_character(se
        """
        Fix for https://github.com/openatx/facebook-wda/iss
        Returns:
            string with properly formated quotes, or non ch
        """
        if text is not None:
            if "'" in text:
                return text.replace("'", "\\'")
            elif '"' in text:
                return text.replace('"', '\\"')
            else:
                return text
        else:
```

```
            return text

    def _wdasearch_single(self, using, value):
        """
        Returns:
            bool, str
                True, element info:
                    {
                        "id": "0F000000-0000-0000-D703-0000
                        "name": "通讯录",
                        "value": None,
                        "text": "通讯录",
                        "label": "通讯录",
                        "rect": {
                            "y": 619,
                            "x": 96,
                            "width": 90,
                            "height": 48
                        },
                        "type": "XCUIElementTypeButton"
                    }

                False, error info:
                    {
                        "error" : "no such element",
                        "message" : "unable to find an eler
                        "traceback" : "(\n\t0   WebDriverAg
                    }
        """
        isFound, respInfo = False, "Unknown error"

        postDict = {
            'using': using,
            'value': value
        }

        if DEBUG:
            print("postDict=%s" % postDict)

        isException = False
        # resp = self.http.post('/element', postDict)
        try:
            resp = self.http.post('/element', postDict)
        except WDARequestError as wdaReqErr:
            isException = True
            if DEBUG:
                print("wdaReqErr=%s" % wdaReqErr)

            errCode = wdaReqErr.status
            errMsg = wdaReqErr.value
            errorInfo = {
```

```
                "error": errCode,
                "message": errMsg
            }
            respInfo = errorInfo

        if not isException:
            # if DEBUG:
            #     print("postDict=%s -> resp=%s" % (postDi

            respValue = resp.value
            # if DEBUG:
            #     print("respValue=%s" % respValue)

            respValueKeys = respValue.keys()
            hasError = "error" in respValueKeys
            hasMessage = "message" in respValueKeys
            isError = hasError and hasMessage

            if isError:
                """
                {
                    "value" : {
                        "error" : "no such element",
                        "message" : "unable to find an ele
                        "traceback" : "(\n\t0   WebDriverAg
                    },
                    "sessionId" : "DCF1A843-9FB9-4415-9A7C-
                }
                """
                isFound = False
                errorInfo = respValue
                respInfo = errorInfo
            else:
                isFound = True
                elementId = respValue['ELEMENT']
                # if DEBUG:
                #     print("elementId=%s" % elementId)

                """
                {
                    "value" : {
                        "text" : null,
                        "element-6066-11e4-a52e-4f735466ce
                        "label" : null,
                        "attribute\/name" : null,
                        "attribute\/value" : null,
                        "ELEMENT" : "00000000-0000-0000-00
                    },
                    "sessionId" : "E649E60F-16A1-48D4-8840-
                }
                """
```

```
                    EmptyId = "00000000-0000-0000-0000-0000000
                    if elementId == EmptyId:
                        isFound = False
                        respInfo = {
                            "error" : "empty id",
                            "message" : "found element but id
                        }
                    else:
                        """
                        {
                            "value" : {
                                "element-6066-11e4-a52e-4f7354
                                "attribute/name" : "添加",
                                "attribute/value" : null,
                                "text" : "添加",
                                "label" : "添加",
                                "rect" : {
                                    "y" : 20,
                                    "x" : 317,
                                    "width" : 58,
                                    "height" : 44
                                },
                                "type" : "XCUIElementTypeButton
                                "name" : "XCUIElementTypeButton
                                "ELEMENT" : "19010000-0000-0000
                            },
                            "sessionId" : "DCF1A843-9FB9-4415-9
                        }
                        """
                        elementInfo = {
                            "id": elementId
                        }
                        # respKeyList = ["type","label","name",
                        # mapKeyList =  ["type","label",    "",
                        respKeyMap = {
                            "type": "type",
                            "label": "label",
                            "name": None,
                            "text": "text",
                            "rect": "rect",
                            "attribute/name": "name",
                            "attribute/value": "value",
                        }

                        for eachKey in respValueKeys:
                            if eachKey in respKeyMap.keys():
                                eachValue = respValue[eachKey]
                                mapRealKey = respKeyMap[eachKey
                                if mapRealKey:
                                    elementInfo[mapRealKey] = e
```

```
                    respInfo = elementInfo

        if DEBUG:
            print("isFound=%s, respInfo=%s" % (isFound, res

        # return elementId
        return isFound, respInfo

    def _wdasearch(self, using, value):
        """
        Returns:
            element_ids (list(string)): example ['id1', 'id

        HTTP example response:
        [
            {"ELEMENT": "E2FF5B2A-DBDF-4E67-9179-91609480D8
            {"ELEMENT": "597B1A1E-70B9-4CBE-ACAD-40943B0A60
        ]
        """
        if DEBUG:
            print("using=%s, value=%s" % (using, value))

        element_ids = []
        # for v in self.http.post('/elements', {
        #         'using': using,
        #         'value': value
        # }).value:
        #     element_ids.append(v['ELEMENT'])
        resp = self.http.post('/elements', {
                'using': using,
                'value': value
        })
        valueList = resp.value

        if DEBUG:
            print("valueList=%s" % valueList)

        for eachValue in valueList:
            eachElement = eachValue['ELEMENT']
            element_ids.append(eachElement)

        if DEBUG:
            print("element_ids=%s" % element_ids)

        return element_ids

    def _gen_class_chain(self):
        # just return if aleady exists predicate
        if self.predicate:
            return '/XCUIElementTypeAny[`' + self.predicate
```

```
        qs = []
        if self.name:
            qs.append("name == '%s'" % self.name)
        if self.name_part:
            qs.append("name CONTAINS '%s'" % self.name_part
        if self.name_regex:
            # escapedNameRegex = self.name_regex.encode('ur
            # if DEBUG:
            #     print("self.name_regex=%s, escapedNameReg
            # qs.append("name MATCHES '%s'" % escapedNameRe
            qs.append("name MATCHES '%s'" % self.name_regex
            if DEBUG:
                print("after add name_regex: qs=%s" % qs)
        if self.label:
            qs.append("label == '%s'" % self.label)
        if self.label_part:
            qs.append("label CONTAINS '%s'" % self.label_pa
        if self.value:
            qs.append("value == '%s'" % self.value)
        if self.value_part:
            # qs.append("value CONTAINS '%s'" % self.value_
            qs.append("value CONTAINS '%s'" % self.value_pa
        if self.value_regex:
            # escapedValueRegex = self.value_regex.encode(
            # if DEBUG:
            #     print("self.value_regex=%s, escapedValueR
            # qs.append("value MATCHES '%s'" % escapedValue
            qs.append("value MATCHES '%s'" % self.value_reg
            if DEBUG:
                print("after add value_regex: qs=%s" % qs)

        # BoolTrueValue = "true"
        # BoolFalseValue = "false"
        BoolTrueValue = "1"
        BoolFalseValue = "0"

        if self.enabled is not None:
            enabledValue = BoolTrueValue if self.enabled e
            enableKey = "enabled"
            # enableKey = "wdEnabled"
            qs.append("%s == %s" % (enableKey, enabledValue

        # 20200427: after merge latest WebDriverAgent, cont
        if DEBUG:
            print("self.visible=%s" % self.visible)
        if self.visible is not None:
            visibleKey = "visible"
            # Note: WebDriverAgent says support visible, bu
            #   so temp change to (maybe similar meaning) a
            # 20200402: sometime use visible or accessible
            # visibleKey = "accessible"
```

```
            # 20200408: debug displayed
            # visibleKey = "displayed"
            visibleValue =  BoolTrueValue if self.visible 
            visibleConditionStr = "%s == %s" % (visibleKey,
            qs.append(visibleConditionStr)
            if DEBUG:
                print("visibleConditionStr=%s" % visibleCon
                print("qs=%s" % qs)

        if self.rect:
            rectKeyList = ["x", "y", "width", "height"]
            for eachRectKey in rectKeyList:
                if eachRectKey in self.rect.keys():
                    eachRectValue = self.rect[eachRectKey]
                    curMatchRule = "rect.%s == %s" % (eachR
                    qs.append(curMatchRule)

        predicate = ' AND '.join(qs)
        chain = '/' + (self.class_name or 'XCUIElementTypeA

        if predicate:
            chain = chain + '[`' + predicate + '`]'
        if self.index:
            chain = chain + '[%d]' % self.index

        if DEBUG:
            print("generated class chain=%s" % chain)

        return chain

    def find_element_ids(self):
        elems = []
        if self.id:
            return self._wdasearch('id', self.id)
        if self.predicate:
            return self._wdasearch('predicate string', self
        if self.xpath:
            return self._wdasearch('xpath', self.xpath)
        if self.class_chain:
            return self._wdasearch('class chain', self.clas

        chain = '**' + ''.join(
            self.parent_class_chains) + self._gen_class_cha
        if DEBUG:
            print('CHAIN: %s', chain)
        return self._wdasearch('class chain', chain)

    def find_element_id(self):
        if self.id:
            return self._wdasearch_single('id', self.id)
```

```
        if self.predicate:
            if DEBUG:
                print('PREDICATE: %s', self.predicate)
            return self._wdasearch_single('predicate string

        if self.xpath:
            if DEBUG:
                print('XPATH: %s', self.xpath)
            return self._wdasearch_single('xpath', self.xpa

        if self.class_chain:
            if DEBUG:
                print('CLASS_CHAIN: %s', self.class_chain)
            return self._wdasearch_single('class chain', se

        chain = '**' + ''.join(
            self.parent_class_chains) + self._gen_class_cha
        if DEBUG:
            print('CHAIN: %s', chain)
        return self._wdasearch_single('class chain', chain)

    def find_element(self):
        """
        Returns:
            True, Element: single element
            False, error info
        """
        # element = None
        # elementId = self.find_element_id()
        isFound, respInfo = self.find_element_id()

        if DEBUG:
            # print('elementId=%s' % elementId)
            print('isFound=%s, respInfo=%s' % (isFound, res

        if isFound:
            elementInfo = respInfo
            elementId = elementInfo["id"]

            # add bounds support
            curBounds = None
            if "rect" in elementInfo:
                rectDict = elementInfo["rect"]
                x = rectDict["x"]
                y = rectDict["y"]
                width = rectDict["width"]
                height = rectDict["height"]
                curBounds = Rect(x, y, width, height)
            if DEBUG:
                print("curBounds=%s" % curBounds)
```

```
            # element = Element(self.http.new_client(''), e
            element = Element(self.http.new_client(''), ele

            if DEBUG:
                print('element=%s' % element)

            respInfo = element

        return isFound, respInfo

    def find_elements(self):
        """
        Returns:
            Element (list): all the elements
        """
        es = []
        for element_id in self.find_element_ids():
            if DEBUG:
                print('find_elements: element_id=%s' % elem
            ele = Element(self.http.new_client(''), element
            if DEBUG:
                print('find_elements: ele=%s' % ele)
            es.append(ele)
        return es

    def count(self):
        if DEBUG:
            print("count")

        elementIdList = self.find_element_ids()
        elementIdNum = len(elementIdList)

        if DEBUG:
            print("count: elementIdList=%s" % elementIdLis

        return elementIdNum

    # def get(self, timeout=None, raise_error=True, isRespS
    # def get(self, timeout=None, raise_error=True, isRespS
    def get(self, timeout=None):
        """
        Args:
            timeout (float): timeout for query element, un
                Default 10s

        Returns:
            True, Element
            False, error info

        Raises:
        """
```

```
            isFound, respInfo = False, "Unknown error"

            if DEBUG:
                # print("get: timeout=%s, raise_error=%s, isRes
                print("get: timeout=%s" % timeout)

            start_time = time.time()
            if timeout is None:
                timeout = self.timeout

            if DEBUG:
                print("get: timeout=%s" % timeout)

            endTime = start_time + timeout
            while True:
                # if isRespSingle:
                #     singleElement = self.find_element()
                #     if singleElement:
                #         return singleElement
                # else:
                #     elems = self.find_elements()
                #     if len(elems) > 0:
                #         return elems[0]
                isFound, respInfo = self.find_element()
                if isFound:
                    return isFound, respInfo

                curTime = time.time()
                if endTime < curTime:
                    break

                # time.sleep(0.01)
                time.sleep(RETRY_INTERVAL)

                if DEBUG:
                    from datetime import datetime
                    curDatetime = datetime.fromtimestamp(float(
                    print("curDatetime=%s -> Not timeout, cont

            # # check alert again
            # sessionAlertExists = self.session.alert.exists
            # httpAlertCallback = self.http.alert_callback

            # if DEBUG:
            #     print("get: sessionAlertExists=%s, httpAlert(

            # if sessionAlertExists and httpAlertCallback:
            #     self.http.alert_callback()

            #     if DEBUG:
            #         print("get: timeout=%s, raise_error=%s" %
```

```
        #     # return self.get(timeout, raise_error)
        #     return self.get(timeout)

        # if raise_error:
        #     raise WDAElementNotFoundError("element not fo
        return isFound, respInfo

    def __getattr__(self, oper):
        if DEBUG:
            print("oper=%s" % oper)

        # return getattr(self.get(), oper)
        isFound, respInfo = self.get()

        if DEBUG:
            print("isFound=%s, respInfo=%s" % (isFound, res

        if isFound:
            element = respInfo
            return getattr(element, oper)
        else:
            return None

    def set_timeout(self, s):
        """
        Set element wait timeout
        """
        self.timeout = s
        return self

    def __getitem__(self, index):
        self.index = index
        return self

    def child(self, *args, **kwargs):
        chain = self._gen_class_chain()
        kwargs['parent_class_chains'] = self.parent_class_c
        return Selector(self.http, self.session, *args, **k

    @property
    def exists(self):
        return len(self.find_element_ids()) > self.index

    def click_exists(self, timeout=0):
        """
        Wait element and perform click

        Args:
            timeout (float): timeout for wait
```

```
        Returns:
            bool: if successfully clicked
        """
        # e = self.get(timeout=timeout, raise_error=False)
        # if e is None:
        #     return False
        isFound, respInfo = self.get(timeout=timeout)
        if isFound:
            e = respInfo

            e.click()
            return True
        else:
            return False

    # def wait(self, timeout=None, raise_error=True):
    def wait(self, timeout=None):
        """ alias of get
        Args:
            timeout (float): timeout seconds

        Raises:
        """
        # return self.get(timeout=timeout, raise_error=rais
        return self.get(timeout=timeout)

    def wait_gone(self, timeout=None, raise_error=True):
        """
        Args:
            timeout (float): default timeout
            raise_error (bool): return bool or raise error

        Returns:
            bool: works when raise_error is False

        Raises:
            WDAElementNotDisappearError
        """
        start_time = time.time()
        if timeout is None or timeout <= 0:
            timeout = self.timeout
        while start_time + timeout > time.time():
            if not self.exists:
                return True
        if not raise_error:
            return False
        raise WDAElementNotDisappearError("element not gone

    # todo
    # pinch
    # touchAndHold
```

```
    # dragfromtoforduration
    # twoFingerTap

    # todo
    # handleGetIsAccessibilityContainer
    # [[FBRoute GET:@"/wda/element/:uuid/accessibilityConta


class Element(object):
    # def __init__(self, httpclient, id):
    # def __init__(self, httpclient, id, rect=None):
    def __init__(self, httpclient, id, bounds=None):
        """
        base_url eg: http://localhost:8100/session/$SESSION
        """
        self.http = httpclient
        self._id = id
        # add cached rect=bounds, to avoid later get rect h
        if DEBUG:
            print("bounds=%s" % bounds)
        if bounds:
            self.bounds = bounds
            if DEBUG:
                print("use pass in bounds=%s" % self.bounds

    def __repr__(self):
        return '<wda.Element(id="{}")>'.format(self.id)

    def _req(self, method, url, data=None):
        return self.http.fetch(method, '/element/' + self.

    def _wda_req(self, method, url, data=None):
        return self.http.fetch(method, '/wda/element/' + se

    def _prop(self, key):
        return self._req('get', '/' + key.lstrip('/')).valu

    def _wda_prop(self, key):
        ret = self._request('GET', 'wda/element/%s/%s' % (s
        return ret

    # @property
    @cached_property
    def id(self):
        return self._id

    # @property
    @cached_property
    def label(self):
        return self._prop('attribute/label')
```

```
    # @property
    @cached_property
    def className(self):
        return self._prop('attribute/type')

    # @property
    @cached_property
    def text(self):
        return self._prop('text')

    # @property
    @cached_property
    def name(self):
        # return self._prop('name')
        # Note: here property name, internally: FBElementC
        # so, for:
        # <XCUIElementTypeButton type="XCUIElementTypeButt
        # name is XCUIElementTypeButton in type="XCUIElemer
        # not expected: ¥1.00 in name="¥1.00"
        # so change to 'attribute/name'
        return self._prop('attribute/name') # ¥1.00

    # @property
    @cached_property
    def displayed(self):
        return self._prop("displayed")

    # @property
    @cached_property
    def enabled(self):
        return self._prop('enabled')

    # @property
    @cached_property
    def accessible(self):
        return self._wda_prop("accessible")
        # return self._prop("accessible")

    # @property
    @cached_property
    def accessibility_container(self):
        return self._wda_prop('accessibilityContainer')
        # return self._prop('accessibilityContainer')

    @property
    # Special: not used cache value for AppStore downloadi
    #   <XCUIElementTypeButton type="XCUIElementTypeButton'
    #   if use cache, later always get None
    # @cached_property
    def value(self):
        curValue = self._prop('attribute/value')
```

```
        # curValue = self._prop('attribute/wdValue')
        if DEBUG:
            print("curValue=%s" % curValue)
        return curValue

    # @property
    @cached_property
    def enabled(self):
        return self._prop('enabled')

    # @property
    @cached_property
    def visible(self):
        # if DEBUG:
        #     print("call attribute/visible to get visible
        return self._prop('attribute/visible')

    # @property
    @cached_property
    def bounds(self):
        if DEBUG:
            print("try get Element bounds=rect")

        value = self._prop('rect')
        x, y = value['x'], value['y']
        w, h = value['width'], value['height']
        rectObj = Rect(x, y, w, h)
        if DEBUG:
            print("rectObj=%s" % rectObj)
        return rectObj

    # operations
    def tap(self):
        return self._req('post', '/click')

    def click(self):
        """ Alias of tap """
        return self.tap()

    def tap_hold(self, duration=1.0):
        """
        Tap and hold for a moment

        Args:
            duration (float): seconds of hold time

        [[FBRoute POST:@"/wda/element/:uuid/touchAndHold"]
        """
        return self._wda_req('post', '/touchAndHold', {'du

    def scroll(self, direction='visible', distance=1.0):
```

```
        """
        Args:
            direction (str): one of "visible", "up", "down"
            distance (float): swipe distance, only works wh

        Raises:
            ValueError

        distance=1.0 means, element (width or height) mult
        """
        if direction == 'visible':
            self._wda_req('post', '/scroll', {'toVisible':
        elif direction in ['up', 'down', 'left', 'right']:
            self._wda_req('post', '/scroll', {
                'direction': direction,
                'distance': distance
            })
        else:
            raise ValueError("Invalid direction")
        return self

    def pinch(self, scale, velocity):
        """
        Args:
            scale (float): scale must > 0
            velocity (float): velocity must be less than ze

        Example:
            pinchIn  -> scale:0.5, velocity: -1
            pinchOut -> scale:2.0, velocity: 1
        """
        data = {'scale': scale, 'velocity': velocity}
        return self._wda_req('post', '/pinch', data)

    def set_text(self, value):
        return self._req('post', '/value', {'value': value}

    def clear_text(self):
        return self._req('post', '/clear')

    # def child(self, **kwargs):
    #     return Selector(self.__base_url, self._id, **kwar

    # todo lot of other operations
    # tap_hold
```

元素处理

此处整理用wda查找元素、点击元素等常见代码段。

getiOSElementQuery 生成wda的元素的query

```
def getiOSElementQuery(self, locator):
    """from element locator to generate iOS element query"'
    query = None
    # if isinstance(locator, list):
    #     logging.error("TODO: add support in future for lo
    # elif isinstance(locator, dict):
    # {'name': '公众号'}
    # query = {}
    # query = locator
    query = copy.deepcopy(locator)

    # Note: not auto convert key for compatible old Androic
    # {'text': '通讯录'}, {'desc': '添加'}
    # locatorText = locator.get("text")
    # locatorDesc = locator.get("desc")
    # if locatorText:
    #     query = {"name": locatorText}
    # elif locatorDesc:
    #     query = {"label": locatorDesc}
    # else:
    #     logging.warning("TODO: add support later for %s",

    # auto add type for iOS
    hasTag = "tag" in query.keys()
    noType = "type" not in query.keys()
    if hasTag and noType:
        query["type"] = query["tag"]
        del query["tag"]

    # 20200402: facebook-wda+WebDriverAgent internally
    #     (1) not support visible
    #     (2) sometime support accessible
    # -> now, not use visible anymore
    # 20200430: updated latest WebDriverAgent source code,
    #     so above not need delete visible
    # 20200430: but after add visible, still not find (some
    if "visible" in query.keys():
        del query["visible"]

    logging.debug("locator=%s -> query=%s", locator, query)
    return query
```

说明：

对于输入的query的dict，进行一定处理：

1. 去掉visible
    - 截止20200615，WebDriverAgent内部还是不能完美支持元素的visible值的判断

- -》有时候支持 有时候不支持，导致传入visible=true，却找不到元素
- -》所以去掉visible

2. 把tag改为type

调用举例：

```
def isFoundElement_iOS(self, locator):
    isFound = False
    foundElement = None
    query = self.getiOSElementQuery(locator)
    if query:
        isFound, foundElement = self.findElement(query=quer
    return isFound, foundElement
```

findElement 查找元素

```
# def findElement(self, query={}, timeout=WaitFind):
# def findElement(self, query={}, timeout=WaitFind, isRetry
def findElement(self, query={}, timeout=WaitFind, isRetryNo
    """Find element from query

    Args:
        query (dict): query condition
        timeout(float): max wait time
        isRetryNoYWhenFail(bool): True to do retry but remo
    Returns:
        bool, Element/str
            True, Element
            False, error message
    Raises:
    """
    logging.debug("query=%s", query)
    isFound, respInfo = False, "Unkown error"

    elementSelector = self.curSession(**query, timeout=time
    logging.debug("elementSelector=%s", elementSelector)
    isFound, respInfo = elementSelector.get()
    logging.debug("query=%s -> isFound=%s, respInfo=%s", qu
    if not isFound:
        if isRetryNoYWhenFail and ("y" in query):
            # Note:
            # for sometime / many cases:
            #     not found element due to y position is no
            #     for these cases, retry to find without y
            noYQuery = copy.deepcopy(query)
            del noYQuery["y"]
            elementSelectorNoY = self.curSession(**noYQuery
            logging.debug("elementSelectorNoY=%s", element
            isFound, respInfo = elementSelectorNoY.get()
            logging.debug("retry no y: noYQuery=%s -> isFou
            # somtime here found out element but y is negat
            # <XCUIElementTypeImage type="XCUIElementTypeIm
            # actually is not visible -> not our expected -
            if isFound:
                curRect = respInfo.bounds
                logging.debug("curRect=%s", curRect)
                isValidY = curRect.y >= 0
                isVisible = respInfo.visible
                logging.debug("isVisible=%s", isVisible)
                isReallyFound = isValidY and isVisible
                logging.debug("isReallyFound=%s", isReallyF
                if not isReallyFound:
                    isFound = False
                    respInfo = respInfo

    return isFound, respInfo
```

调用举例：

（1）简单的

（2）复杂的，带parent的class chain：设置 WiFi列表页 中查找 无线局域网

对于xml

```
<XCUIElementTypeNavigationBar type="XCUIElementTypeNavigat
    <XCUIElementTypeButton type="XCUIElementTypeButton" nar
    <XCUIElementTypeOther type="XCUIElementTypeOther" name=
</XCUIElementTypeNavigationBar>
```

写法是：

```
wifiName = "无线局域网"
parentNaviBarClassChain = "/XCUIElementTypeNavigationBar[`n
wifiQuery = {"type":"XCUIElementTypeOther", "name": wifiNar
wifiQuery["parent_class_chains"] = [ parentNaviBarClassCha
isFoundWifi, respInfo = self.findElement(query=wifiQuery, t
```

和：

```
<XCUIElementTypeTabBar type="XCUIElementTypeTabBar" enable
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
        <XCUIElementTypeOther type="XCUIElementTypeOther" e
            <XCUIElementTypeOther type="XCUIElementTypeOthe
            <XCUIElementTypeOther type="XCUIElementTypeOthe
            <XCUIElementTypeOther type="XCUIElementTypeOthe
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    <XCUIElementTypeButton type="XCUIElementTypeButton" va
    <XCUIElementTypeButton type="XCUIElementTypeButton" nar
    <XCUIElementTypeButton type="XCUIElementTypeButton" va
    <XCUIElementTypeButton type="XCUIElementTypeButton" nar
</XCUIElementTypeTabBar>
```

代码 判断iOS中页面是否处于 微信：

```
weixinTabQuery = {"type":"XCUIElementTypeButton", "name": "
weixinTabQuery["parent_class_chains"] = [ tabBarClassChain
isFoundWeixin, respInfo = self.findElement(query=weixinTabQ
iOSisInWeixin = isFoundWeixin
return iOSisInWeixin
```

（3）复杂的，name中包含的

```
<XCUIElementTypeStaticText type="XCUIElementTypeStaticText"
```

写法：

```
DropDownReturnDetailStr = "下拉返回商品详情"
dropDownReturnDetailQuery = {"type":"XCUIElementTypeStatic"
isFound, foundElement = self.findElement(dropDownReturnDeta
```

- AppStore 详情页 不折叠输入法 带金额的购买按钮 ¥1.00：

```
<XCUIElementTypeCollectionView type="XCUIElementTypeCollect
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
    <XCUIElementTypeCell type="XCUIElementTypeCell" enable
        <XCUIElementTypeImage type="XCUIElementTypeImage" v
        <XCUIElementTypeStaticText type="XCUIElementTypeSta
        <XCUIElementTypeStaticText type="XCUIElementTypeSta
        <XCUIElementTypeStaticText type="XCUIElementTypeSta
        <XCUIElementTypeButton type="XCUIElementTypeButton"
        <XCUIElementTypeStaticText type="XCUIElementTypeSta
        <XCUIElementTypeButton type="XCUIElementTypeButton"
        <XCUIElementTypeOther type="XCUIElementTypeOther" n
    </XCUIElementTypeCell>
```

查找写法是：

```
parentCellClassChain = "/XCUIElementTypeCell[`rect.x = 0 AN
moneyButtonQuery = {"type":"XCUIElementTypeButton", "nameC
moneyButtonQuery["parent_class_chains"] = [ parentCellClass
foundMoneyButton, respInfo = self.findElement(query=moneyBu
```

（4）复杂的，name符合条件的

场景：寻找 微信公众号列表页中 上下滚动后的 顶部浮动的大写字母横条的元素

```
<XCUIElementTypeOther type="XCUIElementTypeOther" name="D"
    <XCUIElementTypeOther type="XCUIElementTypeOther" name=
</XCUIElementTypeOther>

<XCUIElementTypeOther type="XCUIElementTypeOther" name="E"
    <XCUIElementTypeStaticText type="XCUIElementTypeStatic
</XCUIElementTypeOther>

<XCUIElementTypeOther type="XCUIElementTypeOther" name="J"
    <XCUIElementTypeStaticText type="XCUIElementTypeStatic
</XCUIElementTypeOther>
```

代码：

```
floatUpperLetterElement = None
# floatUpperLetterQuery = {"type":"XCUIElementTypeStaticTe
# isFound, foundElement = findElement(curSession, floatUpp
floatUpperLetterQuery = {"type":"XCUIElementTypeOther", "n
isFound, foundElement = self.findElement(floatUpperLetterQ
```

（5）复杂的，value包含的

```
curNoticeQuery = {"type":"XCUIElementTypeStaticText", "valu
curNoticeQuery["parent_class_chains"] = [scrollViewClassCh
isFoundCurNotice, respInfo = self.findElement(query=curNot
```

findAndClickElement 查找并点击元素

```
# def findAndClickElement(self, query={}, timeout=WaitFind,
# def findAndClickElement(self, query={}, timeout=WaitFind,
# def findAndClickElement(self, query={}, timeout=WaitFind,
def findAndClickElement(self, query={}, timeout=WaitFind, :
    """Find and click element

    Args:
        query (dict): query parameter for find element
        timeout (float): max timeout seconds for find eleme
        enableDebug (bool): if enable debug, then draw clic
        isShowErrLog(bool): True to show error log, False t
    Returns:
        bool
    Raises:
    """
    foundAnClicked = False

    # isFound, respInfo = self.findElement(query=query, tir
    # isFound, respInfo = self.findElement(query=query, tir
    isFound, respInfo = self.findElement(query=query, timec
    logging.debug("isFound=%s, respInfo=%s", isFound, respI
    if isFound:
        curElement = respInfo

        # if enableDebug:
        #     # for debug
        #     curScreenFile = debugSaveScreenshot(curScale=
        #     utils.imageDrawRectangle(curScreenFile, rectI

        clickOk = self.clickElement(curElement)
        logging.info("%s to Clicked element %s", clickOk, c

        foundAnClicked = clickOk
    else:
        if isShowErrLog:
            logging.error("Not found element %s", query)
        else:
            logging.debug("Not found element %s", query)

    return foundAnClicked
```

调用举例：

（1）

```
# <XCUIElementTypeButton type="XCUIElementTypeButton" name=
# <XCUIElementTypeButton type="XCUIElementTypeButton" name=
# searchButtonQuery = {"name": "Search"}
searchButtonQuery = {"name": "Search", "type": "XCUIElement

foundAndClickedDoSearch = self.findAndClickElement(searchBu
```

（2）场景：设置 WiFi 配置代理 从手动切换到 关闭 存储

xml:

```
<XCUIElementTypeNavigationBar type="XCUIElementTypeNavigat
    <XCUIElementTypeButton type="XCUIElementTypeButton" nam
    <XCUIElementTypeOther type="XCUIElementTypeOther" name=
    <XCUIElementTypeButton type="XCUIElementTypeButton" nam
</XCUIElementTypeNavigationBar>
```

代码：

```
storeName = "存储"
parentNaviBarClassChain = "/XCUIElementTypeNavigationBar[`n
storeButtonQuery = {"type":"XCUIElementTypeButton", "name"
storeButtonQuery["parent_class_chains"] = [ parentNaviBarC
foundAndClickedStore = self.findAndClickElement(query=store
```

（3）isShowErrLog=False 当找不到，不要报错

场景：

AppStore 详情页 淘宝 弹框 安装

xml

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
    <XCUIElementTypeTable type="XCUIElementTypeTable" enab
        <XCUIElementTypeCell type="XCUIElementTypeCell" ena
            <XCUIElementTypeOther type="XCUIElementTypeOthe
            <XCUIElementTypeImage type="XCUIElementTypeImag
            <XCUIElementTypeStaticText type="XCUIElementTyp
        </XCUIElementTypeCell>
        <XCUIElementTypeCell type="XCUIElementTypeCell" ena
            <XCUIElementTypeStaticText type="XCUIElementTyp
            <XCUIElementTypeStaticText type="XCUIElementTyp
        </XCUIElementTypeCell>
    </XCUIElementTypeTable>
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
        <XCUIElementTypeOther type="XCUIElementTypeOther" e
        <XCUIElementTypeOther type="XCUIElementTypeOther" e
        <XCUIElementTypeButton type="XCUIElementTypeButton"
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
```

代码：

```
parentOtherClassChain = "/XCUIElementTypeOther[`rect.x = 0
installButtonQuery = {"type":"XCUIElementTypeButton", "name
installButtonQuery["parent_class_chains"] = [ parentOtherC
foundAndClickedInstall = self.findAndClickElement(query=ins
```

findAndClickButtonElementBySoup

```
    def findAndClickButtonElementBySoup(self, curButtonSoup
        """
            iOS的bug：根据bs找到了soup元素（往往是一个button）后
                实际上点击的是别的位置，别的元素
            为了规避此bug，所以去：
                通过soup，再去找button的wda的元素，然后根据元素去
                    则都是可以正常点击，不会有误点击的问题
        """
        # # change to wda element query then click by eleme
        # if not curButtonName:
        #     curSoupAttrs = curButtonSoup.attrs
        #     curButtonName = curSoupAttrs["name"]
        # # rights close white
        # # login close
        # curButtonQuery = {"type":"XCUIElementTypeButton",
        # # foundAndClicked = self.findAndClickElement(curB

        curButtonQuery = {"type":"XCUIElementTypeButton", '
        extraQuery = {}

        # change to wda element query then click by element
        if curButtonName:
            extraQuery["name"] = curButtonName
        else:
            if curButtonSoup:
                curSoupAttrs = curButtonSoup.attrs
                if hasattr(curSoupAttrs, "name"):
                    curButtonName = curSoupAttrs["name"]
                    # rights close white
                    # login close
                    extraQuery["name"] = curButtonName
                else:
                    # no name attribute, use position
                    x = curSoupAttrs["x"]
                    y = curSoupAttrs["y"]
                    width = curSoupAttrs["width"]
                    height = curSoupAttrs["height"]
                    extraQuery["x"] = x
                    extraQuery["y"] = y
                    extraQuery["width"] = width
                    extraQuery["height"] = height
                    # {'enabled': 'true', 'height': '32',

        # merge query
        # curButtonQuery = {**curButtonQuery, **extraQuery}
        curButtonQuery.update(extraQuery) # {'enabled': 'tr

        foundAndClicked = self.findAndClickElement(curButto
        return foundAndClicked
```

不同的地方的各种调用：

```
foundAndProcessedPopup = self.findAndClickButtonElementBySc

foundAndProcessedPopup = self.findAndClickButtonElementBySc

foundAndProcessedPopup = self.findAndClickButtonElementBySc

foundAndProcessedPopup = self.findAndClickButtonElementBySc

foundAndProcessedPopup = self.findAndClickButtonElementBySc

foundAndProcessedPopup = self.findAndClickButtonElementBySc
```

详见：

【已解决】自动抓包iOS的app：优化clickElementCenterPosition点击失效时换用wda寻找元素并点击逻辑

findAndClickCenterPosition 查找并点击元素中间位置

```
def findAndClickCenterPosition(self, bsChainList, soup=None
    """use Beautifulsoup chain list to find soup node then

    Args:
        bsChainList (list): dict list for dict of tag and a
        soup (Soup)): BeautifulSoup soup
        isUseWdaQueryAndClick (bool): for special node bs c
    Returns:
        bool: found and cliked or not
    Raises:
    """
    foundAndClicked = False
    if not soup:
        curPageXml = self.get_page_source()
        soup = CommonUtils.xmlToSoup(curPageXml)
    foundSoup = CommonUtils.bsChainFind(soup, bsChainList)
    if foundSoup:
        if isUseWdaQueryAndClick:
            foundAndClicked = self.findAndClickButtonElemen
        else:
            self.clickElementCenterPosition(foundSoup)
            foundAndClicked = True

    return foundAndClicked
```

场景：米家 弹框 立即体验 xml

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="
    <XCUIElementTypeOther type="XCUIElementTypeOther" enab
        <XCUIElementTypeOther type="XCUIElementTypeOther" e
            <XCUIElementTypeImage type="XCUIElementTypeImag
            <XCUIElementTypeStaticText type="XCUIElementTyp
            <XCUIElementTypeStaticText type="XCUIElementTyp
            <XCUIElementTypeButton type="XCUIElementTypeBut
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
```

调用举例：

```
immediatelyExperienceChainList = [
    {
        "tag": "XCUIElementTypeOther",
        "attrs": self.FullScreenAttrDict
    },
    {
        "tag": "XCUIElementTypeOther",
        "attrs": self.FullScreenAttrDict
    },
    {
        "tag": "XCUIElementTypeButton",
        "attrs": {"enabled":"true", "visible":"true", "name
    },
]
foundAndProcessedPopup = self.findAndClickCenterPosition(in
return foundAndProcessedPopup
```

findAndClickButtonElementBySoup 通过soup去用wda的query查找元素并点击

```
def findAndClickButtonElementBySoup(self, curButtonSoup=Nor
    """
        iOS的bug：根据bs找到了soup元素（往往是一个button）后，用
            实际上点击的是别的位置，别的元素
        为了规避此bug，所以去：
            通过soup，再去找button的wda的元素，然后根据元素去点击
                则都是可以正常点击，不会有误点击的问题
    """
    # # change to wda element query then click by element
    # if not curButtonName:
    #     curSoupAttrs = curButtonSoup.attrs
    #     curButtonName = curSoupAttrs["name"]
    # # rights close white
    # # login close
    # curButtonQuery = {"type":"XCUIElementTypeButton", "er
    # # foundAndClicked = self.findAndClickElement(curButto

    curButtonQuery = {"type":"XCUIElementTypeButton", "enab
    extraQuery = {}

    # change to wda element query then click by element
    if curButtonName:
        extraQuery["name"] = curButtonName
    else:
        if curButtonSoup:
            curSoupAttrs = curButtonSoup.attrs
            if hasattr(curSoupAttrs, "name"):
                curButtonName = curSoupAttrs["name"]
                # rights close white
                # login close
                extraQuery["name"] = curButtonName
            else:
                # no name attribute, use position
                x = curSoupAttrs["x"]
                y = curSoupAttrs["y"]
                width = curSoupAttrs["width"]
                height = curSoupAttrs["height"]
                extraQuery["x"] = x
                extraQuery["y"] = y
                extraQuery["width"] = width
                extraQuery["height"] = height
                # {'enabled': 'true', 'height': '32', 'type

    # merge query
    # curButtonQuery = {**curButtonQuery, **extraQuery}
    curButtonQuery.update(extraQuery) # {'enabled': 'true',

    foundAndClicked = self.findAndClickElement(curButtonQue
    return foundAndClicked
```

说明：

- iOS的bug：根据bs找到了soup元素（往往是一个button）后，用clickCenterPosition=clickElementCenterPosition 去点击中间坐标，往往会有问题
  - 实际上点击的是别的位置，别的元素
- 为了规避此bug，所以去：
  - 通过soup，再去找button的wda的元素，然后根据元素去点击
    - 则都是可以正常点击，不会有误点击的问题

调用举例：

```
commonCloseSoup = CommonUtils.bsChainFind(soup, commonClose
if commonCloseSoup:
    # self.clickElementCenterPosition(commonCloseSoup) # sc
    # foundAndProcessedPopup = True

    # so change to wda query element then click
    foundAndProcessedPopup = self.findAndClickButtonElement
```

clickElementCenterPosition 点击元素（通过属性中找到元素坐标，点击中间位置）

```
def clickElementCenterPosition(self, curElement):
    """Click center position of element

    Args:
        curElement (Element): Beautiful soup / lxml element
    Returns:
        bool
    Raises:
    """
    hasClicked = False
    # centerPos = None
    centerX = None
    centerY = None

    hasBounds = hasattr(curElement, "bounds")
    curBounds = None
    if hasBounds:
        curBounds = curElement.bounds

    if hasBounds and curBounds:
        # wda element
        if hasattr(curBounds, "center"):
            # is wda Rect
            curRect = curBounds
            rectCenter = curRect.center
            centerX = rectCenter[0]
            centerY = rectCenter[1]
    else:
        attrDict = None
        if hasattr(curElement, "attrs"):
            # Beautiful soup node
            attrDict = curElement.attrs
        elif hasattr(curElement, "attrib"):
            # lxml element
            attrDict = dict(curElement.attrib)

        if attrDict:
            logging.info("attrDict=%s", attrDict)
            hasCoordinate = ("x" in attrDict) and ("y" in a
            if hasCoordinate:
                x = int(attrDict["x"])
                y = int(attrDict["y"])
                width = int(attrDict["width"])
                height = int(attrDict["height"])
                centerX = x + int(width / 2)
                centerY = y + int(height / 2)

    if centerX and centerY:
        centerPos = (centerX, centerY)
        self.tap(centerPos)
```

```
            logging.info("Clicked center position: %s", centerP
            hasClicked = True

        return hasClicked
```

调用举例：

```
confirmSoup = parentOtherSoup.find(
    "XCUIElementTypeButton",
    attrs={"visible":"true", "enabled":"true", "name": "确定
)
if confirmSoup:
    self.clickElementCenterPosition(confirmSoup)
    foundAndProcessedPopup = True
```

iOS设备

此处整理和iOS设备，比如iPhone相关的常用代码段。

设备是否解锁

```
def is_device_unlock_iOS(self, device):
    # Note: before later real init dirver, init here for he
    self.init_device_driver_iOS()
    isLocked = self.wdaClient.locked()
    isUnlocked = not isLocked
    return isUnlocked
```

注：其中 `init_device_driver_iOS` 定义详见：wda基本的初始化

获取电池状态

```
def is_device_charged_iOS(self):
    batteryInfo = self.wdaClient.battery_info()
    logging.debug("batteryInfo=%s", batteryInfo)
    # batteryInfo={'level': 1, 'state': 2}
    # batteryInfo={'level': 0.49000000953674316, 'state': 2

    batteryLevelFloat = batteryInfo["level"]
    batteryLevelPercentInt = int(batteryLevelFloat * 100) #
    batteryStateInt = batteryInfo["state"]
    logging.debug("batteryStateInt=%s", batteryStateInt)

    curBattryState = BatteryState(batteryStateInt)
    logging.debug("curBattryState=%s", curBattryState) # cu
    curBattryStateName = curBattryState.name
    logging.debug("curBattryStateName=%s", curBattryStateNa
    logging.info("Battery state: %s", curBattryStateName) #
    return batteryLevelPercentInt
```

相关定义：

```
from wda import ScreenshotQuality, BatteryState, Applicatio
```

其中是我自己定义的：

```
##############################################################
# Definition
##############################################################

# https://developer.apple.com/documentation/uikit/uidevice/
class BatteryState(Enum):
    Unknown = 0
    Unplugged = 1
    Charging = 2
    Full = 3

# https://developer.apple.com/documentation/xctest/xctimage
class ScreenshotQuality(Enum):
    Original = 0 # lossless PNG image
    Medium = 1 # high quality lossy JPEG image
    Low = 2 # highly compressed lossy JPEG image

# https://developer.apple.com/documentation/xctest/xcuiapp
class ApplicationState(Enum):
    Unknown = 0
    NotRunning = 1
    RunningBackgroundSuspended = 2
    RunningBackground = 3
    RunningForeground = 4
```

get_devices_iOS 当前连接（到Mac）的iOS设备

```
def get_devices_iOS(self):
    deviceStrList = self.get_iOS_deviceStrList()
    iOSDevIdList = []
    for eachDevStr in deviceStrList:
        # Mo iPhone (11.4.1) [fc9e1f9f2e1339328b9580f826a30
        # iPhone 11 Pro Max (13.3) + Apple Watch Series 5 -
        # iPhone 11 (13.3) [509BC7C7-9C0E-42FA-8AB2-F5220EB
        foundDevId = re.search("\[(?P<iOSdevId>[\w\-]+)\](
        if foundDevId:
            iOSdevId = foundDevId.group("iOSdevId")
            iOSDevIdList.append(iOSdevId)
    # ['fc9e1f9f2e1339328b9580f826a30fded3bc69ff', 'D86F0BD
    return iOSDevIdList
```

相关函数：

```
def get_iOS_deviceStrList(self):
    deviceStrList = []
    deviceListCmd = 'instruments -s devices'
    deviceListStr = CommonUtils.get_cmd_lines(deviceListCmd
    if deviceListStr:
        deviceRawList = deviceListStr.split("\n")
        """
            Known Devices:
            limao的MacBook Pro [F9089371-1060-5CE3-99BB-817
            Mo iPhone (11.4.1) [fc9e1f9f2e1339328b9580f826a
            Apple TV (13.3) [6680F059-4DE1-430C-B696-228AC2
            Apple TV 4K (13.3) [048E58E8-6A27-4D81-BDEB-881
            Apple TV 4K (at 1080p) (13.3) [384D5E60-B6B1-48
            Apple Watch Series 4 - 40mm (6.1.1) [1B98415B-3
            Apple Watch Series 4 - 44mm (6.1.1) [661838E9-B
            iPad (7th generation) (13.3) [7F8EDE89-74E0-4BA
            iPad Air (3rd generation) (13.3) [BBC48526-3922
            iPad Pro (11-inch) (13.3) [04DD3B8A-5B78-48E8-8
            iPad Pro (12.9-inch) (3rd generation) (13.3) [D
            iPad Pro (9.7-inch) (13.3) [B11D5D40-FEA2-4114-
            iPhone 11 (13.3) [509BC7C7-9C0E-42FA-8AB2-F522C
            iPhone 11 Pro (13.3) [3E8E7E92-66F2-4AF3-A405-2
            iPhone 11 Pro (13.3) + Apple Watch Series 5 - 4
            iPhone 11 Pro Max (13.3) [50C15135-1532-44C5-B8
            iPhone 11 Pro Max (13.3) + Apple Watch Series 5
            iPhone 8 (13.3) [54589698-0C9F-407D-B21A-83432C
            iPhone 8 Plus (13.3) [509B7103-97DB-4AB9-B829-C
        """
        # CoreData: annotation:  Failed to load optimized m
        # InvalidCoreData = "CoreData"
        # Known Devices:
        # InvalidKnownDevices = "Known Devices"
        for eachDeviceStr in deviceRawList:
            eachDeviceStr = eachDeviceStr.strip()
            # isNotCoreData = InvalidCoreData not in eachDe
            # isNotKnownDevices = InvalidKnownDevices not i
            # if isNotCoreData and isNotKnownDevices:
            # if isNotKnownDevices:
            # Note: current only support iPhone devices, bc
            #     with device name contain 'iphone'
            foundiPhone = re.search("iPhone", eachDeviceStr
            if foundiPhone:
                deviceStrList.append(eachDeviceStr)
                """
                    Mo iPhone (11.4.1) [fc9e1f9f2e1339328b9
                    iPhone 11 (13.3) [509BC7C7-9C0E-42FA-8A
                    iPhone 11 Pro (13.3) [3E8E7E92-66F2-4AF
                    iPhone 11 Pro (13.3) + Apple Watch Seri
                    iPhone 11 Pro Max (13.3) [50C15135-1532
                    iPhone 11 Pro Max (13.3) + Apple Watch
```

```
                    iPhone 8 (13.3) [54589698-0C9F-407D-B2
                    iPhone 8 Plus (13.3) [509B7103-97DB-4A
                """
    return deviceStrList
```

调用举例：

```
def get_phone_name_iOS(self):
    deviceName = ""
    deviceStrList = self.get_iOS_deviceStrList()
    # Mo iPhone (11.4.1) [fc9e1f9f2e1339328b9580f826a30fde
    curDevP = "(?P<deviceName>[\w ]+)\s+\((?P<iOSVersion>[
    for eachDeviceStr in deviceStrList:
        foundCurDev = re.search(curDevP, eachDeviceStr)
        if foundCurDev:
            deviceName = foundCurDev.group("deviceName")
            iOSVersion = foundCurDev.group("iOSVersion")
            break

    return deviceName
```

app

此处整理和app相关的常用代码段。

app管理

说明：下面app管理期间用到的iOS的内置的app，其bundle id可参考：
iOS内置app的bundle id

get_PackageActivity_iOS 当前正在运行的app的信息

```
def get_PackageActivity_iOS(self):
    """
        {
            "running" : true,
            "state" : 4,
            "generation" : 0,
            "processArguments" : {
                "env" : {},
                "args" : []
            },
            "title" : "",
            "bundleId" : "com.tencent.xin",
            "label" : "微信",
            "path" : "",
            "name" : "",
            "pid" : 1357
        }
    """
    curAppInfo = self.wdaClient.app_current()
    logging.debug("curAppInfo=%s", curAppInfo)
    return curAppInfo
```

调用：

```
# 获取获取当前活跃app及activity方法
curAppInfo = self.get_PackageActivity_iOS()
curAppStateInt = curAppInfo["state"]
curStateEnum = ApplicationState(curAppStateInt)
logging.debug("curStateEnum=%s", curStateEnum)
bundleId = curAppInfo["bundleId"]
logging.debug("bundleId=%s", bundleId)
curAppName = curAppInfo["label"]
logging.debug("curAppName=%s", curAppName)

package = bundleId
```

获取app状态

```
def iOSGetAppState(self, appBundleId):
    """get iOS app state

    Args:
        appBundleId (str): iOS app bundle id
    Returns:
        bool, enum/dict:
            true, app status enum
            false, error info dict
    Raises:
    """
    curAppState = self.wdaClient.app_state(appBundleId)
    logging.debug("curAppState=%s", curAppState)
    """
    {
        "value" : 4,
        "sessionId" : "5BBD460B-F420-461D-A5E3-244A74CDF5CE
    }
    """
    # # <GenericDict, len() = 3>
    # # curAppStateValue = curAppState[0]
    # # curAppStatus = curAppState.status
    # # curAppSessionId = curAppState.sessionId
    # # logging.debug("curAppStatus=%s, curAppSessionId=%s"
    # curAppStateValue = curAppState.value
    # logging.debug("curAppStateValue=%s", curAppStateValue
    # curStateEnum = ApplicationState(curAppStateValue)
    # logging.debug("curStateEnum=%s", curStateEnum)
    # return curStateEnum
    isGetOk, respInfo = self.processWdaResponse(curAppState
    if isGetOk:
        respValue = respInfo
        curStateEnum = ApplicationState(respValue)
        logging.debug("curStateEnum=%s", curStateEnum)
        respInfo = curStateEnum
    return isGetOk, respInfo
```

启动app

```
def iOSLaunchApp(self, appBundleId):
    """Launch iOS app

    Args:
        appBundleId (str): iOS app bundle id
    Returns:
        bool, None/str:
            true, None
            false, str: error message
    Raises:
    """
    launchResp = self.wdaClient.app_launch(appBundleId)
    logging.debug("launchResp=%s", launchResp)
    isLaunchOk, respInfo = self.processWdaResponse(launchR
    return isLaunchOk, respInfo
```

停止app

```
def iOSTerminateApp(self, appBundleId):
    """Terminate iOS app

    Args:
        appBundleId (str): iOS app bundle id
    Returns:
        bool, bool/str:
            true, bool
                True: terminal Ok
                False: terminal fail
                    eg: current not running 设置, if termin
            false, str: error message
    Raises:
    """
    # isTerminalOk = False
    # respInfo = None
    # self.wdaClient.session().app_terminate(appBundleId)
    # self.wdaClient().app_terminate(appBundleId)
    terminateResp = self.wdaClient.app_terminate(appBundle
    logging.debug("terminateResp=%s", terminateResp)
    # respStatus = resp.status
    # respValue = resp.value
    # respSessionId = resp.sessionId
    # logging.info("respStatus=%s, respValue=%s, respSessi
    # if respStatus == 0:
    #     isTerminalOk = True
    #     respInfo = None
    # else:
    #     errInfo = {
    #         "status": respStatus,
    #         "value": respValue,
    #     }
    #     respInfo = errInfo
    isTerminalOk, respInfo = self.processWdaResponse(termi
    return isTerminalOk, respInfo
```

调用：

```
if isTerminateFirst:
    if isDebugState:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_S
        logging.info("before terminal: curState=%s", curSta
    # stop before start to avoid current page is not homepa
    isTerminalOk, respInfo = self.iOSTerminateApp(iOS_AppId
    logging.info("%s: isTerminalOk=%s, respInfo=%s", iOS_Ap
    if isDebugState:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_S
        logging.info("after terminal: curState=%s", curStat
```

首次登录引导页

首次登录app时，引导页，多次左滑进入app主页

```
def swipeLeftGuideToMain(self):
    """
        处理iOS中app，在首次登录后，引导页，需要多次左滑，最后进入主
    """

    """
        途虎养车 左滑引导页 1页/3页:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeScrollView type="XCUIE
                        <XCUIElementTypeImage type="XCUIEle
                    </XCUIElementTypeScrollView>
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        途虎养车 左滑引导页 2页/3页:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeScrollView type="XCUIE
                        <XCUIElementTypeImage type="XCUIEle
                    </XCUIElementTypeScrollView>
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        途虎养车 左滑引导页 3页/3页 立即进入:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeScrollView type="XCUIE
                        <XCUIElementTypeImage type="XCUIEle
                    </XCUIElementTypeScrollView>
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    # GuideText = "guide"
    # parentScrollViewClassChain = "/XCUIElementTypeScrollV
    # guideImgeQuery = {"type":"XCUIElementTypeImage", "nam
    # guideImgeQuery["parent_class_chains"] = [ parentScro
    # isFound, guideImgeElement = findElement(curSession, g
    # if isFound:
    #     # foundAndClicked = clickElement(curSession, guid
    #     swipeLeft(curSession)

    foundAndSwipeGuideToMain = False
```

```
    swipeNum = 0

    guideP = re.compile("guide") # guide0, guide1, guide2
    guideImageChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict,
        },
        {
            "tag": "XCUIElementTypeScrollView",
            "attrs": self.FullScreenAttrDict,
        },
        {
            "tag": "XCUIElementTypeImage",
            "attrs": {"name": guideP, **self.FullScreenAtt
        },
    ]

    while True:
        curPageXml = self.get_page_source()
        soup = CommonUtils.xmlToSoup(curPageXml)

        guideImageSoup = CommonUtils.bsChainFind(soup, gui
        if not guideImageSoup:
            break

        parentScrollViewSoup = guideImageSoup.parent
        if not parentScrollViewSoup:
            break

        validButtonSoupList = []

        nextSiblingList = parentScrollViewSoup.next_sibling
        for eachNextSiblingSoup in nextSiblingList:
            if hasattr(eachNextSiblingSoup, "attrs"):
                soupAttrDict = eachNextSiblingSoup.attrs #
                soupType = soupAttrDict.get("type") # 'XCU
                soupName = soupAttrDict.get("name") # 'loa
                soupVisible = soupAttrDict.get("visible") #
                isButton = soupType == "XCUIElementTypeBut
                # isLoadingName = bool(re.search(soupName,
                isLoadingName = bool(re.search("loading", s
                isVisible = soupVisible == "true"
                isValid = isButton and isLoadingName and is
                if isValid:
                    validButtonSoupList.append(eachNextSib

        if validButtonSoupList:
            validButtonNum = len(validButtonSoupList)
            if validButtonNum == 1:
```

```
                # end page, click button, into main page
                lastPageButtonSoup = validButtonSoupList[0]
                self.clickElementCenterPosition(lastPageBut
                foundAndSwipeGuideToMain = True
                break
            elif validButtonNum > 1:
                swipeNum += 1
                # not end, should continue to swipe left
                self.swipe("SwipeLeft")
                logging.info("Swipe left for guide page %d"

    return foundAndSwipeGuideToMain
```

对应页面：

- 恒易贷
  - 第一页：
    -  hengyidai_left_guide_page_1
  - 最后一页：
    -  hengyidai_left_guide_page_3
- 途虎养车
  - 第一页
    -  tuhucar_left_guide_3_1
  - 第二页
    -  tuhucar_left_guide_3_2
  - 第三页
    -  tuhucar_left_guide_3_3

微信相关

判断是否处于微信

```
def iOSisInWeixin(self):
    tabBarClassChain = "/XCUIElementTypeTabBar[`rect.width

    """
        <XCUIElementTypeTabBar type="XCUIElementTypeTabBar"
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
            <XCUIElementTypeButton type="XCUIElementTypeBut
            <XCUIElementTypeButton type="XCUIElementTypeBut
            <XCUIElementTypeButton type="XCUIElementTypeBut
            <XCUIElementTypeButton type="XCUIElementTypeBut
        </XCUIElementTypeTabBar>
    """
    weixinTabQuery = {"type":"XCUIElementTypeButton", "name
    weixinTabQuery["parent_class_chains"] = [ tabBarClassCh
    isFoundWeixin, respInfo = self.findElement(query=weixin
    iOSisInWeixin = isFoundWeixin
    return iOSisInWeixin
```

调用：

```
beInWeixin = self.iOSisInWeixin()
```

get_navigationBar_bounds 计算微信顶部系统导航栏的区域bounds

```
# def get_navigationBar_bounds(self, pageSrcXml):
def get_navigationBar_bounds(self):
    """
        计算微信顶部系统导航栏的区域bounds
    """
    bounds = None
    """
        微信 顶部 导航栏:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeStaticText type="XCUIElemer
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>
    """
    naviBarQuery = {"type":"XCUIElementTypeNavigationBar",
    isFound, naviBarElement = self.findElement(naviBarQuery
    if isFound:
        isVisible = naviBarElement.visible
        if isVisible:
            curRect = naviBarElement.bounds
            logging.debug("curRect=%s", curRect)
            x0 = curRect.x
            y0 = curRect.y
            x1 = curRect.x1
            y1 = curRect.y1
            bounds = [x0, y0, x1, y1] # [0, 20, 375, 64]

    logging.debug("bounds=%s", bounds)
    return bounds
```

调用举例：

```
def calcHeadY(curSession):
    bounds = get_navigationBar_bounds(curSession)
    y1 = bounds[-1]
    headY = y1
    return headY
```

安全模式

背景

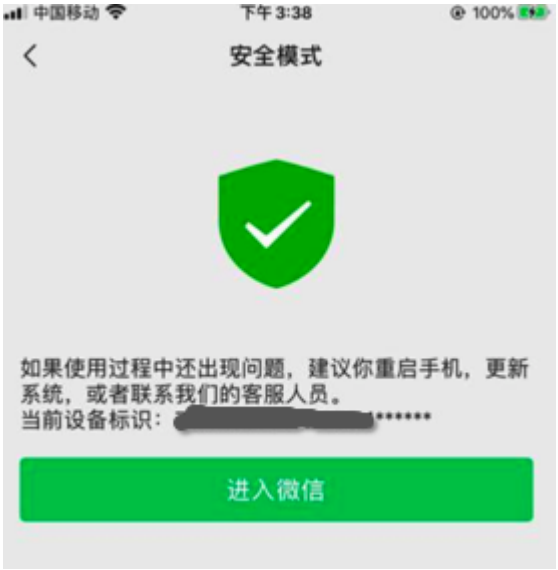
在 `facebook-wda` 调试期间，通过代码控制微信app的退出和启动，多次如此调试后就会导致微信检测到以为发生了多次的异常，所以会进入**安全模式**。

此时只能根据提示，一次次点击 `下一步` ，才能最终退出 `安全模式` ：


中国移动
下午 3:09
100%
安全模式
建议你重启手机，避免微信启动异常
重启手机
不重启手机，进入下一步
下一步


中国移动
下午 3:22
100%
安全模式
清理缓存会清理你的手机本地缓存文件，但不会清理你的消息数据，使用后需要重新登录微信
清理缓存
不清理缓存，进入下一步
下一步


中国移动
下午 3:35
100%
安全模式
如果问题还没解决，你可以上传手机日志文件，协助技术人员解决问题。所上传的文件不会包含聊天记录等私人内容，且不会被对外传播
上传文件
显示上传文件列表
不上传文件，进入下一步
下一步



详见：

【已解决】自动抓包工具适配iOS：当前检测出微信连续异常，你可以尝试一下方法修复 下一步

代码：`iOSWeixinExceptionNextStep`

自动检测是否处于 `安全模式` ，并自动点击直到退出 `安全模式` 的代码如下，供参考：

```
def iOSMakesureIntoWeixin(self):
    """Makesure into weixin main page
        if exception, process it, until into weixin page
    """
    # maxRetryNum = 3
    maxRetryNum = 5
    beInWeixin = self.iOSisInWeixin()

    while (not beInWeixin) and (maxRetryNum > 0):
        beInWeixin = self.iOSisInWeixin()
        if not beInWeixin:
            # try process for exception
            foundAndProcessedException = self.iOSWeixinExce
            if foundAndProcessedException:
                beInWeixin = self.iOSisInWeixin()

        maxRetryNum -= 1

    return beInWeixin

def iOSWeixinExceptionNextStep(self):
    foundAndClicked = False

    scrollViewClassChain = "/XCUIElementTypeScrollView[`re

    ButtonLabelNextStep = "下一步"
    ButtonLabelIntoWeixin = "进入微信"

    # nextStepButtonQuery = {"type":"XCUIElementTypeButton'
    # nextStepButtonQuery["parent_class_chains"] = [scroll

    # intoWeixinButtonQuery = {"type":"XCUIElementTypeButt
    # intoWeixinButtonQuery["parent_class_chains"] = [scro

    """
        <XCUIElementTypeScrollView type="XCUIElementTypeScr
            <XCUIElementTypeImage type="XCUIElementTypeImag
            <XCUIElementTypeStaticText type="XCUIElementTyp
            <XCUIElementTypeButton type="XCUIElementTypeBut
                <XCUIElementTypeStaticText type="XCUIElemer
            </XCUIElementTypeButton>
        </XCUIElementTypeScrollView>
    """
    # continousExceptionQuery = {"type":"XCUIElementTypeSta
    # continousExceptionQuery["parent_class_chains"] = [scr
    # isFoundContinousException, respInfo = self.findElemer
    # if isFoundContinousException:
    #     isFoundNextStep, respInfo = self.findElement(quer
    #     if isFoundNextStep:
    #         nextStepElement = respInfo
```

```
        #         foundAndClicked = self.clickElement(nextStepE

        """
            <XCUIElementTypeScrollView type="XCUIElementTypeScr
                <XCUIElementTypeImage type="XCUIElementTypeImag
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementTypeOthe
            </XCUIElementTypeScrollView>
        """
        # # if not foundAndClicked:
        # rebootWeixinQuery = {"type":"XCUIElementTypeStaticTex
        # rebootWeixinQuery["parent_class_chains"] = [scrollVie
        # isFoundRebootWeixin, respInfo = self.findElement(quer
        # if isFoundRebootWeixin:
        #     isFoundNextStep, respInfo = self.findElement(quer
        #     if isFoundNextStep:
        #         nextStepElement = respInfo
        #         foundAndClicked = self.clickElement(nextStepE

        """
            <XCUIElementTypeScrollView type="XCUIElementTypeScr
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                    <XCUIElementTypeImage type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElementType
                </XCUIElementTypeOther>
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementTypeOthe
            </XCUIElementTypeScrollView>
        """
        # # if not foundAndClicked:
        # clearCacheQuery = {"type":"XCUIElementTypeStaticText"
        # clearCacheQuery["parent_class_chains"] = [scrollView(
        # isFoundClearCache, respInfo = self.findElement(query=
        # if isFoundClearCache:
        #     isFoundNextStep, respInfo = self.findElement(quer
```

```
        #     if isFoundNextStep:
        #         nextStepElement = respInfo
        #         foundAndClicked = self.clickElement(nextStepE

        """
            <XCUIElementTypeScrollView type="XCUIElementTypeScr
                <XCUIElementTypeImage type="XCUIElementTypeImag
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementTypeOthe
            </XCUIElementTypeScrollView>
        """

        """
            <XCUIElementTypeScrollView type="XCUIElementTypeScr
                <XCUIElementTypeImage type="XCUIElementTypeImag
                <XCUIElementTypeButton type="XCUIElementTypeBut
                <XCUIElementTypeStaticText type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTypeBut
                    <XCUIElementTypeStaticText type="XCUIElemen
                </XCUIElementTypeButton>
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementTypeOthe
            </XCUIElementTypeScrollView>
        """

        EachStepNoticeList = [
            # ("当前检测出微信连续异常，你可以尝试以下方法修复：", Butt
            ("当前检测出微信连续异常", ButtonLabelNextStep),
            ("建议你重启手机，避免微信启动异常", ButtonLabelNextStep
            ("清理缓存会清理你的手机本地缓存文件，但不会清理你的消息数据
            ("如果问题还没解决，你可以上传手机日志文件，协助技术人员解决
            ("如果使用过程中还出现问题，建议你重启手机，更新系统，或者联
        ]
        for (curStepNotice, buttonLabel) in EachStepNoticeList:
            # curNoticeQuery = {"type":"XCUIElementTypeStaticTe
            # curNoticeQuery = {"type":"XCUIElementTypeStaticTe
            curNoticeQuery = {"type":"XCUIElementTypeStaticText
            curNoticeQuery["parent_class_chains"] = [scrollView
            isFoundCurNotice, respInfo = self.findElement(query
            if isFoundCurNotice:
```

```
            # isFoundNextStep, respInfo = self.findElement(
            # isFoundButton, respInfo = self.findElement(qu
            buttonQuery = {"type":"XCUIElementTypeButton",
            buttonQuery["parent_class_chains"] = [scrollVie
            isFoundButton, respInfo = self.findElement(quer
            if isFoundButton:
                buttonElement = respInfo
                clickOk = self.clickElement(buttonElement)
                if clickOk:
                    foundAndClicked = clickOk

    return foundAndClicked
```

微信公众号

此处整理微信中关于微信公众号的部分的常见代码段。

isPublicAccountSearchPage_iOS 判断是否处于微信公众号搜索页

```
def isPublicAccountSearchPage_iOS(self, page):
    """Check whether current page is Weixin Public account
    isPublicAccountSearch = False
    curInputValue = ""

    """
        is search:
            <XCUIElementTypeImage type="XCUIElementTypeImag
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeSearchField typ
                            <XCUIElementTypeButton type="XC
                                <XCUIElementTypeStaticText
                            </XCUIElementTypeButton>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeImage>

            <XCUIElementTypeImage type="XCUIElementTypeImag
            <XCUIElementTypeButton type="XCUIElementTypeBut
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeImage type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeSearchField type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                    </XCUIElementTypeSearchField>
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeButton>
                </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
            </XCUIElementTypeImage>

            <XCUIElementTypeImage type="XCUIElementTypeImag
            <XCUIElementTypeButton type="XCUIElementTypeBut
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeImage type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeSearchField type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                    </XCUIElementTypeSearchField>
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
```

```
                    </XCUIElementTypeButton>
                </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
            </XCUIElementTypeImage>
    """
    soup = CommonUtils.xmlToSoup(page)
    widthStr = str(self.X)
    foundImage = soup.find(
        'XCUIElementTypeImage',
        attrs={"type":"XCUIElementTypeImage", "enabled":"tr
    )
    if foundImage:
        foundSearchField = foundImage.find("XCUIElementType
            attrs={"type": "XCUIElementTypeSearchField", "r
        )
        if foundSearchField:
            isPublicAccountSearch = True
            # curInputValue = foundSearchField.attrs["value
            curInputValue = foundSearchField.attrs.get("va

    return isPublicAccountSearch, curInputValue
```

isPublicAccountFocusOrIntoPage_iOS

```
def isPublicAccountFocusOrIntoPage_iOS(self, page):
    """Check whether current page is Weixin Public focus o
    isPublicAccountFocusOrInto = False

    """
        is Focus:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>

                    <XCUIElementTypeTable type="XCUIElement
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeStaticText type="X
                        <XCUIElementTypeImage type="XCUIEl
                        <XCUIElementTypeButton type="XCUIE
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeButton>
                        <XCUIElementTypeStaticText type="X
                        <XCUIElementTypeButton type="XCUIE
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeButton>
                        <XCUIElementTypeButton type="XCUIE
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeButton>
                        <XCUIElementTypeButton type="XCUIE
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeButton>

        is enter into:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>

                <XCUIElementTypeTable type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemen
                        <XCUIElementTypeStaticText type="X
                    </XCUIElementTypeButton>
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                        <XCUIElementTypeStaticText type="X
                    </XCUIElementTypeButton>
```

```
                    <XCUIElementTypeButton type="XCUIElemer
                        <XCUIElementTypeStaticText type="X(
                    </XCUIElementTypeButton>
                    <XCUIElementTypeButton type="XCUIElemer
                        <XCUIElementTypeStaticText type="X(
                    </XCUIElementTypeButton>

        """
        soup = CommonUtils.xmlToSoup(page)
        foundNaviBar = soup.find(
            'XCUIElementTypeNavigationBar',
            attrs={"type":"XCUIElementTypeNavigationBar", "enal
        )
        # logging.debug("foundNaviBar=%s", foundNaviBar)
        if foundNaviBar:
            foundTypeOther = foundNaviBar.find("XCUIElementType
                attrs={"type": "XCUIElementTypeOther", "enable(
            )
            logging.debug("foundTypeOther=%s", foundTypeOther)
            if foundTypeOther:
                # typeOtherName = foundTypeOther.attrs["name"]
                typeOtherName = foundTypeOther.attrs.get("name'
                logging.debug("typeOtherName=%s", typeOtherName
                isTypeOtherNameNotEmpty = bool(typeOtherName)
                isTypeOtherNameEmpty = not isTypeOtherNameNotEr
                logging.debug("isTypeOtherNameEmpty=%s", isType
                if isTypeOtherNameEmpty:
                    foundIntoAccount = soup.find(
                        'XCUIElementTypeButton',
                        attrs={"type":"XCUIElementTypeButton",
                    )
                    logging.debug("foundIntoAccount=%s", found]
                    if foundIntoAccount:
                        prevSiblingList = foundIntoAccount.prev
                        logging.debug("prevSiblingList=%s", pre
                        curAccountZhcnName = self.account_zw
                        TypeStaticText = "XCUIElementTypeStatic
                        for eachPrevSibling in prevSiblingList:
                            # curType = eachPrevSibling.attrs['
                            # curName = eachPrevSibling.attrs['
                            # curType = eachPrevSibling.attrs.g
                            # curName = eachPrevSibling.attrs.g
                            if hasattr(eachPrevSibling, "attrs"
                                curType = eachPrevSibling.attrs
                                curName = eachPrevSibling.attrs
                                if (curType == TypeStaticText)
                                    isPublicAccountFocusOrInto
                            else:
                                logging.debug("eachPrevSibling=
```

```
    logging.debug("isPublicAccountFocusOrInto=%s", isPublic
    return isPublicAccountFocusOrInto
```

调用：

```
def isCurPageInOffline_iOS(self, page):
    """Check whether current page is belong to (unexpeced)

        offline page include:
            * 微信首页
            * 微信通讯录
            * 微信公众号列表页
            * 微信公众号搜索-待输入
            * 微信公众号搜索-输入公众号ID
            * 微信公众号搜索-搜索结果
            * 微信公众号-关注公众号
            * 微信公众号-进入公众号
    """
    isOffline = False

    OfflinePageNaviBarNameList = [
        "微信",
        "通讯录",
        "公众号",
    ]
    hasNaviBar, naviBarName = self.isPageHasNaviBar_iOS(pag
    if hasNaviBar:
        if naviBarName in OfflinePageNaviBarNameList:
            # is in some weixin page
            isOffline = True

    if not isOffline:
        isPublicAccountSearch, curInputValue = self.isPubli
        if isPublicAccountSearch:
            # is in public account search page
            isOffline = True

    if not isOffline:
        isPublicAccountFocusOrIntoPage = self.isPublicAccou
        if isPublicAccountFocusOrIntoPage:
            # is in public account focus or enter into page
            isOffline = True

    return isOffline
```

findWeixinPublicAccountZhcnFullName 微信公众号搜索结果列表页 查找 微信公众号的中文（全）名

对于页面：


中国移动
上午 11:50
100%
gh_cfcfcee032cc
取消
公众号
动卡空间
分享用卡活动，传播正能量
微信号：gh_cfcfcee032cc
已关注
今日推荐
兴业 信用卡
交通 信用卡
没有更多的搜索结果

从中查找出 公众号的中文名（的全称）

```
# def findWeixinPublicAccountZhcnSoup(self, soup, curAccour
def findWeixinPublicAccountZhcnFullName(self, soup, curAcc
    """Find weixin public account element's zh-CN full name

    Args:
        soup (soup): soup of current page xml
    Returns:
        public account zh-CN full name
    Raises:
    """
    # accountZhcnTextSoup = None
    accountZhcnFullName = ""
    parentNodeLocator = None

    """
        搜索结果中文名节点是Text
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>

        搜索结果中文名节点是Other，其下是多个Text节点:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeImage type="XCUIElementType
                <XCUIElementTypeImage type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
```

```
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>

            公众号中文名全部是绿色的:
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                    <XCUIElementTypeOther type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                            <XCUIElementTypeStaticText type="XC
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElementType
                        <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElementType
                        <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElementType
                        <XCUIElementTypeStaticText type="XCUIE
                        <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElementType
                        <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            """
            foundAccountId = soup.find(
                'XCUIElementTypeStaticText',
                attrs={"value": curAccountId, "name": curAccountId,
            )
            logging.debug("foundAccountId=%s", foundAccountId)
            # foundAccountId=<XCUIElementTypeStaticText enabled="tr
            if foundAccountId:
                idParent = foundAccountId.parent
                logging.debug("idParent=%s", idParent)
                if idParent:
                    # method 1: two prev.prev
                    # # idParentPrev = idParent.previous_sibling
                    # idParentPrev = idParent.previous_sibling.prev
                    # accountDescNode = idParentPrev
                    # logging.info("accountDescNode=%s", accountDes
                    # if accountDescNode:
                    #     # accountZhcnNode = accountDescNode.prev
                    #     accountZhcnNode = accountDescNode.previou
                    #     logging.info("accountZhcnNode=%s", accoun

                    # # method 2: siblings[-2] of XCUIElementTypeOt
```

```
            # idParentPrevSiblingList = idParent.previous_s

            # accountDescNode = None
            # accountZhcnNode = None

            # TypeOther = "XCUIElementTypeOther"
            # typeOtherNodeCurIdx = 0
            # AccountDescNodeIdx = 1
            # AccountZhcnNodeIdx = 2

            # for eachPrevSiblingNode in idParentPrevSiblir
            #     curNodeName = eachPrevSiblingNode.name
            #     isTypeOtherNode = curNodeName == TypeOthe
            #     if isTypeOtherNode:
            #         typeOtherNodeCurIdx += 1

            #         if AccountDescNodeIdx == typeOtherNod
            #             accountDescNode = eachPrevSiblinç
            #         elif AccountZhcnNodeIdx == typeOtherN
            #             accountZhcnNode = eachPrevSiblinç

            #     hasFoundAll = accountDescNode and account
            #     if hasFoundAll:
            #         break

            # logging.info("accountDescNode=%s", accountDes
            # logging.info("accountZhcnNode=%s", accountZhc

            # if accountZhcnNode:
            #     accountZhcnTextSoup = accountZhcnNode.fir
            #         'XCUIElementTypeStaticText',
            #         attrs={ "type": "XCUIElementTypeStat
            #     )

            # method 3: parent'parent is 搜一搜, direct chi
            idParentParent = idParent.parent
            if idParentParent:
                otherSoupList = idParentParent.find_all(
                    "XCUIElementTypeOther",
                    attrs={"type": "XCUIElementTypeOther",
                    recursive=False,
                )
                if otherSoupList and (len(otherSoupList) >=
                    firstOtherSoup = otherSoupList[0]
                    if firstOtherSoup.attrs["name"] == "公众
                        secondOtherSoup = otherSoupList[1]
                        zhcnNameSoupList = secondOtherSoup.
                            "XCUIElementTypeStaticText",
                            attrs={"type": "XCUIElementType
                        )
                        if zhcnNameSoupList:
```

```
                            for eachTextSoup in zhcnNameSou
                                curPartName = eachTextSoup.
                                accountZhcnFullName += curP

                            if accountZhcnFullName:
                                secondOtherAttrDict = secon
                                parentX = secondOtherAttrDi
                                parentY = secondOtherAttrDi
                                parentWidth = secondOtherAt
                                parentHeight = secondOtherA
                                parentNodeLocator = {
                                    "type": "XCUIElementTyp
                                    "enabled": "true",
                                    "visible": "true",
                                    "x": parentX,
                                    "y": parentY,
                                    "width": parentWidth,
                                    "height": parentHeight,
                                }

    # return accountZhcnTextSoup
    # return accountZhcnFullName
    return accountZhcnFullName, parentNodeLocator
```

调用举例：

```
curAccountId = "gh_cfcfcee032cc"
curPageXml = self.get_page_source()
soup = CommonUtils.xmlToSoup(curPageXml)

accountZhcnName, parentNodeLocator = self.findWeixinPublicA
```

设置

此处整理iOS的内置app，bundle id是 `com.apple.Preferences` 的 `设置` 相关的通用代码段。

启动内置app：`设置`

```
def iOSLaunchSettings(self, isTerminateFirst=True, isDebugS
    """iOS terminal and re-launch Settings=Preferences app

    Args:
        isTerminateFirst (bool): force terminal Settings be
        isDebugState (bool): enable debug for app state
    Returns:
        bool
    Raises:
    """

    iOS_AppId_Settings = "com.apple.Preferences"

    if isTerminateFirst:
        if isDebugState:
            isGetOk, curState = self.iOSGetAppState(iOS_App
            logging.info("before terminal: curState=%s", cu
        # stop before start to avoid current page is not ho
        isTerminalOk, respInfo = self.iOSTerminateApp(iOS_A
        logging.info("%s: isTerminalOk=%s, respInfo=%s", iO
        if isDebugState:
            isGetOk, curState = self.iOSGetAppState(iOS_App
            logging.info("after terminal: curState=%s", cur

    # settingsSession = self.wdaClient.session(iOS_AppId_Se
    # logging.debug("settingsSession=%s" % settingsSession)
    # launchResult = self.wdaClient.app_launch(iOS_AppId_Se
    # logging.debug("launchResult=%s", launchResult)
    isLaunchOk, respInfo = self.iOSLaunchApp(iOS_AppId_Set
    logging.info("isLaunchOk=%s, respInfo=%s", isLaunchOk,
    # logging.info("launchResult: value=%s, status=%s, sess
    # launchResult: value=None, status=0, sessionId=79A39B7
    # logging.info("launchResult=%s", str(launchResult))
    # launchResult=GenericDict(value=None, sessionId='79A39
    if isDebugState:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_S
        logging.info("after launch: curState=%s", curState)
    return isLaunchOk, respInfo
```

调用：

```
isLaunchOk, respInfo = self.iOSLaunchSettings()
```

更新WiFi代理

此处整理用 `facebook-wad` 自动实现关闭和恢复代理的整个自动化过程。

此处把

【已解决】iOS自动化安装app：给当前WiFi去掉代理以及自动安装app后再恢复之前代理

中：

- 关闭代理
  - 给当前WiFi去掉代理
- 恢复代理
  - 给当前WiFi 加上代理

贴出来相关的逻辑和代理，供了解：iOS的自动化，是什么样的，代码大概怎么写

流程和相关代码：

```
newProxyInfo = {
    "type": "关闭",
    "value": None,

}
isUpdateOk, respInfo = self.iOSWifiDetailUpdateProxy(newPr
```

用于去关闭代理

（自动通过AppStore去安装iOS的app后）后续去恢复（之前保存的）代理：

```
# # for debug
# # 'authUser': 'user', 'authPassword': '•••'
# proxyConfigInfo = {
#     'type': '手动',
#     'value': {
#         'server': '192.168.31.46',
#         'port': '8081',
#         # 'authenticate': '0'
#         'authenticate': '1',
#         'authUser': 'user',
#         'authPassword': 'pwd',
#     }
# }

# restore proxy if necessary
if proxyConfigInfo:
    # isRestoreOk = self.settingsRestoreWiFiProxy(proxyCon
    isRestoreOk, respInfo = self.iOSWifiDetailUpdateProxy(
    logStr = "%s to restore previous proxy %s" % (isRestore
    if isRestoreOk:
        logging.info(logStr)
    else:
        logging.error(logStr)
```

具体实现：

```
def iOSWifiDetailUpdateProxy(self, newProxyInfo):
    """iOS launch Settings, into WiFi list page, find curre
        then try update to new proxy config info
    Args:
        newProxyInfo (dict): new proxy config info
    Returns:
        bool, dict/str
            True, old proxy config info dict
            False, error message str
    Raises:
    """
    isUpdateProxyOk = False
    respInfo = None

    isGetProxyTypeOk, respInfo = self.iOSLaunchSettingsAndG
    if not isGetProxyTypeOk:
        respInfo = "Not find current WiFi proxy config type
        return isUpdateProxyOk, respInfo

    curProxySoup = respInfo
    curProxyAttrDict = curProxySoup.attrs
    curTypeName = curProxyAttrDict.get("value")

    newTypeName = newProxyInfo["type"]
    if (newTypeName == "关闭") and (curTypeName == "关闭"):
        isUpdateProxyOk = True
        oldProxyInfo = {
            "type": "关闭",
            "value": None,
        }
        respInfo = oldProxyInfo
        return isUpdateProxyOk, respInfo
    else:
        # into config proxy page
        self.clickElementCenterPosition(curProxySoup)
        # get old proxy value
        # update to close
        # save
        isUpdateProxyOk, respInfo = self.iOSProxyConfigUpda
        logging.info("Update proxy result: %s, %s", isUpdat
        return isUpdateProxyOk, respInfo
```

其逻辑是：

启动 `设置` ，直到进入 当前已连接的WiFi的详情页，获取到当前代理的类型

其具体实现是：

```
def iOSLaunchSettingsAndGetProxyType(self):
    """
        iOS launch Settings, and into WiFi list page, clic
    Args:
    Returns:
        bool, soup/str
            True, proxy type soup
            False, error message str
    Raises:
    """
    isGetTypeOk = False
    respInfo = None

    isIntoWiFiListOk, respInfo = self.iOSLaunchSettingsAnd
    if not isIntoWiFiListOk:
        errMsg = respInfo
        respInfo = "Fail into WiFi list page for %s" % errM
        return isGetTypeOk, respInfo

    isIntoDetailOk = self.iOSFromWifiListIntoWifiDetail()
    if not isIntoDetailOk:
        respInfo = "Fail go into WiFi detail page"
        return isGetTypeOk, respInfo

    proxyTypeSoup = self.iOSGetCurrentWiFiProxyType()
    if proxyTypeSoup:
        isGetTypeOk = True
        respInfo = proxyTypeSoup

    return isGetTypeOk, respInfo
```

即：

- 先启动 设置
- 再进去WiFi的列表页
- 再进去当前已连接WiFi的详情页

具体实现：

```
def iOSLaunchSettingsAndIntoWiFiList(self):
    """
        iOS launch Settings, then into / (sometime) already

    Args:
    Returns:
        bool, None/str
            True, None
            False, error message str
    Raises:
    """
    isIntoWiFiListOk = False
    respInfo = None

    isLaunchOk, respInfo = self.iOSLaunchSettings()
    if not isLaunchOk:
        respInfo = "Fail to launch 设置"
        return isIntoWiFiListOk, respInfo

    # Special: sometime alreay being in WiFi list page, so
    isInWifiList = self.iOSIsInWiFiList()
    if isInWifiList:
        isIntoWiFiListOk = True
        respInfo = None
        return isIntoWiFiListOk, respInfo
    else:
        # foundAndClickedWifi = self.iOSFromSettingsIntoWi
        foundAndClickedWifi = CommonUtils.multipleRetry({"
        if not foundAndClickedWifi:
            respInfo = "Not find 无线局域网 in 设置"
            return isIntoWiFiListOk, respInfo

        isInWifiList = self.iOSIsInWiFiList()
        if isInWifiList:
            isIntoWiFiListOk = True
            respInfo = None
            return isIntoWiFiListOk, respInfo
        else:
            isIntoWiFiListOk = False
            respInfo = "Unknown Error"
            return isIntoWiFiListOk, respInfo
```

启动设置

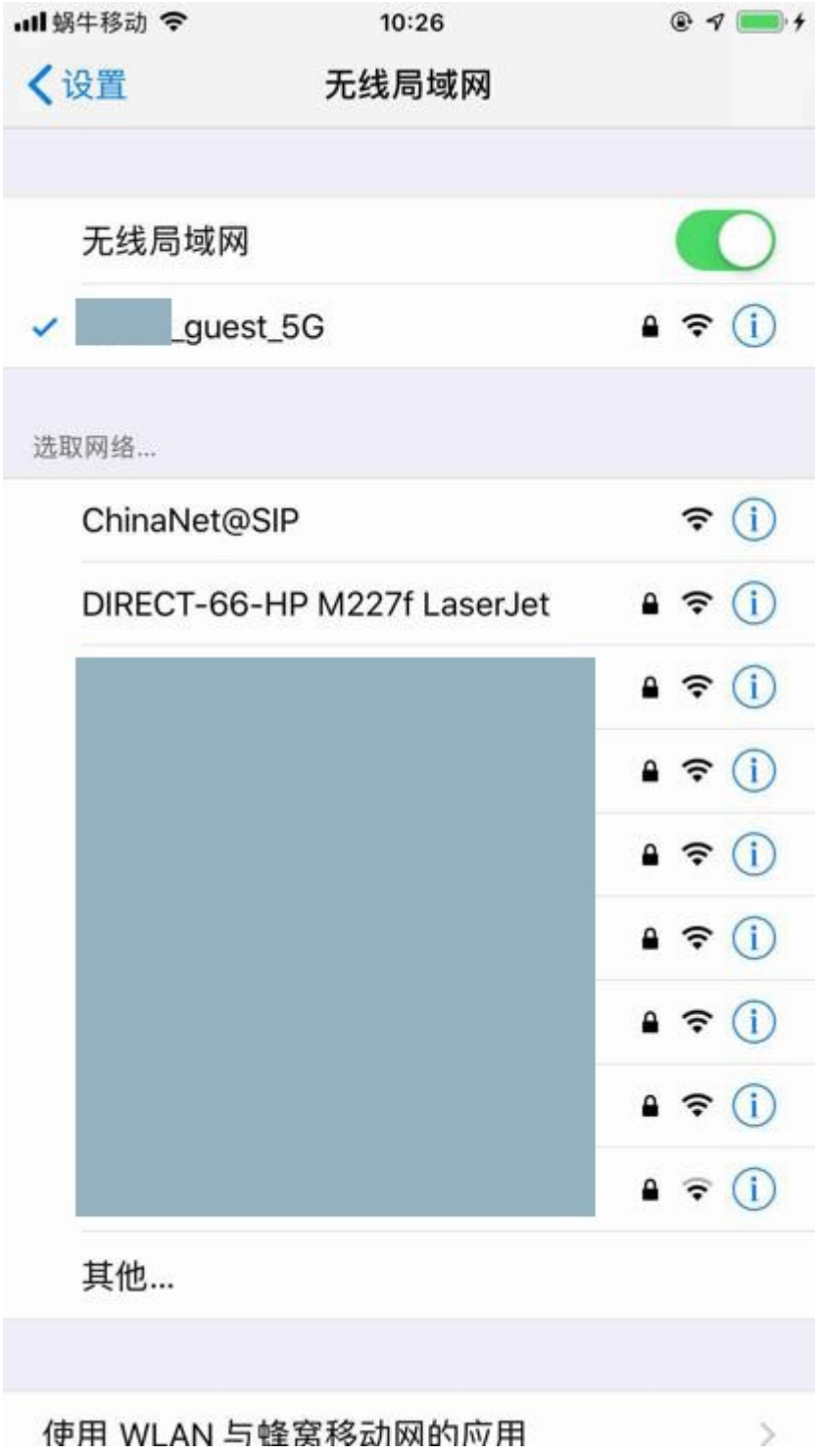
```
def iOSLaunchSettings(self, isTerminateFirst=True, isDebugS
    """iOS terminal and re-launch Settings=Preferences app

    Args:
        isTerminateFirst (bool): force terminal Settings be
        isDebugState (bool): enable debug for app state
    Returns:
        bool
    Raises:
    """

    iOS_AppId_Settings = "com.apple.Preferences"

    if isTerminateFirst:
        if isDebugState:
            isGetOk, curState = self.iOSGetAppState(iOS_App
            logging.info("before terminal: curState=%s", cu
        # stop before start to avoid current page is not ho
        isTerminalOk, respInfo = self.iOSTerminateApp(iOS_A
        logging.info("%s: isTerminalOk=%s, respInfo=%s", iO
        if isDebugState:
            isGetOk, curState = self.iOSGetAppState(iOS_App
            logging.info("after terminal: curState=%s", cur

    # settingsSession = self.wdaClient.session(iOS_AppId_Se
    # logging.debug("settingsSession=%s" % settingsSession)
    # launchResult = self.wdaClient.app_launch(iOS_AppId_Se
    # logging.debug("launchResult=%s", launchResult)
    isLaunchOk, respInfo = self.iOSLaunchApp(iOS_AppId_Sett
    logging.info("isLaunchOk=%s, respInfo=%s", isLaunchOk,
    # logging.info("launchResult: value=%s, status=%s, sess
    # launchResult: value=None, status=0, sessionId=79A39B7
    # logging.info("launchResult=%s", str(launchResult))
    # launchResult=GenericDict(value=None, sessionId='79A39
    if isDebugState:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_S
        logging.info("after launch: curState=%s", curState)
    return isLaunchOk, respInfo
```

对应页面 设置 首页：

进入WiFi列表页

在WiFi列表页中 找 无线局域网，即可以找到当前已连接的Wifi

页面 WiFi列表页：



点击无线局域网 进去 WiFi列表页

```
def iOSFromSettingsIntoWifiList(self):
    """from settings page, click 无线局域网 into WiFi list p
    foundAndClickedWifi = False
    """
        设置 顶部 无线局域网:
            <XCUIElementTypeTable type="XCUIElementTypeTab

                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeCell>
    """
    # parentWifiCellClassChain = "/XCUIElementTypeCell[`nar
    parentWifiCellClassChain = "/XCUIElementTypeCell[`name
    wifiTextQuery = {"type":"XCUIElementTypeStaticText", "
    wifiTextQuery["parent_class_chains"] = [ parentWifiCel
    foundAndClickedWifi = self.findAndClickElement(query=wi
    return foundAndClickedWifi
```

判断是否已进入WiFi列表页：

```
def iOSIsInWiFiList(self):
    """Check whether is in WiFi list page or not

    Args:
    Returns:
        bool
    Raises:
    """
    isFoundWifi = False
    """
        设置 WiFi列表页:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
            </XCUIElementTypeNavigationBar>
    """
    wifiName = "无线局域网"
    parentNaviBarClassChain = "/XCUIElementTypeNavigationBa
    wifiQuery = {"type":"XCUIElementTypeOther", "name": wi
    wifiQuery["parent_class_chains"] = [ parentNaviBarClass
    isFoundWifi, respInfo = self.findElement(query=wifiQue
    return isFoundWifi
```

且用多次判断，防止单次的失败。

再去从WiFi列表页进入详情页：

```
def iOSFromWifiListIntoWifiDetail(self):
    """from WiFi list page, click more info button into Wi
    isIntoDetailOk = False
    """
        设置 无线局域网 列表页:
            <XCUIElementTypeTable type="XCUIElementTypeTab
            ° ° °
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEl
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeCell>
    """
    curPageXml = self.get_page_source()
    soup = CommonUtils.xmlToSoup(curPageXml)
    blueCheckChainList = [
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeImage",
            # "attrs": {"enabled":"true", "visible":"true",
            "attrs": {"enabled":"true", "name": "UIPreferen
        },
    ]
    blueCheckSoup = CommonUtils.bsChainFind(soup, blueChecl
    if blueCheckSoup:
        parentOtherSoup = blueCheckSoup.parent
        parentCellSoup = parentOtherSoup.parent
        moreInfoSoup = parentCellSoup.find(
            'XCUIElementTypeButton',
            attrs={"type": "XCUIElementTypeButton", "name"
        )
        if moreInfoSoup:
            clickedOk = self.clickElementCenterPosition(mo
            isIntoDetailOk = clickedOk
    return isIntoDetailOk
```

页面 WiFi详情页：

再从当前WiFi详情页，找到代理的类型：

```
def iOSGetCurrentWiFiProxyType(self):
    """from WiFi detail page, get current proxy config type
    proxyTypeSoup = None
    """
        设置 无线局域网 详情页 代理 关闭:
            <XCUIElementTypeTable type="XCUIElementTypeTab
            。。。
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeCell>

        设置 无线局域网 详情页 代理 手动:
            <XCUIElementTypeCell type="XCUIElementTypeCell"
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeStaticText type="XCUIEleme
                <XCUIElementTypeStaticText type="XCUIEleme
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeCell>
    """
    curPageXml = self.get_page_source()
    soup = CommonUtils.xmlToSoup(curPageXml)
    configProxyChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled":"true", "visible":"true",
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled":"true", "visible":"true",
        },
    ]
    configProxySoup = CommonUtils.bsChainFind(soup, configP
    if configProxySoup:
        parentCellSoup = configProxySoup.parent
        # proxyTypeP = re.compile("(手动)|(自动)|(关闭)")
        # proxyTypeSoup = parentCellSoup.find(
        #     'XCUIElementTypeStaticText',
        #     attrs={"type": "XCUIElementTypeStaticText",
```

```
            # )
            proxyTypeSoup = self.iOSFindChildProxyType(parentCe

        return proxyTypeSoup
```

此处返回是 str （类型的字符串名称）或soup（类型的soup节点）

```
def iOSFindChildProxyType(self, parentCellSoup, isReturnSou
    """from parent cell soup, find child proxy type node /

    Args:
        parentCellSoup (soup): Beautifulsoup soup of parent
        isReturnSoup (bool): return soup if true, otherwise
    Returns:
        str/soup:
            str: 手动/自动/关闭
            soup: soup node
    Raises:
    """
    # proxyTypeName = None
    # some cases:
    """
        设置 无线局域网 详情页 代理 关闭:
            <XCUIElementTypeCell type="XCUIElementTypeCell"
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeCell>

        设置 无线局域网 详情页 代理 手动:
            <XCUIElementTypeCell type="XCUIElementTypeCell"
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeCell>

        设置 无线局域网 配置代理 手动:
            <XCUIElementTypeCell type="XCUIElementTypeCell"
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeCell>

        设置 无线局域网 配置代理 关闭:
            <XCUIElementTypeCell type="XCUIElementTypeCell"
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeCell>

        设置 无线局域网 配置代理 自动:
            <XCUIElementTypeCell type="XCUIElementTypeCell"
```

```
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeStaticText type="XCUIElemen
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeCell>
    """
    proxyTypeP = re.compile("(手动)|(自动)|(关闭)")
    proxyTypeSoup = parentCellSoup.find(
        'XCUIElementTypeStaticText',
        attrs={"type": "XCUIElementTypeStaticText", "value"
    )
    if isReturnSoup:
        return proxyTypeSoup
    else:
        proxySoupAttrDict = proxyTypeSoup.attrs
        proxyTypeName = proxySoupAttrDict.get("value")
        return proxyTypeName # '手动'
```

从xml中找到 类型名

之后点击此soup节点

```
def clickElementCenterPosition(self, curElement):
    """Click center position of element

    Args:
        curElement (Element): Beautiful soup / lxml element
    Returns:
        bool
    Raises:
    """
    hasClicked = False
    # centerPos = None
    centerX = None
    centerY = None

    hasBounds = hasattr(curElement, "bounds")
    curBounds = None
    if hasBounds:
        curBounds = curElement.bounds

    if hasBounds and curBounds:
        # wda element
        if hasattr(curBounds, "center"):
            # is wda Rect
            curRect = curBounds
            rectCenter = curRect.center
            centerX = rectCenter[0]
            centerY = rectCenter[1]
    else:
        attrDict = None
        if hasattr(curElement, "attrs"):
            # Beautiful soup node
            attrDict = curElement.attrs
        elif hasattr(curElement, "attrib"):
            # lxml element
            attrDict = dict(curElement.attrib)

        if attrDict:
            logging.info("attrDict=%s", attrDict)
            hasCoordinate = ("x" in attrDict) and ("y" in a
            if hasCoordinate:
                x = int(attrDict["x"])
                y = int(attrDict["y"])
                width = int(attrDict["width"])
                height = int(attrDict["height"])
                centerX = x + int(width / 2)
                centerY = y + int(height / 2)

    if centerX and centerY:
        centerPos = (centerX, centerY)
        self.tap(centerPos)
```

```
            logging.info("Clicked center position: %s", centerP
            hasClicked = True

        return hasClicked
```

进入 配置代理 页面：

去更新代理：

```
def iOSProxyConfigUpdateProxy(self, newProxyInfo):
    """in proxy config = 代理配置 page, update proxy type

    Args:
        newProxyInfo (dict): new proxy info
    Returns:
        bool, dict/str
            True, old proxy config info
            False, error message str
    Raises:
    Examples:
        newProxyInfo exmaples:
            (1) to close:
                {
                    "type": "关闭",
                    "value": None
                }
            (2) to manual, no auth:
                {
                    'type': '手动',
                    'value': {
                        'server': '192.168.31.47',
                        'port': '8081',
                        'authenticate': '0',
                        'authUser': None,
                        'authPassword': None
                    }
                }
            (3) to manual, with auth:
                {
                    'type': '手动',
                    'value': {
                        'server': '192.168.31.47',
                        'port': '8081',
                        'authenticate': '1'
                        'authUser': 'user',
                        'authPassword': 'password'
                    }
                }
            (4) to auto:
                {
                    'type': '自动',
                    'value': {
                        'url': 'your_auto_proxy_url'
                    }
                }
    """
    isUpdateOk = False
    respInfo = None
```

```
        """
            设置 无线局域网 配置代理 关闭:
                <XCUIElementTypeTable type="XCUIElementTypeTab
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XCUIE
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeButton type="XCUIEleme
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeStaticText type="XCUIE
                        <XCUIElementTypeOther type="XCUIElement
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeCell>
                </XCUIElementTypeTable>

            设置 无线局域网 配置代理 手动:
                <XCUIElementTypeTable type="XCUIElementTypeTab
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XCUIE
                        <XCUIElementTypeOther type="XCUIElement
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeStaticText type="XCUIE
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeButton type="XCUIEleme
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XCUIE
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeTextField type="XCUIEle
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIElement
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeTextField type="XCUIEle
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIElement
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XCUIE
                        <XCUIElementTypeSwitch type="XCUIEleme
                    </XCUIElementTypeCell>
```

```
            </XCUIElementTypeTable>

        设置 无线局域网 配置代理 自动:
            <XCUIElementTypeTable type="XCUIElementTypeTab
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
            </XCUIElementTypeTable>
    """
    # check current proxy type: 手动/自动/关闭
    curPageXml = self.get_page_source()
    soup = CommonUtils.xmlToSoup(curPageXml)
    morenInfoChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeButton",
            # "attrs": {"enabled":"false", "visible":"true"
            "attrs": {"visible":"true", "name": "更多信息"}
        },
    ]
    morenInfoSoup = CommonUtils.bsChainFind(soup, morenInfo
    if not morenInfoSoup:
        respInfo = "Fail to find 更多信息 in config proxy pa
        return isUpdateOk, respInfo

    parentCellSoup = morenInfoSoup.parent
    curTypeName = self.iOSFindChildProxyType(parentCellSoup
```

```
        if not curTypeName:
            respInfo = "Fail to find current proxy type name i
            return isUpdateOk, respInfo

        parentTableSoup = parentCellSoup.parent

        newTypeName = newProxyInfo["type"]
        newProxyValue = newProxyInfo["value"]

        curProxyValue = None
        isSameType = False
        isNeedSwitchType = True
        isNeedGetCurValue = True
        isNeedCompareValue = False
        isNeedUpdateNewValue = True
        isNeedStore = True

        if newTypeName == curTypeName:
            isSameType = True
            isNeedSwitchType = False
            isNeedCompareValue = True
        else:
            isNeedSwitchType = True

        if curTypeName == "关闭":
            isNeedGetCurValue = False

        if newTypeName == "关闭":
            isNeedUpdateNewValue = False

        if (newTypeName == "关闭") and (curTypeName == "关闭"):
            isNeedStore = False
            isNeedCompareValue = False

        if isNeedGetCurValue:
            if curTypeName == "手动":
                curProxyValue = self.getManualProxyValue(parent
            elif curTypeName == "自动":
                curProxyValue = self.getAutoProxyValue(parentTa

            if not curProxyValue:
                respInfo = "Fail to get %s proxy value" % curTy
                return isUpdateOk, respInfo

        if isNeedCompareValue:
            # need check value is same or not
            if newProxyValue == curProxyValue:
                # if same, do nothing
                isNeedUpdateNewValue = False
                logging.info("No need change for same proxy typ
```

```
    # common logic process

    if isNeedSwitchType:
        # switch to new type
        isSwitchOk = self.switchToProxyType(parentTableSou
        if isSwitchOk:
            isNeedStore = True
        else:
            respInfo = "Fail to switch to %s proxy" % curT
            return isUpdateOk, respInfo

    if isNeedUpdateNewValue:
        if newTypeName == "手动":
            isUpateValueOk = self.setManualProxyValue(pare
        elif newTypeName == "自动":
            isUpateValueOk = self.setAutoProxyValue(parent

        if isUpateValueOk:
            isNeedStore = True
        else:
            respInfo = "Fail to update new %s proxy config
            return isUpdateOk, respInfo

    if isNeedStore:
        # type and/or value changed, need store
        isStoredOk = self.storeChangedProxyType()
        if not isStoredOk:
            respInfo = "Fail to store after proxy from %s
            return isUpdateOk, respInfo

    isUpdateOk = True
    oldProxyInfo = {
        "type": curTypeName,
        "value": curProxyValue,
    }
    return isUpdateOk, oldProxyInfo
```

实现了，复杂的逻辑处理。总体上支持如下全部各种状态之间互相切换：

- 关闭
- 自动
  - 没开 鉴定
    - 只有
      - 服务器
      - 端口
      - 鉴定=0
  - 开启 鉴定
    - 服务器
    - 端口

        - 鉴定=1
            - 有额外的
                - 用户名
                - 密码
- 自动
    - 有
        - url值

且给出了传入的典型参数值：

(1) to close:

```
{
    "type": "关闭",
    "value": None
}
```

(2) to manual, no auth:

```
{
    'type': '手动',
    'value': {
        'server': '192.168.31.47',
        'port': '8081',
        'authenticate': '0',
        'authUser': None,
        'authPassword': None
    }
}
```

(3) to manual, with auth:

```
{
    'type': '手动',
    'value': {
        'server': '192.168.31.47',
        'port': '8081',
        'authenticate': '1'
        'authUser': 'user',
        'authPassword': 'password'
    }
}
```

(4) to auto:

```
{
    'type': '自动',
    'value': {
        'url': 'your_auto_proxy_url'
    }
}
```

其中内部调用到的函数是：

获取 手动 时的值

```
def getManualProxyValue(self, parentTableSoup):
    """in 配置代理 page, from parent table soup, find 手动 pr

    Args:
        parentTableSoup (soup): parent table soup
    Returns:
        dict
    Raises:
    """
    manualProxyValue = None
    """
        设置 无线局域网 配置代理 手动:
            <XCUIElementTypeTable type="XCUIElementTypeTab
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeSwitch type="XCUIElemen
                </XCUIElementTypeCell>
            </XCUIElementTypeTable>

        设置 无线局域网 配置代理 手动 开启 鉴定:
            <XCUIElementTypeTable type="XCUIElementTypeTab
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
```

```
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeSwitch type="XCUIEleme
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeSecureTextField type="
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
            </XCUIElementTypeTable>
    """
    # proxyServer = None
    # proxyPort = None
    # proxyAuthenticate = None
    # <XCUIElementTypeTextField type="XCUIElementTypeTextF
    proxyServerSoup = parentTableSoup.find(
        'XCUIElementTypeTextField',
        attrs={"type": "XCUIElementTypeTextField", "name":
    )
```

```
        if not proxyServerSoup:
            return manualProxyValue
        proxyServer = proxyServerSoup.attrs.get("value", None)

        # <XCUIElementTypeTextField type="XCUIElementTypeTextF
        proxyPortSoup = parentTableSoup.find(
            'XCUIElementTypeTextField',
            attrs={"type": "XCUIElementTypeTextField", "name":
        )
        if not proxyPortSoup:
            return manualProxyValue
        proxyPort = proxyPortSoup.attrs.get("value", None) # '8

        # <XCUIElementTypeSwitch type="XCUIElementTypeSwitch" v
        proxyAuthenticateSoup = parentTableSoup.find(
            'XCUIElementTypeSwitch',
            attrs={"type": "XCUIElementTypeSwitch", "name": "鉴
        )
        if not proxyAuthenticateSoup:
            return manualProxyValue
        proxyAuthenticate = proxyAuthenticateSoup.attrs.get("va

        authUser = None
        authPassword = None

        if proxyAuthenticate == "1":
            # need save user and password
            # <XCUIElementTypeTextField type="XCUIElementTypeTe
            authUserSoup = parentTableSoup.find(
                'XCUIElementTypeTextField',
                attrs={"type": "XCUIElementTypeTextField", "nam
            )
            if not authUserSoup:
                return manualProxyValue
            authUser = authUserSoup.attrs.get("value", None) #

            # <XCUIElementTypeSecureTextField type="XCUIElement
            authPasswordSoup = parentTableSoup.find(
                'XCUIElementTypeSecureTextField',
                attrs={"type": "XCUIElementTypeSecureTextField"
            )
            if not authPasswordSoup:
                return manualProxyValue
            authPassword = authPasswordSoup.attrs.get("value",
            if '•' in authPassword:
                logging.warning("Get proxy autheticate passwor

        manualProxyValue = {
            "server": proxyServer,
            "port": proxyPort,
            "authenticate": proxyAuthenticate,
```

```
        "authUser": authUser,
        "authPassword": authPassword,
    }
    logging.info("manualProxyValue=%s", manualProxyValue)
    # manualProxyValue={'server': '192.168.31.46', 'port':
    # manualProxyValue={'server': '192.168.31.46', 'port':
    return manualProxyValue
```

获取 自动时的值：

页面：


蜗牛移动
15:57
guest_5G
配置代理
存储
关闭
手动
自动
URL
your_auto_proxy_url

```
def getAutoProxyValue(self, parentTableSoup):
    """in 配置代理 page, from parent table soup, find 自动 pr

    Args:
        parentTableSoup (soup): parent table soup
    Returns:
        dict
    Raises:
    """
    autoProxyValue = None
    """
        设置 无线局域网 配置代理 自动:
            <XCUIElementTypeTable type="XCUIElementTypeTab
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTextField type="XCUIEl
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
            </XCUIElementTypeTable>
    """
    autoUrlSoup = parentTableSoup.find(
        'XCUIElementTypeTextField',
        attrs={"type": "XCUIElementTypeTextField", "name":
    )
    if not autoUrlSoup:
        return autoProxyValue
    autoProxyValue = autoUrlSoup.attrs.get("value", None) #
    return autoProxyValue
```

点击去切换类型：

```
def switchToProxyType(self, parentTableSoup, newProxyTypeNa
    """in 配置代理 page, switch to new proxy type

    Args:
        parentTableSoup (soup): parent table soup
        newProxyTypeName (str): new proxy type name
    Returns:
        bool
    Raises:
    """
    isSwitchOk = False
    """
        设置 无线局域网 配置代理 手动:
            <XCUIElementTypeTable type="XCUIElementTypeTab
                <XCUIElementTypeCell type="XCUIElementType(
                    ...
                    <XCUIElementTypeStaticText type="XCUIE


                    ...
                    <XCUIElementTypeStaticText type="XCUIE


                    ...
                    <XCUIElementTypeStaticText type="XCUIE
    """
    newProxySoup = parentTableSoup.find(
        'XCUIElementTypeStaticText',
        attrs={"type": "XCUIElementTypeStaticText", "value"
    )
    if newProxySoup:
        clickedNewProxy = self.clickElementCenterPosition(n
        if clickedNewProxy:
            isSwitchOk = True
    return isSwitchOk
```

比如从 手动 切换到 关闭：

然后去恢复设置对应的值：

设置 手动 时的值：

```
def setManualProxyValue(self, parentTableSoup, newManualPro
    """in 配置代理 page, after changed to 手动
        set new manual proxy value

    Args:
        parentTableSoup (soup): parent table soup
        newManualProxyValue (dict): new manual proxy value
    Returns:
        bool
    Raises:
    """
    isUpateManualOk = False
    """
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =

    newServerValue = newManualProxyValue["server"]
    serverFieldQuery = {"type":"XCUIElementTypeTextField",
    serverFieldQuery["parent_class_chains"] = [ parentCell
    # isFoundServer, respInfo = self.findElement(query=serv
    # if not isFoundServer:
    #     return False
    isInputServerOk = self.wait_element_setText_iOS(serverF
    if not isInputServerOk:
        return False

    """
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    newPortValue = newManualProxyValue["port"]
    portFieldQuery = {"type":"XCUIElementTypeTextField", "n
    portFieldQuery["parent_class_chains"] = [ parentCellCla
    isInputPortOk = self.wait_element_setText_iOS(portFiel
    if not isInputPortOk:
        return False

    """
                <XCUIElementTypeCell type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
```

```
                    <XCUIElementTypeSwitch type="XCUIEleme
                </XCUIElementTypeCell>
    """
    newAuthenticateValue = newManualProxyValue["authenticat
    authSwitchQuery = {"type":"XCUIElementTypeSwitch", "nam
    authSwitchQuery["parent_class_chains"] = [ parentCellC
    foundAuth, respInfo = self.findElement(authSwitchQuery,
    if not foundAuth:
        return False

    authSwitchElement = respInfo

    curAuthValueStr = ""
    # curAuthValue = authSwitchElement.value # '0'
    # curAuthValueStr = str(curAuthValue)
    # Special: sometime wda element value is WRONG, actual
    # so change to bs find then get value from page source
    curPageXml = self.get_page_source()
    soup = CommonUtils.xmlToSoup(curPageXml)
    authSwitchChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeSwitch",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    authSwitchSoup = CommonUtils.bsChainFind(soup, authSwit
    if authSwitchSoup:
        curAuthValue =  authSwitchSoup.attrs.get("value", N
        if curAuthValue:
            curAuthValueStr = str(curAuthValue)

    if curAuthValueStr == "":
        return False

    if curAuthValueStr != newAuthenticateValue:
        # click switch element to change value
        isClickOk = self.clickElement(authSwitchElement)
        if not isClickOk:
            return False

    if newAuthenticateValue == "1":
        # need restore auth user and password
        newAuthUserValue = newManualProxyValue["authUser"]
```

```
        userFieldQuery = {"type":"XCUIElementTypeTextField"
        userFieldQuery["parent_class_chains"] = [ parentCe
        isInputUserOk = self.wait_element_setText_iOS(userI
        if not isInputUserOk:
            return False

        newAuthPasswordValue = newManualProxyValue["authPas
        passwordFieldQuery = {"type":"XCUIElementTypeSecure
        passwordFieldQuery["parent_class_chains"] = [ parer
        isInputPasswordOk = self.wait_element_setText_iOS(p
        if not isInputPasswordOk:
            return False

    isUpateManualOk = True
    return isUpateManualOk
```

页面：

支持 开启 鉴定 的值的恢复：


蜗牛移动
10:23
_guest_5G
配置代理
存储
关闭
手动
自动
服务器
192.168.31.46
端口
8081
鉴定
用户名
user
密码
您的凭证可能以不加密方式发送

设置 自动 时的值：

```
def setAutoProxyValue(self, parentTableSoup, newAutoProxyVa
    """in 配置代理 page, after changed to 自动
        set new manual proxy value
            by click each item then set value

    Args:
        parentTableSoup (soup): parent table soup
        newAutoProxyValue (dict): new auto proxy value dict
    Returns:
        bool
    Raises:
    """
    isUpdateAutoOk = False
    """
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTextField type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    newUrlValue = newAutoProxyValue["url"]
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =
    urlFieldQuery = {"type":"XCUIElementTypeTextField", "na
    urlFieldQuery["parent_class_chains"] = [ parentCellClas
    # foundUrl, respInfo = self.findElement(urlFieldQuery,
    # if not foundUrl:
    #     return False
    isInputUrlOk = self.wait_element_setText_iOS(urlFieldQu
    isUpdateAutoOk = isInputUrlOk
    return isUpdateAutoOk
```

当有变化后，右上角的存储

比如从 关闭 切换到 手动，且已填写完 手动时的配置后，右上角存储是蓝色 可以点击：


App Store
18:39
guest_5G
配置代理
存储
关闭
手动
自动
服务器
192.168.31.46
端口
8081
鉴定
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0

可以点击去保存改动：

```
def storeChangedProxyType(self):
    """in 配置代理 page, save changed proxy type, by click n

    Args:
        parentTableSoup (soup): parent table soup
        newProxyTypeName (str): new proxy type name
    Returns:
        bool
    Raises:
    """
    isStoredOk = False
    """
        设置 WiFi 配置代理 从手动切换到 关闭 存储:
            <XCUIElementTypeNavigationBar type="XCUIElement
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeNavigationBar>
    """
    storeName = "存储"
    parentNaviBarClassChain = "/XCUIElementTypeNavigationBa
    storeButtonQuery = {"type":"XCUIElementTypeButton", "na
    storeButtonQuery["parent_class_chains"] = [ parentNaviB
    foundAndClickedStore = self.findAndClickElement(query=s
    isStoredOk = foundAndClickedStore
    return isStoredOk
```

即可。

AppStore

此处整理用wda自动化操作AppStore的相关常见代码段。

自动安装app

此处整理用wda通过AppStore自动安装iOS的app的完整过程。

-》iPhone中用AppStore自动安装iOS的app的全过程

详见：

【已解决】iOS自动抓包app：iPhone中通过AppStore自动安装iOS的app

公共函数

下面函数中用到的公共函数，比如：

- `get_cmd_lines`
- `multipleRetry`
- `findElement`
- `findAndClickElement`

详见其他部分：

- 常用代码段
- 元素处理

过程

对应着的界面分别是：

启动AppStore后的，本身默认进入了Search的tab页：

当然，代码中防止不是默认搜索Tab，也用代码去切换到此tab页了。

然后再去点击搜索框，输入要搜索的app名字：

点击搜索后，进入搜索结果列表页：

然后找到列表页中第一个匹配的后，点击进入 app下载详情页：

注意：

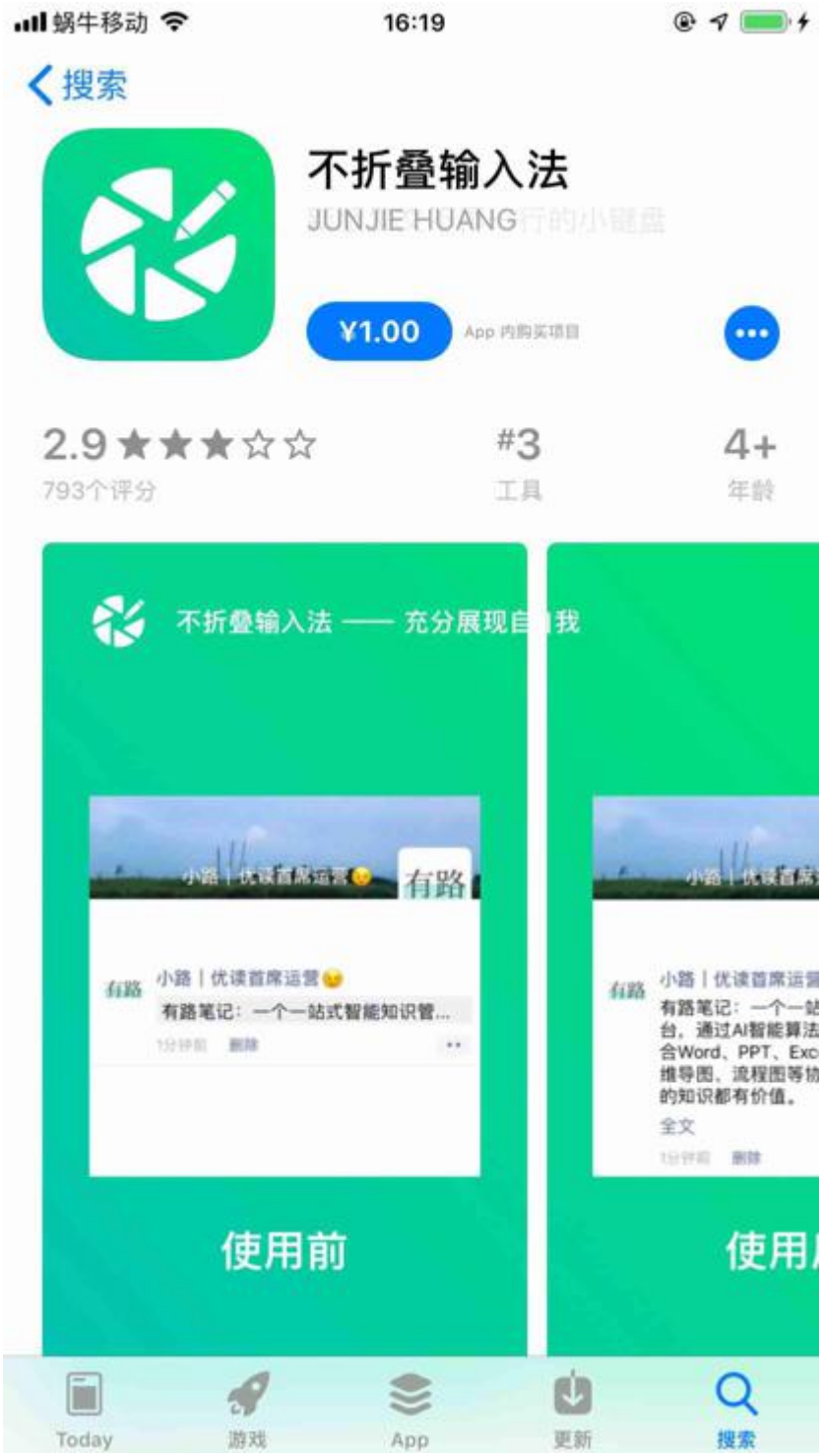
此处的图标是 云朵☁️ 中间有个⬇️ 向下的箭头 表示 **重新下载**

意味着你的apple账号所登录过的iPhone中之前别处已经下载过该app了

如果是全新的没有下载过的app，则前面按钮显示的是**获取**

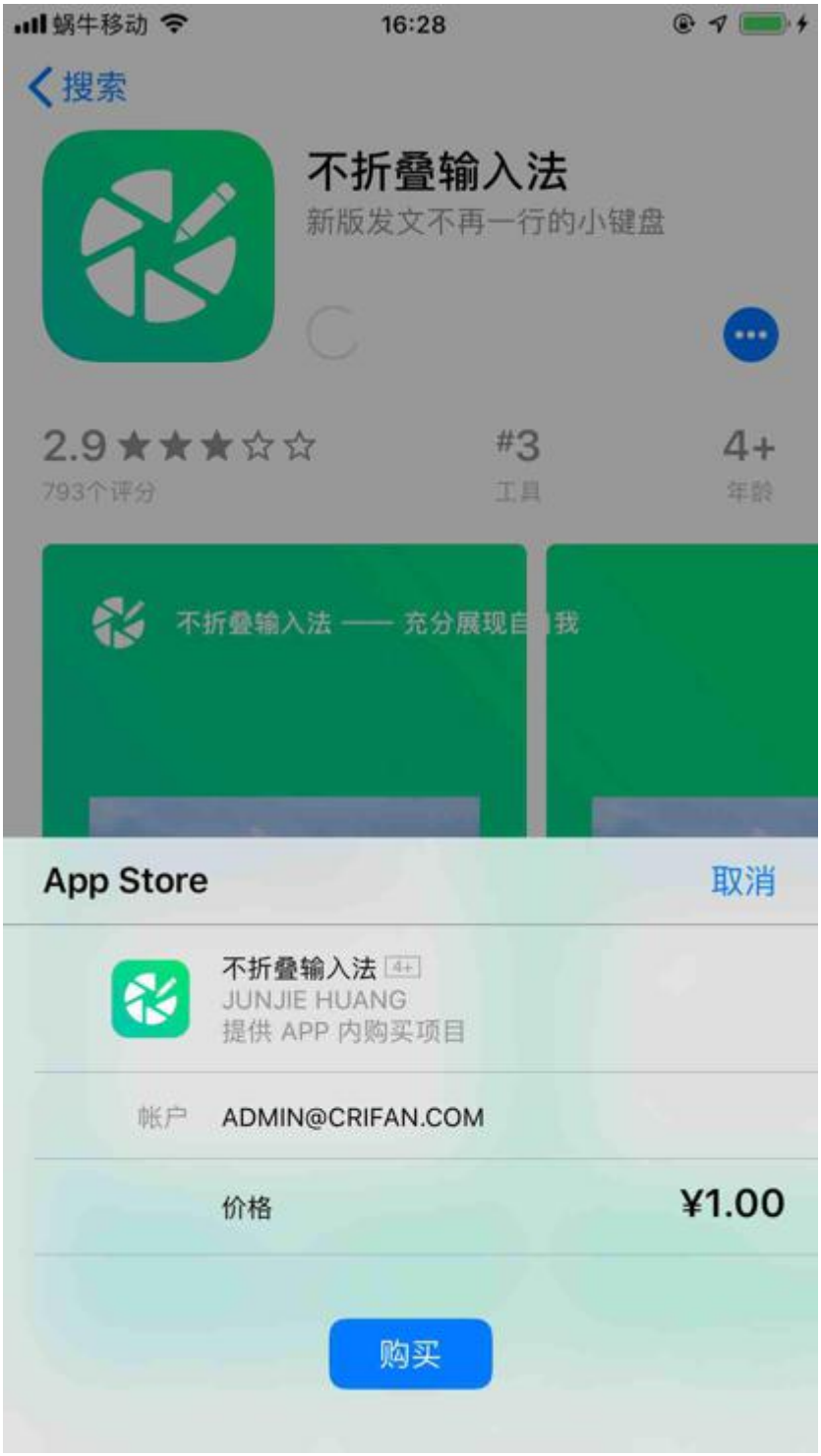
另外：

如果是付费的app，则显示的是金额：



对此：代码中，检测到是付费后，提示 不支持。

除非真的打算付费，否则点击后，弹框购买：



点击购买后，会真的扣费的。

点击下载按钮后：

情况1：对于 （之前已下载过的app）重新下载，则无弹框

往往会出现加载中：

情况2：对于 （全新app）获取，会弹框 安装

（借用别的app的弹框 展示）


蜗牛移动
16:57
搜索
淘宝
Taobao
手机淘宝 - 淘到你说好
随时随地，想淘就淘
4.4
86.3万个评分
#3
购物
17+
年龄
App Store
取消
手机淘宝 - 淘到你说好
17+
TAOBAO (CHINA) SOFTWARE CO.,LTD
APP
帐户
ADMIN@CRIFAN.COM
安装

点击安装，弹框会提示 完成：

（其他app的截图）

之后就是下载过程了：

正在下载 其中圆圈⭕ 会有个蓝色进度条：

解释：

此处通过调试log日志：

```
[200608 14:22:43][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:44][DevicesMethods.py 1658] Cost 1.57 seconds
[200608 14:22:45][DevicesMethods.py 2238] Downloading 京东
[200608 14:22:45][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:45][DevicesMethods.py 1658] Cost 0.73 seconds
[200608 14:22:45][DevicesMethods.py 2238] Downloading 京东
[200608 14:22:45][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:46][DevicesMethods.py 1658] Cost 0.74 seconds
[200608 14:22:46][DevicesMethods.py 2238] Downloading 京东
[200608 14:22:46][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:47][DevicesMethods.py 1658] Cost 0.85 seconds
[200608 14:22:47][DevicesMethods.py 2238] Downloading 京东
[200608 14:22:47][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:48][DevicesMethods.py 1658] Cost 0.80 seconds
[200608 14:22:48][DevicesMethods.py 2238] Downloading 京东
[200608 14:22:48][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:48][DevicesMethods.py 1658] Cost 0.76 seconds
[200608 14:22:48][DevicesMethods.py 2238] Downloading 京东
[200608 14:22:48][DevicesMethods.py 1645] start get iOS pag
[200608 14:22:49][DevicesMethods.py 1658] Cost 0.78 seconds
[200608 14:22:49][DevicesMethods.py 2238] Downloading 京东
。。。
[200608 14:23:08][DevicesMethods.py 2238] Downloading 京东
[200608 14:23:08][DevicesMethods.py 1645] start get iOS pag
[200608 14:23:09][DevicesMethods.py 1658] Cost 0.70 seconds
[200608 14:23:09][DevicesMethods.py 2238] Downloading 京东
```

多次调试发现：

在下载的进度超过76%之后，感觉内部就进入了 自动安装 过程

等安装完毕后，进度立刻就是100%，按钮变成 后续的 打开，表示安装完成了

-》即，推断是整体进度：

- 0~76%：下载进度
- 76%之后：安装进度
  - 但是不会显示，会从76%直接跳到 打开
    - 表示安装完毕

最终安装完成后，显示 `打开` ：

如果点击打开，即可启动app。

最后附上，部分测试后的iOS的app安装后桌面图标：

详见：

【已解决】iOS自动化：通过AppStore自动下载和安装iOS的app京东

代码

install_app_iOS 安装iOS的app

文件：`src/common/DevicesMethods.py`

```
def install_app_iOS(self, item, packages=None):
    """install iOS app
        if install ok, update bundleId for input item
        not install if found already installed
    """
    isInstallOk = False
    appInfo = None

    # {'account': 'bb62512466_米家', 'bundleId': 'com.xiaom
    appName = item["name"]

    # for debug
    # appName = "京东"
    # appName = "斑马AI课"

    # check if already installed
    appInfo = self.getInstalledAppInfo(appName=appName)
    if appInfo:
        isInstallOk = True
        logging.warning("Not install %s for already instal
    else:
        # auto install app from AppStore
        isInstallOk, respInfo = self.iOSinstallAppFromAppSt
        logging.debug("appName=%s -> isInstallOk=%s, respIn
        # {'bundleId': 'com.fenbi.ape.zebstrika', 'name':
        if isInstallOk:
            appInfo = respInfo
        else:
            errMsg = respInfo
            logging.error("Fail to auto install iOS app %s,

    if appInfo:
        # update bundle id
        installedBundleId = appInfo["bundleId"] # 'com.360b
        item["bundleId"] = installedBundleId
    return isInstallOk
```

iOSinstallAppFromAppStore 从AppStore中自动安装iOS的app

```
def iOSinstallAppFromAppStore(self, appName):
    """Install iOS app from AppStore

    Args:
        appName (str): app name
    Returns:
        bool, dict/str
            bool: installed ok or not
                if true:
                    dict: installed app info
                if false:
                    str: fail reason error message
    Raises:
    """
    isInstallOk = False
    respInfo = None

    iOS_AppId_AppStore = "com.apple.AppStore"
    appStoreSession = self.wdaClient.session(iOS_AppId_AppS
    logging.debug("appStoreSession=%s" % appStoreSession)

    isSwitchOk = CommonUtils.multipleRetry(
        {"functionCallback": self.switchToAppStoreSearchTab
        maxRetryNum = 10,
        sleepInterval = 0.5,
    )
    if not isSwitchOk:
        respInfo = "Fail to switch to Search tab of AppStor
        return isInstallOk, respInfo

    """
        <XCUIElementTypeOther type="XCUIElementTypeOther" e
            <XCUIElementTypeSearchField type="XCUIElementTy
        </XCUIElementTypeOther>
    """
    searchInputQuery = {"type":"XCUIElementTypeSearchField"
    isInputOk = CommonUtils.multipleRetry(
        {
            "functionCallback": self.wait_element_setText,
            "functionParaDict": {
                "locator": searchInputQuery,
                "text": appName,
            }
        }
    )
    if not isInputOk:
        respInfo = "Fail to input app name %s into search t
        return isInstallOk, respInfo

    isSearchOk = self.search_iOS(wait=0.2)
```

```
        if not isSearchOk:
            respInfo = "Fail to find and click Search button to
            return isInstallOk, respInfo

        # Special: try add some wait time to avoid some specia
        # for 个人所得税 search result page, found and click 个人
        time.sleep(0.5)
        isIntoDetailOk = CommonUtils.multipleRetry(
            {
                "functionCallback": self.appStoreSearchResultIn
                "functionParaDict": {
                    "appName": appName,
                }
            },
            maxRetryNum = 10,
            sleepInterval = 0.5,
        )
        if not isIntoDetailOk:
            respInfo = "Fail to into app detail page for %s" %
            return isInstallOk, respInfo
        # Special: try add some wait time to avoid some specia
        # for 个人所得税 search result page, found and click 个人
        time.sleep(0.2)

        detectRoundNum = 0
        while True:
            detectRoundNum += 1
            logging.info("%s try auto install, round [%d] %s",

            # foundOpen = False
            # isDownloading = False
            # isLoading = False
            # foundAndClickedPopupInstall = False
            # foundAndClickedDownload = False

            # isDownloading, curProgress = self.appStoreDownloa
            # logging.info("isDownloading=%s, curProgress=%s",
            isDownloading = self.appStoreDownloading()
            logging.info("isDownloading=%s", isDownloading)
            if isDownloading:
                # downloadingWaitTime = 0.5
                downloadingWaitTime = 1.0
                time.sleep(downloadingWaitTime)
                logging.info("Is downloading, wait %s seconds",
                continue

            foundOpen = self.appStoreCheckOpen()
            logging.info("foundOpen=%s", foundOpen)
            if foundOpen:
                logging.info("Found 打开 -> means %s is install
                break
```

```
        isLoading = self.appStoreLoading()
        logging.info("isLoading=%s", isLoading)
        if isLoading:
            # loadingWaitTime = 0.2
            loadingWaitTime = 0.5
            time.sleep(loadingWaitTime)
            logging.info("Is loading, wait %s seconds", loa
            continue

        foundAndClickedDownload, downloadButtonName = self.
        logging.info("foundAndClickedDownload=%s, downloadB
        if foundAndClickedDownload:
            logging.info("Found and clicked button %s to st
        else:
            foundMoneyButton, moneyButtonName = self.appSto
            if foundMoneyButton:
                respInfo = "Not support auto install %s for
                return isInstallOk, respInfo

        foundAndClickedPopupInstall = self.appStoreClickPop
        logging.info("foundAndClickedPopupInstall=%s", foun
        if foundAndClickedPopupInstall:
            popupInstallWaitTime = 0.2
            time.sleep(popupInstallWaitTime)
            logging.info("After click 安装 of popup, wait %

    logging.info("Install complete for %s", appName)
    installedAppInfo = self.getInstalledAppInfo(appName=app
    logging.info("installedAppInfo=%s", installedAppInfo)
    if not installedAppInfo:
        respInfo = "Fail to extract installed app info for
        return isInstallOk, respInfo

    isInstallOk = True
    respInfo = installedAppInfo
    logging.info("isInstallOk=%s, respInfo=%s", isInstallOk
    return isInstallOk, respInfo
```

其他相关函数：

getInstalledAppInfo 获取已安装的app的信息

```
def getInstalledAppInfo(self, appName=None, appBundleId=Nor
    """find app info from app name or app bundle id

    Args:
        appName (str): iOS app name
        appBundleId (str): iOS app bundle id
    Returns:
        dict:
            app info
                eg: {'bundleId': 'com.360buy.jdmobile', 'na
            None if not found
    Raises:
    """
    appInfo = None
    installedAppList = self.get_iOS_installedAppList()
    for eachAppInfo in installedAppList:
        eachAppName = eachAppInfo["name"]
        eachAppBundleId = eachAppInfo["bundleId"]
        if appName:
            if eachAppName == appName:
                appInfo = eachAppInfo
                break

        if appBundleId:
            if eachAppBundleId == appBundleId:
                appInfo = eachAppInfo
                break

    return appInfo
```

get_iOS_installedAppList 获取已安装app的列表信息

```
def get_iOS_installedAppList(self):
    installedAppList = []
    listAppCmd = 'ideviceinstaller -l'
    appListStr = CommonUtils.get_cmd_lines(listAppCmd, text
    logging.debug("appListStr=%s", appListStr)
    if appListStr:
        appRawList = appListStr.split("\n")
        """
            Total: 9 apps
            com.dianping.dpscope - 大众点评 10.27.10.21
            com.tencent.xin - 微信 7.0.12.33
            com.tencent.tiantianptu - 天天P图 603040
            com.didapinche.taxi - 嘀嗒出行 3
            com.luojilab.LuoJiFM-IOS - 得到 7.10.361
            com.suiyi.foodshop1 - 食行生鲜 49267
            com.alipay.iphoneclient - 支付宝 10.1.90.8000
            com.crifan.WebDriverAgentRunner.xctrunner - Web
            com.xiaojukeji.didi - 滴滴出行 5.4.10.904142127
        """
        for eachAppStr in appRawList:
            eachAppStr = eachAppStr.strip()
            # foundApp = re.search("(?P<bundleId>com\.\S+)\
            # rn.notes.best - 爱思极速版 11122019
            # foundApp = re.search("(?P<bundleId>[\w\.]+)\s
            # 'com.kingsoft.www.office.wpsoffice - WPS Offi
            foundApp = re.search("(?P<bundleId>[\w\.]+)\s+-
            if foundApp:
                bundleId = foundApp.group("bundleId") # 'co
                name = foundApp.group("name") # '大众点评'
                version = foundApp.group("version") # '10.2
                curAppInfo = {
                    "bundleId": bundleId,
                    "name": name,
                    "version": version,
                }
                installedAppList.append(curAppInfo)
            else:
                # Total: 9 apps
                if eachAppStr and (not eachAppStr.startswit
                    logging.error("not match installed app
    logging.debug("installedAppList=%s", installedAppList)
    return installedAppList
```

switchToAppStoreSearchTab 切换到AppStore的Search的tab页

```
def switchToAppStoreSearchTab(self):
    """try find and click to switch to AppStore search tab'
    isSwitchOk = False
    """
        AppStore 底部tab:
            <XCUIElementTypeTabBar type="XCUIElementTypeTab
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
                <XCUIElementTypeButton type="XCUIElementTyp
            </XCUIElementTypeTabBar>
    """
    parentTabBarClassChain = "/XCUIElementTypeTabBar[`rect
    searchButtonQuery = {"type":"XCUIElementTypeButton", "
    searchButtonQuery["parent_class_chains"] = [ parentTabB
    foundAndClicked = self.findAndClickElement(query=searcl
    isSwitchOk = foundAndClicked
    return isSwitchOk
```

wait_element_setText_iOS 给元素输入值并等待一段时间

```
def wait_element_setText_iOS(self, query, text):
    isInputOk = False
    isFound, respInfo = self.findElement(query=query)
    logging.debug("isFound=%s, respInfo=%s", isFound, respI
    if isFound:
        searchAccountElement = respInfo
        searchAccountElement.set_text(text)
        logging.info("has input text: %s", text)
        isInputOk = True
    return isInputOk
```

search_iOS iOS中点击弹出键盘中的Search触发搜索

```
def search_iOS(self, wait=1):
    # 触发点击搜索按钮
    foundAndClickedDoSearch = False
    # <XCUIElementTypeButton type="XCUIElementTypeButton" r
    # <XCUIElementTypeButton type="XCUIElementTypeButton" r
    # searchButtonQuery = {"name": "Search"}
    searchButtonQuery = {"name": "Search", "type": "XCUIEle
    # Note: occasionally not found Search, change to find r
    MaxRetryNumber = 5
    curRetryNumber = MaxRetryNumber
    while curRetryNumber > 0:
        foundAndClickedDoSearch = self.findAndClickElement(
        if foundAndClickedDoSearch:
            break

        curRetryNumber -= 1

    if not foundAndClickedDoSearch:
        logging.error("Not found and/or clicked for %s", se
    return foundAndClickedDoSearch
```

appStoreSearchResultIntoDetail AppStore从搜索结果页中进去详情页

```
def appStoreSearchResultIntoDetail(self, appName):
    """for AppStore search result list page
        try find first match result
            then click into detail page

    Args:
        appName (str): app name
    Returns:
        bool, dict
            bool: is into detail page or not
    Raises:
    """
    isIntoDetailOk = False
    """
        搜索结果列表页 京东 重新下载:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeCollectionView type="XC
                        <XCUIElementTypeCell type="XCUIEler
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIEler
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                    </XCUIElementTypeCollectionView>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        搜索结果列表页 美团 获取:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeCollectionView type="XC
                        <XCUIElementTypeCell type="XCUIEler
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIEler
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeCell>
                    </XCUIElementTypeCollectionView>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    parentCollectionViewClassChain = "/XCUIElementTypeColle
    firstMatchCellQuery = {"type":"XCUIElementTypeCell", "r
    firstMatchCellQuery["parent_class_chains"] = [ parentCo
    foundAndClicked = self.findAndClickElement(query=firstM
    isIntoDetailOk = foundAndClicked
    return isIntoDetailOk
```

appStoreDownloading AppStore中是否是正在下载

```
# def appStoreDownloadingProgress(self):
def appStoreDownloading(self):
    """Detect app store is downloading some app

    Args:
    Returns:
        bool: true for found is downloading
    Raises:
    """
    isDownloading = False
    # curProgress = ""
    """
        AppStore 详情页 京东 正在下载:
            <XCUIElementTypeCollectionView type="XCUIElemer
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    # Note: here change wda query to bs.find, then revert b
    # for later bs.find will need get page source, which to

    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =
    downloadingButtonQuery = {"type":"XCUIElementTypeButton
    downloadingButtonQuery["parent_class_chains"] = [ parer
    isfound, respInfo = self.findElement(query=downloadingB
    # if isfound:
    #     isDownloading = isfound
    #     curElement = respInfo
    #     curValue = curElement.value # always get null
    #     if curValue is not None:
    #         curProgress = curValue
    isDownloading = isfound

    # # Special: above wda query find element, get value, b
    # # so change to bs find
    # curPageXml = self.get_page_source()
    # soup = CommonUtils.xmlToSoup(curPageXml)
    # isDownloadingChainList = [
    #     {
    #         "tag": "XCUIElementTypeCollectionView",
    #         "attrs": self.FullScreenAttrDict
    #     },
```

```
    #     {
    #         "tag": "XCUIElementTypeCell",
    #         "attrs": {"enabled":"true", "visible":"true",
    #     },
    #     {
    #         "tag": "XCUIElementTypeButton",
    #         "attrs": {"enabled":"true", "visible":"true",
    #     },
    # ]
    # isDownloadingSoup = CommonUtils.bsChainFind(soup, isD
    # if isDownloadingSoup:
    #     isDownloading = True
    #     soupAttrDict = isDownloadingSoup.attrs
    #     curValue = soupAttrDict.get("value", "") # '0%'
    #     curProgress = curValue
    # return isDownloading, curProgress

    return isDownloading
```

appStoreCheckOpen 检测AppStore是否是下载完毕可以找到打开

```
def appStoreCheckOpen(self):
    """Detect whether app store is downloading compelete th

    Args:
    Returns:
        bool
    Raises:
    """
    foundOpen = False
    """
        AppStore 详情页 京东 打开:
            <XCUIElementTypeCollectionView type="XCUIElemer
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =
    openButtonQuery = {"type":"XCUIElementTypeButton", "nar
    openButtonQuery["parent_class_chains"] = [ parentCellC
    foundOpen, respInfo = self.findElement(query=openButtor
    return foundOpen
```

appStoreLoading 检测AppStore中是否是正在载入

```
def appStoreLoading(self):
    """after click start download button,  check app store

    Args:
    Returns:
        bool
    Raises:
    """
    foundLoading = False
    """
        AppStore 详情页 京东 正在载入:
            <XCUIElementTypeCollectionView type="XCUIElemer
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =
    loadingButtonQuery = {"type":"XCUIElementTypeButton", "
    loadingButtonQuery["parent_class_chains"] = [ parentCe
    foundLoading, respInfo = self.findElement(query=loading
    return foundLoading
```

appStoreStartDownload 找到重新下载或获取等按钮并点击开始下载

```
def appStoreStartDownload(self, isShowErrWhenNotFound=True)
    """being in app detail page inside AppStore
        try find and click download button

    Args:
        isShowErrWhenNotFound (bool): show error log when n
    Returns:
        bool, str:
            boo: found and clicked download button
            str: button name
    Raises:
    """
    # curPageXml = self.get_page_source()
    # soup = CommonUtils.xmlToSoup(curPageXml)
    foundAndClicked = False
    downloadButtonName = ""
    """
        AppStore 详情页 京东 重新下载:
            <XCUIElementTypeCollectionView type="XCUIElemen
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
                <XCUIElementTypeCell type="XCUIElementType(

        AppStore 详情页 钉钉 获取:
            <XCUIElementTypeCollectionView type="XCUIElemen
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    # downloadP = re.compile("(获取)|(重新下载)")
```

```
    # downloadChainList = [
    #     {
    #         "tag": "XCUIElementTypeCollectionView",
    #         "attrs": self.FullScreenAttrDict
    #     },
    #     {
    #         "tag": "XCUIElementTypeCell",
    #         "attrs": {"enabled":"true", "visible":"true",
    #     },
    #     {
    #         "tag": "XCUIElementTypeButton",
    #         "attrs": {"enabled":"true", "visible":"true",
    #     },
    # ]
    # downloadSoup = CommonUtils.bsChainFind(soup, downloa
    # if downloadSoup:
    #     self.clickElementCenterPosition(downloadSoup)
    #     foundAndClickedDownload = True
    # return foundAndClickedDownload

    # Note: to avoid possible later get page slow or even
    # change to wda query
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =
    downloadButtonNameList = ["获取", "重新下載"]
    for eachDownloadButtonName in downloadButtonNameList:
        curDownloadButtonQuery = {"type":"XCUIElementTypeBu
        curDownloadButtonQuery["parent_class_chains"] = [ p
        foundAndClicked = self.findAndClickElement(query=cu
        if foundAndClicked:
            downloadButtonName = eachDownloadButtonName
            break
    return foundAndClicked, downloadButtonName
```

appStoreBuyAppMoneyButton 检测AppStore中是否有￥金额，表示是收费应用

```
def appStoreBuyAppMoneyButton(self):
    """being in app detail page inside AppStore
        try find buy app button which has money text like

    Args:
    Returns:
        bool, str:
            boo: found and clicked download button
            str: button name
    Raises:
    """
    foundMoneyButton = False
    moneyButtonName = ""
    """
        AppStore 详情页 不折叠输入法 带金额的购买按钮 ¥1.00:
            <XCUIElementTypeCollectionView type="XCUIElemer
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeCell type="XCUIElementType(
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeCell>
    """
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x =
    moneyButtonQuery = {"type":"XCUIElementTypeButton", "na
    moneyButtonQuery["parent_class_chains"] = [ parentCell(
    foundMoneyButton, respInfo = self.findElement(query=mor
    if foundMoneyButton:
        curElement = respInfo
        # internally use first non-empty of element.wdValue
        # curName = curElement.text
        # so change to name, Note: has changed facebook-wda
        curName = curElement.name
        if curName is not None:
            moneyButtonName = curName
    return foundMoneyButton, moneyButtonName
```

appStoreClickPopupInstall AppStore中点击弹框开始安装

```
def appStoreClickPopupInstall(self):
    """after click start download button, try find popup in

    Args:
    Returns:
        bool
    Raises:
    """

    """
        AppStore 详情页 淘宝 弹框 安装:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeTable type="XCUIElementType
                    <XCUIElementTypeCell type="XCUIElement"
                        <XCUIElementTypeOther type="XCUIEle
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeStaticText type="XC
                    </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElement"
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                    </XCUIElementTypeCell>
                </XCUIElementTypeTable>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    parentOtherClassChain = "/XCUIElementTypeOther[`rect.x
    installButtonQuery = {"type":"XCUIElementTypeButton", "
    installButtonQuery["parent_class_chains"] = [ parentOth
    foundAndClickedInstall = self.findAndClickElement(query
    return foundAndClickedInstall
```

iOS弹框处理

背景：iOS的各种app，有各种类型的弹框，目前主要是通过 wda的query和 获取源码后bs的find找到过滤，再去点击让弹框消失 其中此处分成了2部分：

- 首次启动后，可能会遇到的弹框：`process_popup_iOS_FirstLaunchBeforeMain`
- 其他正常运行期间，可能会遇到的弹框：`process_popup_iOS`

process_popup_iOS_FirstLaunchBeforeMain

```
def process_popup_iOS_FirstLaunchBeforeMain(self, soup=None
    """
    处理iOS中app，在首次登录，进入主页之前，的弹框
    """
    foundAndProcessedPopup = False
    if not soup:
        curPageXml = self.get_page_source()
        soup = CommonUtils.xmlToSoup(curPageXml)

    # if not foundAndProcessedPopup:
    #     foundAndProcessedPopup = self.iOSProcessPopupUppe
    # if not foundAndProcessedPopup:
    #     foundAndProcessedPopup = self.iOSProcessPopupUppe
    # Note: put common upper right close, for some app, eg
    if not foundAndProcessedPopup:
        # foundAndProcessedPopup = self.iOSProcessPopupUppe
        foundAndProcessedPopup = self.iOSProcessPopupCommon

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupPossib

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupUnderC

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupIKnow(

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessCommonUserA

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessUserPrivacy

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessAlertUseWi

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessCommonAlert

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessCommonAlert

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessRightUpperJ

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupCommon

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupCommon
```

```
    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupImmed

    return foundAndProcessedPopup
```

process_popup_iOS

```
def process_popup_iOS(self, curPageXml):
    foundAndProcessedPopup = False

    soup = CommonUtils.xmlToSoup(curPageXml)

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.process_popup_iOS_Fi

    if self.isSpecialiOSApp_kags:
        if not foundAndProcessedPopup:
            foundAndProcessedPopup = self.iOSProcessKagsPo

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupPhone

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessSettingsAl

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupCerti

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupNetwo

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupNetwo

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupSyste

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupServe

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupWeixi

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupJumpT

    if not foundAndProcessedPopup:
        # foundAndProcessedPopup = self.iOSProcessPopupGet
        foundAndProcessedPopup = self.iOSProcessPopupDoSom

    if not foundAndProcessedPopup:
        # foundAndProcessedPopup = self.iOSProcessPopupMin
        foundAndProcessedPopup = self.iOSProcessPopupMinip

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupWeixi
```

```
    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupMinipr

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupMinipr

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessWillOpenNor

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessAlertNoteCo

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessAlertCancel

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessAlertCommor

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupPhotoC

    # 注：此逻辑经常误判，没有弹框误以为有弹框，所以放弃此逻辑
    # if not foundAndProcessedPopup:
    #     foundAndProcessedPopup = self.iOSProcessCommonPop

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupSovere

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupCommor

    return foundAndProcessedPopup
```

具体实现：

iOSProcessPopupImmediatelyExperience

```
def iOSProcessPopupImmediatelyExperience(self, soup):
    """
        弹框 other->other-button name=立即体验
    """

    """
        米家 弹框 立即体验:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    immediatelyExperienceChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    foundAndProcessedPopup = self.findAndClickCenterPositio
    return foundAndProcessedPopup
```

iOSProcessPopupCommonConfirmButton

```
def iOSProcessPopupCommonConfirmButton(self, soup):
    """
        other下面other下的button name="确定"
    """
    foundAndProcessedPopup = False
    """
        米家 弹框 米家隐私政策 确定:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTextView type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIEleme
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    confirmChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true",
        },
    ]
    confirmSoup = CommonUtils.bsChainFind(soup, confirmCha
    if confirmSoup:
        self.clickElementCenterPosition(confirmSoup)
        foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessPopupCommonSave

```
def iOSProcessPopupCommonSave(self, soup):
    """
        通用的弹框 other下的other下的button name=保存
    """
    foundAndProcessedPopup = False
    """
        米家 弹框 地区选择 保存:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement

                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeImage type=
                                <XCUIElementTypeSearchFiel
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeOther type="XCU
                        </XCUIElementTypeOther>
                        <XCUIElementTypeTable type="XCUIEle

                        </XCUIElementTypeTable>
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
    """
    commonSaveChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    commonSaveSoup = CommonUtils.bsChainFind(soup, commonSa
    if commonSaveSoup:
        self.clickElementCenterPosition(commonSaveSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupCommonWindowCancel

```
def iOSProcessPopupCommonWindowCancel(self, soup):
    """
        通用的弹框 window下的other下的button name=取消
    """
    foundAndProcessedPopup = False
    """
        斑马AI课 弹框 该音频为专属音频 取消:
            <XCUIElementTypeWindow type="XCUIElementTypeWi
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEl
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>
    """
    commonCancelChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeOther",
            # "attrs": self.FullScreenAttrDict
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true",
        },
    ]
    commonCancelSoup = CommonUtils.bsChainFind(soup, commo
    if commonCancelSoup:
        self.clickElementCenterPosition(commonCancelSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupSovereignHasRead

```
def iOSProcessPopupSovereignHasRead(self, soup):
    """
        必要 弹框 朕已阅
    """
    foundAndProcessedPopup = False
    """
        必要 弹框 朕已阅:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeImage type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTextView type="XCUIEle
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeImage>
            </XCUIElementTypeOther>
    """
    hasReadChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeImage",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    hasReadCloseSoup = CommonUtils.bsChainFind(soup, hasRea
    if hasReadCloseSoup:
        self.clickElementCenterPosition(hasReadCloseSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessAlertCommonGiveup

```
def iOSProcessAlertCommonGiveup(self, soup):
    """
        通用逻辑：对于alert弹框，  找里面的 放弃 button，并点击
    """
    foundAndProcessedPopup = False
    """
        京东金融 弹框 是否放弃注册?
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementTyp
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEl
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEl
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>
    """
    giveupChainList = [
        {
```

```
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    alertGiveupSoup = CommonUtils.bsChainFind(soup, giveup(
    if alertGiveupSoup:
        self.clickElementCenterPosition(alertGiveupSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupServerBusyLaterRetry

```
def iOSProcessPopupServerBusyLaterRetry(self, soup):
    foundAndProcessedPopup = False
    """
        京东金融 异常页面 服务器繁忙，请稍后重试
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeCollectionView type="XC
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    busyRetryChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    busyRetrySoup = CommonUtils.bsChainFind(soup, busyRetry
    if busyRetrySoup:
        # find back button
        parentOtherSoup = busyRetrySoup.parent
        parentParentOtherSoup = parentOtherSoup.parent
        if parentParentOtherSoup:
            backupP = re.compile("backup") # "com icon back
            backupSoup = parentParentOtherSoup.find(
                "XCUIElementTypeButton",
                attrs={"visible":"true", "enabled":"true",
            )
            if backupSoup:
                self.clickElementCenterPosition(backupSoup)
                foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupSystemAbnormalRetryRefresh

```
def iOSProcessPopupSystemAbnormalRetryRefresh(self, soup):
    foundAndProcessedPopup = False
    """
        京东金融 异常页面 系统正在开小差，请稍后再试 再刷新下:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeStaticText type="XC
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    retryRefreshChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    retryRefreshSoup = CommonUtils.bsChainFind(soup, retryR
    if retryRefreshSoup:
        self.clickElementCenterPosition(retryRefreshSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupUnderCloseButton

```
def iOSProcessPopupUnderCloseButton(self, soup):
    """
        弹框 关闭按钮在弹框下面的 尤其是 name="□"
    """
    foundAndProcessedPopup = False
    """
        途虎养车 弹框 关闭按钮在下面 新人618全品类消费券:
            <XCUIElementTypeWindow type="XCUIElementTypeWi
                <XCUIElementTypeOther type="XCUIElementTyp
                    <XCUIElementTypeOther type="XCUIElemen
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeImage type="XCU
                            <XCUIElementTypeButton type="XC
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>
    """
    specialChar = "□"
    underCloseChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": self.FullScreenAttrDict,
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict,
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    underCloseSoup = CommonUtils.bsChainFind(soup, underClo
    if underCloseSoup:
        self.clickElementCenterPosition(underCloseSoup)
        foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessPopupPossibleCloseButton

```
def isPopupWindowSize(self, curSize):
    """判断一个soup的宽高大小是否是弹框类窗口(Image,Other等) 的大小
    # global FullScreenSize
    FullScreenSize = self.X * self.totalY
    curSizeRatio = curSize / FullScreenSize # 0.289
    PopupWindowSizeMinRatio = 0.25
    # PopupWindowSizeMaxRatio = 0.9
    PopupWindowSizeMaxRatio = 0.8
    # isSizeValid = curSizeRatio >= MinPopupWindowSizeRatio
    # is popup like window, size should large enough, but s
    isSizeValid = PopupWindowSizeMinRatio <= curSizeRatio <
    return isSizeValid

def isNormalButtonSize(self, curSize):
    """判断一个soup的宽高大小是否是普通的按钮大小"""
    NormalButtonSizeMin = 30*30
    NormalButtonSizeMax = 100*100
    isNormalSize = NormalButtonSizeMin <= curSize <= Normal
    return isNormalSize

def iOSProcessPopupPossibleCloseButton(self, soup):
    """
    必要 弹框 首单限时福利 右上角 关闭按钮
        特殊：但是内部无有效识别内容，比如name中包含close这类条件
        暂时只能特殊但通用处理：
            window -> other -> button
                且 button的 w和h基本一致，后者差距小于w和h最大值的
                以及 w和h都在一定（普通关闭按钮的大小）范围，比如 1

    必要 我的 蒙层弹框 首次使用引导页：
        基于上面逻辑：
            window->other->button
            button条件不变
        基础上，底层判断逻辑是：
            next的sibling后面，有且只有一个button，且有name
                且按钮宽高大小在合理范围，比如普通按钮的大小，算作为
                    意思是：引导页往往伴随着一个普通大小的按钮，供
    必要 弹框 推荐开通以下授权 关闭按钮在弹框下面：
        和前面逻辑略有不同
            Application -> window -> Other
        button条件不变
        其他条件类似于第一个，但是是 prev的sibling 是个Other元素，
    """
    foundAndProcessedPopup = False
    """
        必要 弹框 首单限时福利 右上角 关闭按钮：
            <XCUIElementTypeWindow type="XCUIElementTypeWin
            。。。
                <XCUIElementTypeOther type="XCUIElementType
```

```
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeImage type="XCUIElement
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>

        必要 我的 蒙层弹框 首次使用引导页:
            <XCUIElementTypeWindow type="XCUIElementTypeWi
                ...
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>

        必要 弹框 推荐开通以下授权 关闭按钮在弹框下面: :
            <XCUIElementTypeApplication type="XCUIElementTy
                <XCUIElementTypeWindow type="XCUIElementTyp
                    ...
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeImage type=
                                <XCUIElementTypeStaticText
                                <XCUIElementTypeStaticText
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                    <XCUIElementTypeButton type="XCUIElemer
    """
    # noNameP = re.compile("???")
    noNameP = False
    possibleCloseChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": {"enabled":"true", "visible":"true",
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true",
        },
        {
            "tag": "XCUIElementTypeButton",
```

```
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    possibleCloseSoup = CommonUtils.bsChainFind(soup, possi

    isCloseUnderPopup = False
    if not possibleCloseSoup:
        # 尝试找 是否是 关闭按钮在弹框下面的 弹框
        closeUnderPopupChainList = [
            {
                "tag": "XCUIElementTypeApplication",
                "attrs": {"enabled":"true", "visible":"true
            },
            {
                "tag": "XCUIElementTypeWindow",
                "attrs": {"enabled":"true", "visible":"true
            },
            {
                "tag": "XCUIElementTypeButton",
                "attrs": {"enabled":"true", "visible":"true
            },
        ]
        possibleCloseSoup = CommonUtils.bsChainFind(soup, c
        if possibleCloseSoup:
            isCloseUnderPopup = True

    if possibleCloseSoup:
        # check is match close button rule or not
        isValidSize = False
        isAlmostSameWH = False
        # isNexSiblingImage = False
        # isImageSizeValid = False
        isPossibleClose = True

        if isPossibleClose:
            soupAttrDict = possibleCloseSoup.attrs # {'enab
            logging.debug("possibleCloseSoup soupAttrDict=%
            # x = int(soupAttrDict["x"])
            # y = int(soupAttrDict["y"])
            width = int(soupAttrDict["width"])
            height = int(soupAttrDict["height"])
            # ButtonMinWH = 20
            # ButtonMinWH = 30
            ButtonMinWH = 25
            ButtonMaxWH = 60
            isValidWidth = ButtonMinWH  <= width  <= Buttor
            isValidHeight = ButtonMinWH <= height <= Buttor
            isValidSize = isValidWidth and isValidHeight

            isPossibleClose = isValidSize
```

```
            if isPossibleClose:
                isAlmostSameWH = False
                isSameWH = width == height
                if isSameWH:
                    isAlmostSameWH = isSameWH
                else:
                    maxWH = max(width, height)
                    diffWH = abs(width - height)
                    MaxAllowDiffRatio = 0.1
                    curDiffRatio = diffWH / maxWH
                    isValidDiff = curDiffRatio <= MaxAllowDiffR
                    isAlmostSameWH = isValidDiff

                isPossibleClose = isAlmostSameWH

            if isPossibleClose:
                # check next sibling is large Image
                # nextSiblingeSoup = possibleCloseSoup.next_sib
                # nextSiblingeSoupList = possibleCloseSoup.next
                nextSiblingeSoupGenerator = possibleCloseSoup.n
                nextSiblingeSoupList = list(nextSiblingeSoupGen

                hasLargeImage = CommonUtils.isContainSpecificSo
                isPossibleClose = hasLargeImage

                if not isPossibleClose:
                    hasNormalButton = CommonUtils.isContainSpec
                    isPossibleClose = hasNormalButton

                if not isPossibleClose:
                    if isCloseUnderPopup:
                        # prevSiblingSoupGenerator = possibleC
                        # prevPrevSiblingSoupList = list(prevSi
                        # hasLargeOther = CommonUtils.isContain
                        prevSiblingSoup = possibleCloseSoup.pre
                        if prevSiblingSoup:
                            prevPrevSiblingSoup = prevSiblingSo
                            prevPrevSiblingSoupList = [ prevPre
                            hasLargeOther = CommonUtils.isConta
                            isPossibleClose = hasLargeOther

            if isPossibleClose:
                foundAndProcessedPopup = self.findAndClickButto

        return foundAndProcessedPopup
```

iOSProcessPopupCommonNameContainClose

```
# def iOSProcessPopupUpperRightCommonClose(self, soup):
def iOSProcessPopupCommonNameContainClose(self, soup):
    foundAndProcessedPopup = False
    """
        京东金融 登录页 弹框 右上角 关闭按钮:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                </XCUIElementTypeOther>

        京东金融 tab2财富 弹框 我是Max alertClose:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                。。。
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemer

        必要 弹框 右上角关闭按钮 name中有close:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeImage type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeOther type="XCUIEle
                    </XCUIElementTypeImage>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        必要 验证码登录页 左上角 关闭 按钮 name中含close:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
    """
    commonCloseP = re.compile("close", flags=re.I)
    # com icon black close u
    # jr fianncing alertClose@3x
    # rights close white
    # login close
    commonCloseChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
```

```
            "tag": "XCUIElementTypeOther",
            # "attrs": {"enabled":"true", "visible":"true",
            # "attrs": {"enabled":"true", "visible":"true",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true",
        },
    ]
    commonCloseSoup = CommonUtils.bsChainFind(soup, commonC
    if commonCloseSoup:
        # self.clickElementCenterPosition(commonCloseSoup)
        # foundAndProcessedPopup = True

        # so change to wda query element then click
        foundAndProcessedPopup = self.findAndClickButtonEle
    return foundAndProcessedPopup
```

iOSProcessPopupUpperRightAlertClose

```
# def iOSProcessPopupUpperRightAlertClose(self, soup):
#     foundAndProcessedPopup = False
#     """
#         京东金融 tab2财富 弹框 我是Max alertClose:
#             <XCUIElementTypeOther type="XCUIElementTypeOt
#                 。。。
#                 <XCUIElementTypeOther type="XCUIElementTy
#                     <XCUIElementTypeButton type="XCUIEler
#     """
#     alertCloseP = re.compile("alertClose")
#     alertCloseChainList = [
#         {
#             "tag": "XCUIElementTypeOther",
#             "attrs": {"enabled":"true", "visible":"true",
#         },
#         {
#             "tag": "XCUIElementTypeOther",
#             "attrs": {"enabled":"true", "visible":"true",
#         },
#         {
#             "tag": "XCUIElementTypeButton",
#             "attrs": {"enabled":"true", "visible":"true",
#         },
#     ]
#     alertCloseSoup = CommonUtils.bsChainFind(soup, alertC
#     if alertCloseSoup:
#         self.clickElementCenterPosition(alertCloseSoup)
#         foundAndProcessedPopup = True
#     return foundAndProcessedPopup

# def iOSProcessPopupUpperRightClose(self, soup):
#     foundAndProcessedPopup = False
#     """
#         京东金融 登录页 弹框 右上角 关闭按钮:
#                 <XCUIElementTypeOther type="XCUIElementTy
#                     <XCUIElementTypeOther type="XCUIEleme
#                         <XCUIElementTypeStaticText type="
#                         <XCUIElementTypeButton type="XCUI
#                     </XCUIElementTypeOther>
#     """
#     containCloseP = re.compile("close") # com icon black
#     closeChainList = [
#         {
#             "tag": "XCUIElementTypeOther",
#             "attrs": {"enabled":"true", "visible":"true",
#         },
#         {
#             "tag": "XCUIElementTypeOther",
#             "attrs": {"enabled":"true", "visible":"true",
#         },
```

```
#         {
#             "tag": "XCUIElementTypeButton",
#             "attrs": {"enabled":"true", "visible":"true",
#         },
#     ]
#     closeSoup = CommonUtils.bsChainFind(soup, closeChainL
#     if closeSoup:
#         self.clickElementCenterPosition(closeSoup)
#         foundAndProcessedPopup = True
#     return foundAndProcessedPopup
```

iOSProcessRightUpperJumpOver

```
def iOSProcessRightUpperJumpOver(self, soup):
    foundAndProcessedPopup = False
    """
        京东金融 登录页 右上角 跳过:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
    """
    jumpOverChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {

            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    jumpOverSoup = CommonUtils.bsChainFind(soup, jumpOverCh
    if jumpOverSoup:
        self.clickElementCenterPosition(jumpOverSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessAlertUseWirelessData

```
def iOSProcessAlertUseWirelessData(self, soup):
    foundAndProcessedPopup = False
    """
        恒易贷 允许使用无线数据 无线局域网与蜂窝移动网络:
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
```

```
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>
    """
    wifiCellularChainList = [
        {
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    wifiCellularSoup = CommonUtils.bsChainFind(soup, wifiCe
    if wifiCellularSoup:
        # found 无线局域网与蜂窝移动网络 but
        # self.clickElementCenterPosition(wifiCellularSoup)
        # foundAndProcessedPopup = True

        # # change to wda query element then click by eleme
        # curName = wifiCellularSoup.attrs["name"] # 好
        # wifiCellularButtonQuery = {"type":"XCUIElementTyp
        # foundAndClicked = self.findAndClickElement(wifiCe
        # foundAndProcessedPopup = foundAndClicked
        foundAndProcessedPopup = self.findAndClickButtonEle
    return foundAndProcessedPopup
```

iOSProcessCommonAlertOk

```
def iOSProcessCommonAlertOk(self, soup):
    """
        对于 alert弹框，点击 好
    """
    foundAndProcessedPopup = False
    """
        恒易贷 弹框 想访问您的通讯录 好:
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementTyp
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEl
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEl
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>
    """
    accessYourP = re.compile("想访问您的") # “恒易贷”想访问您的
```

```
    okChainList = [
        {
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    okSoup = CommonUtils.bsChainFind(soup, okChainList)
    if okSoup:
        # clickCenterPosition(curSession, okSoup.attrs)
        # foundAndProcessedPopup = True

        # # Note: seems actual click is No allow button? ->
        # # change to wda query element then click for make
        # curName = okSoup.attrs["name"] # 好
        # okButtonQuery = {"type":"XCUIElementTypeButton",
        # foundAndClicked = self.findAndClickElement(okBut
        # foundAndProcessedPopup = foundAndClicked
        foundAndProcessedPopup = self.findAndClickButtonEle
    return foundAndProcessedPopup
```

iOSProcessCommonAlertAllow

```
def iOSProcessCommonAlertAllow(self, soup):
    foundAndProcessedPopup = False
    """
        京东金融 弹框 想给您发送信息 允许:
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementTyp
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>

        京东金融 弹框 允许“京东金融”在您使用该应用时访问您的位置吗?
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementTyp
                    <XCUIElementTypeOther type="XCUIElement
```

```
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeAlert>

            京东金融 弹框 允许在您并未使用该应用时访问您的位置吗？仅在使
                <XCUIElementTypeAlert type="XCUIElementTypeAle
                    <XCUIElementTypeOther type="XCUIElementType
                        <XCUIElementTypeOther type="XCUIElement
                            <XCUIElementTypeOther type="XCUIEle
                                <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type="XCU
                                    <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther type=
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCUIEle
```

```
                        <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type=
                                <XCUIElementTypeStatic
                                <XCUIElementTypeStatic
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type=
                            <XCUIElementTypeOther type=
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type=
                                <XCUIElementTypeOther t
                                    <XCUIElementTypeOth
                                        <XCUIElementTyp
                                    </XCUIElementTypeOt
                                    <XCUIElementTypeOth
                                        <XCUIElementTyp
                                        <XCUIElementTyp
                                    </XCUIElementTypeOt
                                    <XCUIElementTypeOth
                                        <XCUIElementTyp
                                    </XCUIElementTypeOt
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeAlert>

    恒易贷 想给您发送通知 允许:
        <XCUIElementTypeAlert type="XCUIElementTypeAle
            <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIEle
                        <XCUIElementTypeOther type="XCU
                        <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type=
                            <XCUIElementTypeOther type=
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIEle
                        <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type=
                                <XCUIElementTypeStatic
                                <XCUIElementTypeStatic
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type=
```

```
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeAlert>
    """
    # allowP = re.compile("(始终)?允许") # will also match "
    allowP = re.compile("^(始终)?允许$") # only match "允许"
    allowChainList = [
        {
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            # "attrs": {"enabled":"true", "visible":"true",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    allowSoup = CommonUtils.bsChainFind(soup, allowChainLis
    if allowSoup:
        # clickCenterPosition(curSession, allowSoup.attrs)
        # foundAndProcessedPopup = True

        # # above click position not work for 允许 -> actua
        # # change to use wda to find 允许 then click eleme
        # curName = allowSoup.attrs["name"] # 允许 / 始终允许
        # # allowButtonQuery = {"type":"XCUIElementTypeBut
        # allowButtonQuery = {"type":"XCUIElementTypeButto
        # foundAndClickeAllow = self.findAndClickElement(a
```

```
        # foundAndProcessedPopup = foundAndClickeAllow
        foundAndProcessedPopup = self.findAndClickButtonEle
    return foundAndProcessedPopup
```

iOSProcessUserPrivacyProtocol

```
def iOSProcessUserPrivacyProtocol(self, soup):
    foundAndProcessedPopup = False
    """
        恒易贷 首次进入 用户隐私保护协议 点击 我接受
    """

    """
        恒易贷 用户意思保护协议 我接收:
            <XCUIElementTypeWindow type="XCUIElementTypeWin
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeOther type="XCUIEle
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>
    """
    userPrivacyIAcceptChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]

    userPrivacyIAcceptSoup = CommonUtils.bsChainFind(soup,
    if userPrivacyIAcceptSoup:
        self.clickElementCenterPosition(userPrivacyIAcceptS
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessCommonUserAgreementAgree

```
def iOSProcessCommonUserAgreementAgree(self, soup):
    """
        京东金融、必要 首次进入 授权协议提示 点击 同意
    """
    foundAndProcessedAgreement = False
    """
        京东金融 首次进入 协议提示 同意:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
            。。。
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        必要 用户服务协议及必要隐私政策 同意:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTextView type="XCUIEle
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        斑马AI课 弹框 个人信息保护政策 同意:
            <XCUIElementTypeWindow type="XCUIElementTypeWin
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeTextView type="XCUI
                            <XCUIElementTypeLink type="XCUI
                            <XCUIElementTypeLink type="XCUI
                            <XCUIElementTypeLink type="XCUI
                        </XCUIElementTypeTextView>
                        <XCUIElementTypeOther type="XCUIEle
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>
    """
    commonAgreeChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "
```

```
        },
        {
            "tag": "XCUIElementTypeOther",
            # "attrs": {"enabled":"true", "visible":"true",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    commonAgreeSoup = CommonUtils.bsChainFind(soup, commonA
    if commonAgreeSoup:
        self.clickElementCenterPosition(commonAgreeSoup)
        foundAndProcessedAgreement = True
    return foundAndProcessedAgreement
```

iOSProcessPopupIKnow

```
def iOSProcessPopupIKnow(self, soup):
    """
        京东金融 首次打开 协议弹框 我知道了
    """
    foundAndProcessedPopup = False
    """
        京东金融 弹框 我知道了:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeScrollView type="XCUIE
                        <XCUIElementTypeStaticText type="X
                        <XCUIElementTypeOther type="XCUIEl
                    </XCUIElementTypeScrollView>
                    <XCUIElementTypeOther type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        恒易贷 权限获取说明 我知道了:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTable type="XCUIElemen
                        <XCUIElementTypeOther type="XCUIEl
                        <XCUIElementTypeCell type="XCUIEle
                            <XCUIElementTypeImage type="XC
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIEle
                            <XCUIElementTypeImage type="XC
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIEle
                            <XCUIElementTypeImage type="XC
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeCell>
                        <XCUIElementTypeCell type="XCUIEle
                            <XCUIElementTypeImage type="XC
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeCell>
                        <XCUIElementTypeOther type="XCUIEl
                    </XCUIElementTypeTable>
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeOther>
```

```
            </XCUIElementTypeOther>
        """
        iKnowChainList = [
            {
                "tag": "XCUIElementTypeOther",
                "attrs": {"enabled":"true", "visible":"true", '
            },
            {
                "tag": "XCUIElementTypeOther",
                "attrs": {"enabled":"true", "visible":"true"}
            },
            {
                "tag": "XCUIElementTypeButton",
                "attrs": {"enabled":"true", "visible":"true", '
            },
        ]
        iKnowSoup = CommonUtils.bsChainFind(soup, iKnowChainLis
        if iKnowSoup:
            self.clickElementCenterPosition(iKnowSoup)
            foundAndProcessedPopup = True
        return foundAndProcessedPopup
```

iOSProcessPopupPhoneCall

```
def iOSProcessPopupPhoneCall(self, soup):
    """iOS 通用弹框 呼叫 -> 点击 取消"""
    foundAndProcessedPopup = False
    """
        iOS 通用弹框 呼叫 打电话:
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>
    """
    callChainList = [
        {
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true"}
```

```
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    callSoup = CommonUtils.bsChainFind(soup, callChainList)
    if callSoup:
        # parentOtherSoup = callSoup.parent
        # if parentOtherSoup:
        #     parentParentOtherSoup = parentOtherSoup.parer
        #     if parentParentOtherSoup:
        #         cancelSoup = parentParentOtherSoup.find(
        #             "XCUIElementTypeButton",
        #             attrs={"enabled":"true", "visible":"t
        #         )
        #         if cancelSoup:
        #             clickCenterPosition(curSession, cance
        #             foundAndProcessedPopup = True

        # # above click position not work for 取消 !!!
        # # change to find 取消 then click element
        # cancelButtonQuery = {"type":"XCUIElementTypeButto
        # foundAndClickeCancel = self.findAndClickElement(c
        # foundAndProcessedPopup = foundAndClickeCancel
        foundAndProcessedPopup = self.findAndClickButtonEle
    return foundAndProcessedPopup
```

iOSProcessCommonPopupWindowUpperRightClose

```
# 注：此逻辑经常误判，没有弹框误以为有弹框，所以放弃此逻辑
def iOSProcessCommonPopupWindowUpperRightClose(self, soup):
    """iOS 常见 通用弹框：  中间有多层 XCUIElementTypeOther 且 是
        enabled="true" visible="true" x="0" y="0" width="41
            至少3层，其下 再去找 就是 弹框本身最外层元素了
                然后尝试点击右上角关闭按钮
    """
    foundAndProcessedPopup = False
    """
    弹框中找 外层Other内部的 非x=0 y=0的元素，则为弹框区域 计算出其
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=

            。。。
    """
    allTopOtherSoupList = soup.find_all(
        "XCUIElementTypeOther",
        attrs={"enabled":"true", "visible":"true", "x":"0",
    )
    for eachTopOtherSoup in allTopOtherSoupList:
        popupTopOtherSoup = self.findPopupTopFrameElement(e
        if popupTopOtherSoup:
            curAttrsDict = popupTopOtherSoup.attrs
            curX = int(curAttrsDict["x"])
            curY = int(curAttrsDict["y"])
            curWidth = int(curAttrsDict["width"])
            curHeight = int(curAttrsDict["height"])
            curX1 = curX + curWidth
            curY1 = curY + curHeight
            PossibleCloseButtonWidth = 40
            PossibleCloseButtonHeight = 40
            possibleCloseButtonX0 = curX1 - PossibleCloseBu
            # possibleCloseButtonY0 = curY + PossibleCloseB
            possibleCloseButtonY0 = curY
            possibleCloseButtonX1 = curX1
            possibleCloseButtonY1 = curY + PossibleCloseBut
            # possibleCloseButtonCenterX = possibleCloseBut
            # possibleCloseButtonCenterY = possibleCloseBut
            possibleCloseButtonCenterX = int( (possibleClos
            possibleCloseButtonCenterY = int( (possibleClos
            possibleCloseButtonCenterPositon = (possibleClo
            self.tap(possibleCloseButtonCenterPositon)
            foundAndProcessedPopup = True
            break

    return foundAndProcessedPopup
```

findPopupTopFrameElement

```
def findPopupTopFrameElement(self, topOtherSoup):
    """
    寻找符合条件的子节点，即当前节点向下找，符合一直是
        type="XCUIElementTypeOther" enabled="true" visible=
        且层数 >= 3，然后再找其下一个 非x=0 y=0的节点
            很可能就是 弹框的主体元素
    """
    popupTopFrameElement = None

    # MaxFullScreenSizeLevel = 3
    # FullScreenSizeMinLevel = 3
    # FullScreenSizeMaxLevel = 6 # see many invalid case is
    FullScreenSizLevel = 3 # special: 无忧筹 点击 专属老师 后的

    FullScreenAttr = {"type": "XCUIElementTypeOther", "enab

    curFullScreenOtherLevel = 0
    curSoup = topOtherSoup
    while True:
        subOtherSoupList = curSoup.find_all(
            "XCUIElementTypeOther",
            attrs=FullScreenAttr,
            recursive=False,
        )

        if not subOtherSoupList:
            break

        # only have one child
        childOtherSoupNum = len(subOtherSoupList)
        if childOtherSoupNum != 1:
            break

        firstOnlyOtherSoup = subOtherSoupList[0]
        if not firstOnlyOtherSoup:
            break

        if not hasattr(firstOnlyOtherSoup, "attrs"):
            break

        curAttrDict = firstOnlyOtherSoup.attrs

        allAttrSame = True
        for eachToCompareKey in FullScreenAttr.keys():
            eachToCompareValue = FullScreenAttr[eachToCompa
            curValue = curAttrDict.get(eachToCompareKey)
            if curValue != eachToCompareValue:
                allAttrSame = False
                break
```

```
            if not allAttrSame:
                break

            curSoup = firstOnlyOtherSoup

            curFullScreenOtherLevel += 1

        # if curFullScreenOtherLevel >= MaxFullScreenSizeLevel:
        # isPossiblePopup = FullScreenSizeMinLevel <= curFullSc
        isPossiblePopup = curFullScreenOtherLevel == FullScreen
        if isPossiblePopup:
            curChildOtherList = curSoup.find_all(
                "XCUIElementTypeOther",
                attrs={"type": "XCUIElementTypeOther", "enabled
                recursive=False,
            )
            if curChildOtherList:
                curChildOtherNum = len(curChildOtherList)
                if curChildOtherNum == 1:
                    popupTopFrameElement = curChildOtherList[0]

        return popupTopFrameElement
```

iOSProcessPopupPhotoCamera

```
def iOSProcessPopupPhotoCamera(self, soup):
    """iOS 系统弹框:  拍照或录像 照片图库 浏览 取消"""
    foundAndProcessedPopup = False
    """
        系统弹框 拍照或录像:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                                <XCUIElemer
                                                    <XCUIE
                                                        <X
                                                        <X
                                                        <X
                                                        <X
                                                    </XCUIE
                                                    <XCUIE
                                                        <X
                                                        <X
                                                        <X
                                                        <X
                                                    </XCUIE
                                                    <XCUIE
                                                        <X
                                                        <X
                                                        <X
                                                        <X
                                                    </XCUIE
                                                    <XCUIE
                                                </XCUIEleme
                                            </XCUIElementTy
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementTypeOthe
                    <XCUIElementTypeButton type="XCUIElementTyp
                </XCUIElementTypeOther>
    """
    photoCameraChainList = [
        {
```

```
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeTable",
            "attrs": {"enabled":"true", "visible":"true", '
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    photoCameraSoup = CommonUtils.bsChainFind(soup, photoCa
    if photoCameraSoup:
        topParentOtherSoup = None
        # find top most parent
        parentLevelNum = 11
        curParentSoup = photoCameraSoup
        for curLevelIdx in range(parentLevelNum):
            curParentSoup = curParentSoup.parent
            if not curParentSoup:
                break
        if curParentSoup:
            topParentOtherSoup = curParentSoup

        if topParentOtherSoup:
            nextSiblingList = topParentOtherSoup.next_sibli
            if nextSiblingList:
                nextOtherSoup = None
                for eachNextSibling in nextSiblingList:
                    if hasattr(eachNextSibling, "attrs"):
                        curType = eachNextSibling.attrs["ty
                        if curType == "XCUIElementTypeOthe
                            # next sibling's first XCUIEle
                            nextOtherSoup = eachNextSiblin
                            break

            if nextOtherSoup:
                cancelSoup = nextOtherSoup.find(
                    "XCUIElementTypeButton",
                    attrs={"visible":"true", "enabled":"tru
                )
                if cancelSoup:
                    self.clickElementCenterPosition(cancelS
                    foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessAlertNoteConfrim

```
def iOSProcessAlertNoteConfrim(self, soup):
    """只要符合XCUIElementTypeAlert 且其中有 确定 按钮，则点击"""
    foundAndProcessedPopup = False
    """
        廉爱公社 弹框 提醒 您的身份信息不完整 确定:
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>

        廉爱公社 弹框 提示 请登录后再进行操作:
            <XCUIElementTypeAlert type="XCUIElementTypeAle
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
```

```
                        </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeAlert>
        """
        noteTipP = re.compile("((提醒)|(提示))")
        alertConfirmChainList = [
            {
                "tag": "XCUIElementTypeAlert",
                # "attrs": {"enabled":"true", "visible":"true"}
                "attrs": {"enabled":"true", "visible":"true", "
            },
            {
                "tag": "XCUIElementTypeOther",
                "attrs": {"enabled":"true", "visible":"true"}
            },
            {
                "tag": "XCUIElementTypeButton",
                "attrs": {"enabled":"true", "visible":"true", "
            },
        ]
        alertConfirmSoup = CommonUtils.bsChainFind(soup, alertC
        if alertConfirmSoup:
            self.clickElementCenterPosition(alertConfirmSoup)
            foundAndProcessedPopup = True
        return foundAndProcessedPopup
```

iOSProcessAlertCancel

```
def iOSProcessAlertCancel(self, soup):
    """对于常见的XCUIElementTypeAlert，内有 取消，则点击取消"""
    foundAndProcessedPopup = False
    """
        益路通行 弹框 您是否要退出登录:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeAlert type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                    <XCUIElementTypeOther t
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeSta
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                    <XCUIElementTypeOther t
                                </XCUIElementTypeOther>
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                                <XCUIElemer
                                            </XCUIElementTy
                                            <XCUIElementTyp
                                                <XCUIElemer
                                                <XCUIElemer
                                            </XCUIElementTy
                                            <XCUIElementTyp
                                                <XCUIElemer
                                            </XCUIElementTy
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeAlert>
    """
    alertCancelChainList = [
```

```
        {
            "tag": "XCUIElementTypeAlert",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", '
        },
    ]
    alertCancelSoup = CommonUtils.bsChainFind(soup, alertCa
    if alertCancelSoup:
        self.clickElementCenterPosition(alertCancelSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessKagsPopupStillNotLogin

```
def iOSProcessKagsPopupStillNotLogin(self, soup):
    foundAndProcessedPopup = False
    """
        廉爱公社 弹框 您还未登录:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
            。。。
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    stillNotLoginChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "x":"0", "width": "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible":"true", "enabled":"true"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible":"true", "enabled":"true", "
        },
    ]
    stillNotLoginSoup = CommonUtils.bsChainFind(soup, stil
    if stillNotLoginSoup:
        parentOtherSoup = stillNotLoginSoup.parent
        parentParentOtherSoup = parentOtherSoup.parent
        if parentParentOtherSoup:
            tempNotLoginSoup = parentParentOtherSoup.find(
                "XCUIElementTypeStaticText",
                attrs={"visible":"true", "enabled":"true",
            )
            if tempNotLoginSoup:
                self.clickElementCenterPosition(tempNotLog
                foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessWillOpenNonOfficialPage

```
def iOSProcessWillOpenNonOfficialPage(self, soup):
    foundAndProcessed = False

    """
        将要访问 非有赞官方网页，请确认是否继续访问
        <XCUIElementTypeWebView type="XCUIElementTypeWebVie
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                                <XCUIElementTypeButton type
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeStatic
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWebView>
    """
    willOpenChainList = [
        {
            "tag": "XCUIElementTypeWebView",
            "attrs": {"visible":"true", "enabled":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible":"true", "enabled":"true", "
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible":"true", "enabled":"true", "
        },
    ]
    willOpenSoup = CommonUtils.bsChainFind(soup, willOpenCh
    if willOpenSoup:
        parentOtherSoup = willOpenSoup.parent
        parentParentOtherSoup = parentOtherSoup.parent
        if parentParentOtherSoup:
```

```
            continueOpenSoup = parentParentOtherSoup.find(
                "XCUIElementTypeButton",
                attrs={"visible":"true", "enabled":"true",
            )
            if continueOpenSoup:
                self.clickElementCenterPosition(continueOpe
                foundAndProcessed = True

    return foundAndProcessed
```

iOSProcessSettingsAllowUseLocation

```
def iOSProcessSettingsAllowUseLocation(self, soup):
    """处理 设置中的 使用我的地理位置，并返回前一页
        注：往往是出现 弹框-是否授权当前位置，点击 允许后，跳转到此处
    """

    """
        设置 使用我的地理位置：
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeNavigationBar type="XCUIEle
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeNavigationBar>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeTable type="XCU
                                <XCUIElementTypeStaticText
                                <XCUIElementTypeCell type="
                                    <XCUIElementTypeOther t
                                    <XCUIElementTypeOther t
                                    <XCUIElementTypeStatic
                                    <XCUIElementTypeSwitch
                                </XCUIElementTypeCell>
                                <XCUIElementTypeOther type=
                            </XCUIElementTypeTable>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    foundAndProcessedPopup = False
    useLocationSwitchQuery = {"type": "XCUIElementTypeSwitc
    parentCellClassChain = "/XCUIElementTypeCell[`enabled =
    useLocationSwitchQuery["parent_class_chains"] = [ parer
    foundAndClickeUseLocation = self.findAndClickElement(us
    logging.debug("foundAndClickeUseLocation=%s", foundAndC
    if foundAndClickeUseLocation:
        parentNaviBarClassChain = '/XCUIElementTypeNavigat
        backButtonQuery = {"type": "XCUIElementTypeButton",
        backButtonQuery["parent_class_chains"] = parentNavi
        foundAndClickeBack = self.findAndClickElement(backB
        logging.debug("foundAndClickeBack=%s", foundAndClic
        foundAndProcessedPopup = foundAndClickeBack

    return foundAndProcessedPopup
```

iOSProcessPopupNetworkNotStableRetry

```
def iOSProcessPopupNetworkNotStableRetry(self, soup):
    """Process iOS popup 京东金融 网络不稳定,请点击重试"""
    foundAndProcessed = False
    """
        京东金融 页面 网络不稳定 请点击重试:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemer
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemer
                </XCUIElementTypeOther>
                <XCUIElementTypeTable type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeCell type="XCUIElement
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeImage type="XCUIEle
                    </XCUIElementTypeCell>
                    <XCUIElementTypeOther type="XCUIElement
                </XCUIElementTypeTable>
            </XCUIElementTypeOther>
    """
    networkNotStableChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled":"true", "visible":"true", "
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled":"true", "visible":"true", "
        },
    ]
    networkNotStableSoup = CommonUtils.bsChainFind(soup, ne
    if networkNotStableSoup:
        self.clickElementCenterPosition(networkNotStableSou
        foundAndProcessed = True
    return foundAndProcessed
```

iOSProcessPopupNetworkNotStableRefresh

```
def iOSProcessPopupNetworkNotStableRefresh(self, soup):
    """Process iOS popup 京东金融 页面 网络不稳定 刷新试试"""
    foundAndProcessedPopup = False
    """
        京东金融 页面 网络不稳定 刷新试试:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeTable type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
    """
    tryRefreshChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true",
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true",
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true",
        },
    ]
    tryRefreshSoup = CommonUtils.bsChainFind(soup, tryRefre
    if tryRefreshSoup:
        self.clickElementCenterPosition(tryRefreshSoup)
        foundAndProcessedPopup = True

        # # Special: above click by position not work
        # # so change to wda query element then click
        # tryRefreshButtonQuery = {"type":"XCUIElementTypeB
        # foundAndClicked = findAndClickElement(curSession,
        # foundAndProcessedPopup = foundAndClicked

    return foundAndProcessedPopup
```

iOSProcessPopupCertificateError

```
def iOSProcessPopupCertificateError(self, soup):
    """Process iOS popup 安全证书存在问题 网络出错，轻触屏幕重新加
    foundAndProcessed = False

    """
        <XCUIElementTypeButton type="XCUIElementTypeButton"
            <XCUIElementTypeStaticText type="XCUIElementTyp
        </XCUIElementTypeButton>
    """
    netErrTouchReloadP = re.compile("网络出错，轻触屏幕重新加载
    foundTouchReload = soup.find(
        "XCUIElementTypeButton",
        attrs={"visible":"true", "name": netErrTouchReloadP
    )
    if foundTouchReload:
        self.clickElementCenterPosition(foundTouchReload)
        foundAndProcessed = True

    if not foundAndProcessed:
        """
            <XCUIElementTypeButton type="XCUIElementTypeBut
                <XCUIElementTypeStaticText type="XCUIElemen
            </XCUIElementTypeButton>

            <XCUIElementTypeWindow type="XCUIElementTypeWin
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIElemen
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                    </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>
        """
        securityWarningChainList = self.generateCommonPopup
        securityWarningSoup = CommonUtils.bsChainFind(soup,
        if securityWarningSoup:
            parentButtonSoup = securityWarningSoup.parent
            if parentButtonSoup:
                certificateProblemP = re.compile("该网站的安
                foundCertProblem = parentButtonSoup.find(
                    "XCUIElementTypeStaticText",
                    attrs={"visible":"true", "enabled":"tru
                )
                if foundCertProblem:
```

```
                    foundCancel = parentButtonSoup.find(
                        "XCUIElementTypeStaticText",
                        attrs={"visible":"true", "enabled"
                    )
                    if foundCancel:
                        self.clickElementCenterPosition(fou
                        foundAndProcessed = True

    return foundAndProcessed
```

iOSProcessPopupWeixinLogin

```
def iOSProcessPopupWeixinLogin(self, soup):
    """Process iOS popup 微信登录"""
    foundAndProcessedPopup = False

    """
        <XCUIElementTypeWindow type="XCUIElementTypeWindow"
            <XCUIElementTypeOther type="XCUIElementTypeOthe
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeTable type="XCUIElement
                        <XCUIElementTypeCell type="XCUIEle
                        <XCUIElementTypeImage type="XCUIEl
                        <XCUIElementTypeStaticText type="X
                        </XCUIElementTypeCell>
                    </XCUIElementTypeTable>
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIEleme
                        <XCUIElementTypeStaticText type="X
                    </XCUIElementTypeButton>
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeButton type="XCUIEleme
                        <XCUIElementTypeStaticText type="X
                    </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
    """
    weixinLoginChainList = self.generateCommonPopupItemCha
        secondLevelTag="XCUIElementTypeOther",
        thirdLevelValue="微信登录"
    )
    foundWeixinLogin = CommonUtils.bsChainFind(soup, weixi

    tableChainList = self.generateCommonPopupItemChainList
        secondLevelTag="XCUIElementTypeOther",
        thirdLevelTag="XCUIElementTypeTable",
    )
    foundTable = CommonUtils.bsChainFind(soup, tableChainL

    if foundWeixinLogin and foundTable:
```

```
            parentOtherSoup = foundWeixinLogin.parent
            if parentOtherSoup:
                foundAllowButton = parentOtherSoup.find(
                    "XCUIElementTypeButton",
                    attrs={"visible":"true", "enabled":"true",
                )
                if foundAllowButton:
                    self.clickElementCenterPosition(foundAllowB
                    foundAndProcessedPopup = True

        return foundAndProcessedPopup
```

iOSProcessPopupJumpThirdApp

```
def iOSProcessPopupJumpThirdApp(self, soup):
    """Process iOS popup 可能离开微信，打开第三方应用"""
    foundAndProcessedPopup = False

    """
        <XCUIElementTypeWindow type="XCUIElementTypeWindow"
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeButton type="XCUIE
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                        </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
    """
    leaveWeixinChainList = self.generateCommonPopupItemCha
    foundLeaveWeixin = CommonUtils.bsChainFind(soup, leaveW
    if foundLeaveWeixin:
        parentButtonSoup = foundLeaveWeixin.parent
        if parentButtonSoup:
            foundCancel = parentButtonSoup.find(
                "XCUIElementTypeStaticText",
                attrs={"visible":"true", "enabled":"true",
            )
            if foundCancel:
                self.clickElementCenterPosition(foundCance
                foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessPopupDoSomeFail

```
# def iOSProcessPopupGetInfoFail(self, soup):
def iOSProcessPopupDoSomeFail(self, soup):
    """Process iOS popup 获取信息失败 微信登录失败 等"""
    foundAndProcessedPopup = False
    """
        微信:
            获取信息失败:
            <XCUIElementTypeWindow type="XCUIElementTypeWi
            。。。
                <XCUIElementTypeButton type="XCUIElementTyp
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                </XCUIElementTypeButton>
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeButton type="XCUIE
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            。。。
            </XCUIElementTypeWindow>

            微信登录失败:
            <XCUIElementTypeWindow type="XCUIElementTypeWi
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeButton type="XCUIElemen
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeImage type="XCU
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeButton type="XC
                        </XCUIElementTypeOther>
                    </XCUIElementTypeButton>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>

        小程序:
            获取信息失败:
            <XCUIElementTypeButton type="XCUIElementTypeBut
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTextView type="XCUIEle
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
    """
    # getInfoFailChainList = self.generateCommonPopupItemCh
```

```
        # getInfoFailSoup = CommonUtils.bsChainFind(soup, getIn
        # if getInfoFailSoup:
        #     parentButtonOrOtherSoup = getInfoFailSoup.parent
        doSomeFailP = re.compile("\S+失败$") # 获取信息失败, 微信登
        doSomeFailChainList = self.generateCommonPopupItemChain
        doSomeFailSoup = CommonUtils.bsChainFind(soup, doSomeFa
        if doSomeFailSoup:
            parentButtonOrOtherSoup = doSomeFailSoup.parent
            if parentButtonOrOtherSoup:
                # 微信 弹框: 获取信息失败
                foundConfirm = parentButtonOrOtherSoup.find(
                    "XCUIElementTypeStaticText",
                    attrs={"visible":"true", "enabled":"true",
                )

                if not foundConfirm:
                    # 小程序 弹框: 获取信息失败
                    # 微信 弹框: 微信登录失败
                    foundConfirm = parentButtonOrOtherSoup.find
                        "XCUIElementTypeButton",
                        attrs={"visible":"true", "enabled":"tru
                    )

                if foundConfirm:
                    self.clickElementCenterPosition(foundConfir
                    foundAndProcessedPopup = True

        return foundAndProcessedPopup
```

iOSProcessPopupWeixinAuthorizeLocation

```
def iOSProcessPopupWeixinAuthorizeLocation(self, soup):
    """Process iOS popup 是否授权当前位置"""
    foundAndProcessedPopup = False
    """
        微信 弹框 是否授权当前位置:
            <XCUIElementTypeButton type="XCUIElementTypeBut
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTextView type="XCUIEle
                    <XCUIElementTypeButton type="XCUIElemen
                    <XCUIElementTypeButton type="XCUIElemen
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
    """
    authorizeLocationChainList = [
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"visible":"true", "enabled":"true",
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible":"true", "enabled":"true"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible":"true", "enabled":"true",
        },
    ]
    authorizeLocationSoup = CommonUtils.bsChainFind(soup, a
    if authorizeLocationSoup:
        parentOtherSoup = authorizeLocationSoup.parent
        confirmSoup = parentOtherSoup.find(
            "XCUIElementTypeButton",
            attrs={"visible":"true", "enabled":"true", "nam
        )
        if confirmSoup:
            self.clickElementCenterPosition(confirmSoup)
            foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessPopupMiniprogamWarning

```
def iOSProcessPopupMiniprogamWarning(self, soup):
    """
        微信-小程序 弹框 警告 尚未进行授权，请点击确定跳转到授权页面
    """
    foundAndProcessedPopup = False
    """
        微信-小程序 弹框 警告 尚未进行授权:
            <XCUIElementTypeButton type="XCUIElementTypeBut
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTextView type="XCUIEle
                    <XCUIElementTypeButton type="XCUIEleme
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
    """
    warningChainList = [
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"visible":"true", "enabled":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible":"true", "enabled":"true"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible":"true", "enabled":"true", "
        },
    ]
    warningSoup = CommonUtils.bsChainFind(soup, warningCha
    if warningSoup:
        parentOtherSoup = warningSoup.parent
        confirmSoup = parentOtherSoup.find(
            "XCUIElementTypeButton",
            attrs={"visible":"true", "enabled":"true", "nam
        )
        if confirmSoup:
            self.clickElementCenterPosition(confirmSoup)
            foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessPopupMiniprogamGetYour

```
# def iOSProcessPopupMiniprogamAuthority(self, soup):
def iOSProcessPopupMiniprogamGetYour(self, soup):
    """Process iOS popup 获取你的昵称、头像、地区及性别 / 获取你
    foundAndProcessedPopup = False
    """
        弹框 获取你的昵称、头像、地区及性别:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeTable type="XCUIElement
                        <XCUIElementTypeCell type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeImage type="XCU
                            <XCUIElementTypeOther type="XCU
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeStaticText type
                            <XCUIElementTypeImage type="XCU
                        </XCUIElementTypeCell>
                    </XCUIElementTypeTable>
                    <XCUIElementTypeButton type="XCUIEleme
                    <XCUIElementTypeButton type="XCUIEleme
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        弹框 获取你的通讯地址:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeImage type="XCUIElement
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeStaticText type="XCUIE
                    <XCUIElementTypeButton type="XCUIEleme
                    <XCUIElementTypeButton type="XCUIEleme
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>

        小程序 弹框 需要获取你的地理位置:
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeOther type="XCUIElementType
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeOther type="XCUIElement
                    <XCUIElementTypeAlert type="XCUIElement
```

```
                        <XCUIElementTypeOther type="XCUIEle
                            <XCUIElementTypeOther type="XCU
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                        <XCUIElementTypeOth
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                                <XCUIElementTypeOther type=
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                        <XCUIElementTypeOth
                                    </XCUIElementTypeOther>
                                    <XCUIElementTypeOther t
                                        <XCUIElementTypeOth
                                            <XCUIElementTyp
                                                <XCUIElemer
                                                    <XCUIE
                                                </XCUIEleme
                                                <XCUIElemer
                                                    <XCUIE
                                                    <XCUIE
                                                </XCUIEleme
                                                <XCUIElemer
                                                    <XCUIE
                                                </XCUIEleme
                                            </XCUIElementTy
                                        </XCUIElementTypeOt
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeAlert>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
    """
    # getYourP = re.compile("获取你的\S+") # 获取你的通讯地址,
    getYourP = re.compile("(需要)?获取你的\S+") # 获取你的通讯地
    getYourChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible":"true", "enabled":"true", "
        },
        {
            "tag": "XCUIElementTypeOther",
```

```
            "attrs": {"visible":"true", "enabled":"true", '
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible":"true", "enabled":"true", '
        },
    ]
    getYourSoup = CommonUtils.bsChainFind(soup, getYourCha
    if getYourSoup:
        parentOther = getYourSoup.parent
        parentParentOther = None
        parentParentParentOther = None

        allowTag = "XCUIElementTypeButton"
        allowAttrs = {"visible":"true", "enabled":"true", '

        # 获取你的通讯地址, 获取你的昵称、头像、地区及性别
        foundAllow = parentOther.find(allowTag, attrs=allow

        if not foundAllow:
            # for support future possible case
            parentParentOther = parentOther.parent
            foundAllow = parentParentOther.find(allowTag, a

        if not foundAllow:
            # "贝德玛会员中心" 需要获取你的地理位置
            parentParentParentOther = parentParentOther.par
            foundAllow = parentParentParentOther.find(allow

        if foundAllow:
            self.clickElementCenterPosition(foundAllow)
            foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

iOSProcessPopupMiniprogamDisclaimer

```
def iOSProcessPopupMiniprogamDisclaimer(self, soup):
    """Process iOS popup 免责声明"""
    foundAndProcessedPopup = False
    """
        <XCUIElementTypeWindow type="XCUIElementTypeWindow"
            <XCUIElementTypeOther type="XCUIElementTypeOthe
                <XCUIElementTypeButton type="XCUIElementTyp
                    <XCUIElementTypeOther type="XCUIElement
                        <XCUIElementTypeImage type="XCUIEle
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeStaticText type="XC
                        <XCUIElementTypeButton type="XCUIE
                    </XCUIElementTypeOther>
                </XCUIElementTypeButton>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
    """
    disclaimerChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": {"visible":"true", "enabled":"true", "
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"visible":"true", "enabled":"true", "
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible":"true", "enabled":"true", "
        },
    ]
    disclaimerSoup = CommonUtils.bsChainFind(soup, disclai
    if disclaimerSoup:
        parentOtherSoup = disclaimerSoup.parent
        if parentOtherSoup:
            foundIKnow = parentOtherSoup.find(
                "XCUIElementTypeButton",
                attrs={"visible":"true", "enabled":"true",
            )
            if foundIKnow:
                self.clickElementCenterPosition(foundIKnow
                foundAndProcessedPopup = True

    return foundAndProcessedPopup
```

期间部分相关函数：

```
def generateCommonPopupItemChainList(self,
        firstLevelTag="XCUIElementTypeWindow",
        secondLevelTag="XCUIElementTypeButton",
        thirdLevelTag="XCUIElementTypeStaticText",
        thirdLevelValue=None,
        thirdLevelName=None,
    ):
    CommonAttrs_VisibleEnabledFullWidthFullHeight = {"visib
    commonItemChainList = [
        {
            "tag": firstLevelTag,
            "attrs": CommonAttrs_VisibleEnabledFullWidthFu
        },
        {
            "tag": secondLevelTag,
            "attrs": CommonAttrs_VisibleEnabledFullWidthFu
        },
    ]
    thirdItemAttrs = {"visible":"true", "enabled":"true"}
    if thirdLevelValue:
        thirdItemAttrs["value"] = thirdLevelValue
    if thirdLevelName:
        thirdItemAttrs["name"] = thirdLevelName
    thirdItemDict = {
        "tag": thirdLevelTag,
        "attrs": thirdItemAttrs
    }
    commonItemChainList.append(thirdItemDict)
    return commonItemChainList
```

调试辅助

在开发期间，往往要调试一些内容，比如页面元素有哪些，位置大小等如何，所以会想要一些函数，可以给图片上元素加框显示等等，这类函数，属于方便调试，辅助调试方面的内容。

之前实现了很多复制调试的函数，如下，供参考。

debugDrawElementRect debugDrawScreenRect 给图片画框

```
def debugDrawScreenRect(self, curRect, curImgPath=None, isS
    """for debug, draw rectange for current screenshot"""
    if not curImgPath:
        curImgPath = self.getCurScreenshot()

    curImg = CommonUtils.imageDrawRectangle(
        curImgPath,
        curRect,
        isShow=isShow,
        isAutoSave=isAutoSave,
        isDrawClickedPosCircle=isDrawClickedPosCircle,
    )

    return curImg

def debugDrawElementRect(self, elementList, curImgPath=None
    """for debug, to draw rectange for each element in curr
    if not curImgPath:
        curImgPath = self.getCurScreenshot()

    curImg = Image.open(curImgPath)

    for eachElement in elementList:
        curBoundList = self.get_ElementBounds(eachElement)
        curWidth = curBoundList[2] - curBoundList[0]
        curHeight = curBoundList[3] - curBoundList[1]
        curRect = [curBoundList[0], curBoundList[1], curWid
        curTimeStr = CommonUtils.getCurDatetimeStr("%H%M%S"
        curSaveTal = "_rect_{}_%x|%y|%w|%h".format(curTimeS
        curInputImg = None
        if isDrawInSinglePic:
            curInputImg = curImg
        else:
            curInputImg = curImgPath
        curImg = CommonUtils.imageDrawRectangle(
            curInputImg,
            curRect,
            isShow=isShowEach,
            isAutoSave=isSaveEach,
            saveTail=curSaveTal,
            isDrawClickedPosCircle=False,
        )

    # always save final result
    curTimeStr = CommonUtils.getCurDatetimeStr("%H%M%S")
    finalSaveTail = "_rect_all_%s" % curTimeStr
    imgFolderAndName, pointSuffix = os.path.splitext(curImg
    imgFolderAndName = imgFolderAndName + finalSaveTail
    finalImgPath = imgFolderAndName + pointSuffix
    curImg.save(finalImgPath)
```

```
    return
```

调用：

```
# # for debug
# self.debugDrawScreenRect(curRect=bounds)
```

和：

```
Elements = self.get_elements_valuable(Elements)
# for debug
# self.debugDrawElementRect(Elements, isShowEach=True, isSa
self.debugDrawElementRect(Elements, isShowEach=False, isSav

Elements = [element for element in Elements if self.is_eler
# for debug
# self.debugDrawElementRect(Elements, isShowEach=True, isSa
self.debugDrawElementRect(Elements, isShowEach=False, isSav
```

用于批量给多个元素加上框，输出到单个图片中 -> 方便调试时，知道哪些元素被当前一轮循环过滤掉了

scaleToOrginSize 缩放图片到原始大小

背景：

iPhone的scale往往都是2或3等值

用iOS的截图都是大图，希望缩写到原图

```
def scaleToOrginSize(self, screenshotImgPath, curScale):
    """resize to original screen size, according to sessior
    curScreenImg = Image.open(screenshotImgPath)
    originSize = curScreenImg.size # 750x1334
    newWidthInt = int(float(originSize[0]) / curScale)
    newHeightInt = int(float(originSize[1]) / curScale)
    scaledSize = (newWidthInt, newHeightInt) # 375x667
    scaledFile = screenshotImgPath
    CommonUtils.resizeImage(curScreenImg, newSize=scaledSiz
    return scaledFile
```

调用举例：

```
curScale = 3
if isResizeToOriginal:
    # and resize to original screen size
    savedImgFile = self.scaleToOrginSize(savedImgFile, curS
```

saveiOSScreenshot 保存当前屏幕截图（图片文件）

```
def saveiOSScreenshot(self, filePrefix="", imageFormat="jpg
    """
        do screehsot for ios device and saved to jpg
            same screenshot image file size compare:
                png: 734KB
                jpg: 100KB
            so better to use jpg
    """
    savedScreenFile = None


    curDatetimeStr = CommonUtils.getCurDatetimeStr()
    # screenFilename = "%s_screen.%s" % (curDatetimeStr, im
    screenFilename = "%s.%s" % (curDatetimeStr, imageFormat
    if filePrefix:
        screenFilename = "%s_%s" % (filePrefix, screenFilen
        # 'com.netease.cloudmusic_20200221_170305.jpg'
    screenFilename = os.path.join(saveFolder, screenFilenam


    if imageFormat == "png":
        curPillowObj = self.wdaClient.screenshot(png_filena
        savedScreenFile = screenFilename
    elif (imageFormat == "jpg") or (imageFormat == "jpeg"):
        curPillowObj = self.wdaClient.screenshot()
        # logging.debug("curPillowObj=%s", curPillowObj)
        # curPillowObj=<PIL.PngImagePlugin.PngImageFile ima
        # convert to PIL.Image and then save as jpg
        curPillowObj.save(screenFilename)

        savedScreenFile = screenFilename
    else:
        logging.debug("Unsupported image format: %s", image


    if savedScreenFile:
        logging.debug("saved screenshot: %s", savedScreenFi
    return savedScreenFile
```

调用举例 + 相关函数：

```
    def debugiOSSaveScreenshot(self, saveFolder, isResizeTo
        """for debug, save iOS screenshot"""
        # # for if enable debug screenshot image content to
        # # so disable debug screenshot
        # needEnableLater = False
        # if wda.DEBUG:
        #     wda.DEBUG = False
        #     needEnableLater = True
        # savedImgFile = self.saveiOSScreenshot(filePrefix=
        # if needEnableLater:
        #     wda.DEBUG = True
        def _saveScreenshot():
            return self.saveiOSScreenshot(filePrefix="", sa
        savedImgFile = self.runButNotOutputWdaDebug(_saveSc


        logging.debug("Saved iOS screenshot %s", savedImgF
        if isResizeToOriginal:
            # and resize to original screen size
            savedImgFile = self.scaleToOrginSize(savedImgF
        return savedImgFile
```

调用：

```
        elif self.isiOS:
            fullImgFilePath = self.debugiOSSaveScreenshot(s
```

以及：

```
    def runButNotOutputWdaDebug(self, funcToRun):
        # disable then re-enable debug for
        # avoid debug print too many binary image data of c
        # for internal session scale will trigger do screen
        needEnableLater = False
        if wda.DEBUG:
            wda.DEBUG = False
            needEnableLater = True


        retValue = funcToRun()


        if needEnableLater:
            wda.DEBUG = True


        return retValue
```

源码分析

下面给出 `facebook-wda` 的部分源码的内部实现逻辑分析：

wda内部用到了Appium

文件： `refer/WebDriverAgent/WebDriverAgentLib/Commands/FBSessionCommands.m`

```
+ (NSArray *)routes
{
  return
  @[
    [[FBRoute POST:@"/url"] respondWithTarget:self action:@
    [[FBRoute POST:@"/session"].withoutSession respondWithT
    [[FBRoute POST:@"/wda/apps/launch"] respondWithTarget:s
    [[FBRoute POST:@"/wda/apps/activate"] respondWithTarget
    [[FBRoute POST:@"/wda/apps/terminate"] respondWithTarge
    [[FBRoute POST:@"/wda/apps/state"] respondWithTarget:se
    [[FBRoute GET:@"/wda/apps/list"] respondWithTarget:self
    [[FBRoute GET:@""] respondWithTarget:self action:@selec
    [[FBRoute DELETE:@""] respondWithTarget:self action:@se
    [[FBRoute GET:@"/status"].withoutSession respondWithTar

    // Health check might modify simulator state so it shou
    [[FBRoute GET:@"/wda/healthcheck"].withoutSession respo

    // Settings endpoints
    [[FBRoute GET:@"/appium/settings"] respondWithTarget:se
    [[FBRoute POST:@"/appium/settings"] respondWithTarget:s
  ];
}
```

中的 `/appium/settings` 就是Appium的。

对应着支持外部的python的wda去调用：

文件： `/Users/limao/.pyenv/versions/3.8.0/Python.framework/Versions/3.8/lib/python3.8/site-packages/wda/__init__.py`

```
def get_settings(self):
    resp = self.http.get("/appium/settings")
    respSettings = resp.value
    if DEBUG:
        # print("resp=%s" % resp) #  TypeError not all argu
        print("respSettings=%s" % respSettings)
    return respSettings


def set_settings(self, newSettings):
    postResp = self.http.post("/appium/settings", {
        "settings": newSettings,
    })
    respNewSettings = postResp.value
    if DEBUG:
        print("respNewSettings=%s" % respNewSettings)
    return respNewSettings
```

所以settings等参数，也要参考对应文档：

- GitHub中的
  - appium/settings.md at master · appium/appium
- readthedocs中的
  - Settings - appium
  - The Settings API - Appium

其中的screenshotQuality的值，是来自苹果的XCTest中的定义：

- XCTImageQuality - XCTest | Apple Developer Documentation

以及另外相关的Capabilities

- appium/caps.md at master · appium/appium
- Update Settings - Appium

而wda中其他还有地方也是用到Appium的，比如：

Attribute - Appium

```
HTTP API Specifications
Endpoint
GET /session/:session_id/elements/:element_id/attribute/:na
```

对应着能看到python的wda中就是：

文件：`/Users/limao/.pyenv/versions/3.8.0/Python.framework/Versions/3.8/lib/python3.8/site-packages/wda/__init__.py`

```
def _wdasearch(self, using, value):
    """
    Returns:
        element_ids (list(string)): example ['id1', 'id2']

    HTTP example response:
    [
        {"ELEMENT": "E2FF5B2A-DBDF-4E67-9179-91609480D80A"}
        {"ELEMENT": "597B1A1E-70B9-4CBE-ACAD-40943B0A6034"}
    ]
    """
    element_ids = []
    for v in self.http.post('/elements', {
            'using': using,
            'value': value
    }).value:
        element_ids.append(v['ELEMENT'])
    return element_ids
```

和：

文件：`refer/WebDriverAgent/WebDriverAgentLib/Commands/FBFindElementCommands.m`

```
+ (NSArray *)routes
{
  return
  @[
    [[FBRoute POST:@"/element"] respondWithTarget:self act
    [[FBRoute POST:@"/elements"] respondWithTarget:self ac
    [[FBRoute POST:@"/element/:uuid/element"] respondWithT
    [[FBRoute POST:@"/element/:uuid/elements"] respondWith
...
```

调试时开启debug后可以看到：

```
HAIN: **/XCUIElementTypeAny[`name == '通讯录'`]
Shell: curl -X POST -d '{"using": "class chain", "value": '
Return (984ms): {
  "value" : [
    {
      "ELEMENT" : "15000000-0000-0000-0502-000000000000",
      "element-6066-11e4-a52e-4f735466cecf" : "15000000-000
    }
  ],
  "sessionId" : "C69F63F6-80EA-47DD-BA04-0A912945CE39"
}
...
Shell: curl -X GET -d '' 'http://192.168.31.43:8100/session
Return (688ms): {
  "value" : {
    "y" : 619,
    "x" : 96,
    "width" : 90,
    "height" : 48
  },
  "sessionId" : "C69F63F6-80EA-47DD-BA04-0A912945CE39"
}
```

去获取element的属性值的。

附录

下面列出相关参考资料。

iOS内置app的bundle ID

用wda开发期间，常会涉及到操作app，其中需要知道内置app的bundle id。

下面是iOS 13自带的app的bundle ID：

App Name	Bundle ID
Activity	com.apple.Fitness
App Store	com.apple.AppStore
Apple Store	com.apple.store.Jolly
Books	com.apple.iBooks
Calculator	com.apple.calculator
Calendar	com.apple.mobilecal
Camera	com.apple.camera
Clips	com.apple.clips
Clock	com.apple.mobiletimer
Compass	com.apple.compass
Contacts	com.apple.MobileAddressBook
FaceTime	com.apple.facetime
Files	com.apple.DocumentsApp
Find My	com.apple.findmy
GarageBand	com.apple.mobilegarageband
Health	com.apple.Health
Home	com.apple.Home
iCloud Drive	com.apple.iCloudDriveApp
iMovie	com.apple.iMovie
iTunes Store	com.apple.MobileStore
iTunes U	com.apple.itunesu
Mail	com.apple.mobilemail
Maps	com.apple.Maps
Messages	com.apple.MobileSMS
Measure	com.apple.measure
Music	com.apple.Music
News	com.apple.news
Notes	com.apple.mobilenotes
Phone	com.apple.mobilephone
Photos	com.apple.mobileslideshow
Photo Booth	com.apple.Photo-Booth
Podcasts	com.apple.podcasts

App Name	Bundle ID
Reminders	com.apple.reminders
Safari	com.apple.mobilesafari
Settings	com.apple.Preferences
Shortcuts	com.apple.shortcuts
Stocks	com.apple.stocks
Tips	com.apple.tips
TV	com.apple.tv
Videos	com.apple.videos
Voice Memos	com.apple.VoiceMemos
Wallet	com.apple.Passbook
Watch	com.apple.Bridge
Weather	com.apple.weather

文档

facebook的WebDriverAgent

facebookarchive/WebDriverAgent: A WebDriver server for iOS that runs inside the Simulator.

已archive归档，不更新了。

-》常见问题：

Common Issues · facebookarchive/WebDriverAgent Wiki

其中提到了：

关于中国的iPhone，无法通过IP访问的问题：

Can't connect via Wifi to iOS10.x devices in China · Issue #288 · facebookarchive/WebDriverAgent

USB的支持：

USB support · facebookarchive/WebDriverAgent Wiki

-》关于Inspector：

Using the Inspector · facebookarchive/WebDriverAgent Wiki

关于内部query查询api接口：

Queries · facebookarchive/WebDriverAgent Wiki

关于Predicate Queries：

Predicate Queries Construction Rules · facebookarchive/WebDriverAgent Wiki

apple的相关资料

Predicate Format String Syntax

如何提高搜索查找的速度

How To Achieve The Best Lookup Performance · facebookarchive/WebDriverAgent Wiki

参考资料

- 【已解决】Mac中用facebook-wda自动化测试操作iOS设备
- 【已解决】Mac中用facebook-wda操作iOS真机iPhone6
- 【已解决】Mac中xcodebuild警告：xcode-select error tool xcodebuild requires Xcode
- 【未解决】自动抓包iOS的app：左滑引导页进入首页
- 【已解决】自动抓包工具适配iOS：当前检测出微信连续异常，你可以尝试一下方法修复 下一步
- 【已解决】iPhone中AppStore的bundle id即appId是啥
- 【已解决】facebook-wda中元素clear清除文本值无效
- 【已解决】facebook-wda获取鉴定的value值是错误的
- 【未解决】自动抓包iOS的app：无忧筹点击首页的筹款首页后无法返回
- 【未解决】facebook-wda点击个人所得税元素无效：没有进入AppStore详情页
- 【未解决】研究facebook-wda和WebDriverAgent中attribute/value始终是null无法获取有效值
- 【未解决】WebDriverAgent获取iPhone页面源码报错：Code 5 Error kAXErrorIPCTimeout getting snapshot for element
- 【已解决】自动抓包iOS公众号：niuer-tmall中丢失部分主菜单
- 【已解决】Python中facebook-wda和WebDriverAgent中是否可以支持displayed以及是否能替换visible
- 【已解决】自动抓包iOS公众号：niuer-tmall定位主菜单失败
- 【已解决】合并最新版WebDriverAgent后测试是否支持元素的visible属性的query查询
- 【未解决】自动抓包iOS公众号：小程序中可关闭弹框签到赢好礼
- 【规避解决】自动抓包iOS公众号：获取微信公众号unesunes搜索结果页面源码失败
- 【记录】更新WebDriverAgent后测试iOS 12的iPhone6抓包微信公众号
- 【规避解决】XCode实时调试NSXPCConnection的_XCT_fetchSnapshotForElement:attributes:parameters:reply错误
- 【规避解决】WebDriverAgent获取页面源码报错：xpc.exceptions NSXPCConnection com.apple.testmanagerd _XCT_fetchSnapshotForElement
- 【已解决】wda用source()获取页面源码xml速度极其慢
- 【已解决】把旧版WebDriverAgent自己优化改动合并到最新版代码中
- 【已解决】验证最新WebDriverAgent代码功能上是否正常
- 【已解决】XCode编译最新版WebDriverAgent

- 【已解决】用XCode实时调试WebDriverAgent希望找到并解决获取页面源码慢的原因
- 【已解决】尝试解决facebook-wda和WebDriverAgent的获取源码很慢的原因
- 【未解决】WebDriverAgent和wda获取源码提速：尝试shouldLoadSnapshotWithAttributes参数
- 【未解决】调节Appium的Capability的参数去提高facebook-wda和WebDriverAgent获取源码的速度
- 【未解决】WebDriverAgent获取源码慢尝试调节参数：shouldUseTestManagerForVisibilityDetection
- 【已解决】Xcode调试WebDriverAgent研究fb_waitUntilSnapshotIsStable含义希望提高获取源码速度
- 【已解决】WebDriverAgent报错：Internal error Error Domain com.apple.dt.xctest.automation-support.error Code 5 Error kAXErrorServerNotFound getting snapshot for element
- 【已解决】WebDriverAgent中fb_waitUntilSnapshotIsStable的作用和含义即为何加上
- 【已解决】WebDriverAgent获取源码慢尝试调节参数：FB_ANIMATION_TIMEOUT
- 【未解决】Mac中用facebook-wda自动操作安卓手机浏览器实现百度搜索
- 【已解决】Mac中安装facebook-wda依赖的一些库
- 【已解决】XCode中编译报错：A build only device cannot be used to run this target o supported iOS devices are available
- 【已解决】Mac中用weditor辅助调试iOS设备
- 【未解决】Mac中XCode编译报错：The operation couldn’t be completed. Unable to log in with account account rejected
- 【已解决】Mac中安装和初始化facebook-wda环境
- 【已解决】Mac中XCode如何编译WebDriverAgent.xcodeproj
- 【已解决】用XCode给WebDriverAgent.xcodeproj的WebDriverAgentRunner添加设置code signing代码签名
- 【已解决】Mac中XCode中WebDriverAgent编译报错：Signing for IntegrationApp requires a development team
- 【已解决】XCode中WebDriverAgent.xcodeproj自动签名报错：Failed to register bundle identifier com.facebook.WebDriverAgentRunner
- 【已解决】XCode 11中如何修改WebDriverAgentRunner的bundle identifier
-
- crifan (Crifan Li)
- 【记录】给Gitbook添加更多配置和功能
- 【已解决】提取Gitbook中Makefile公共部分
- 【已解决】gitbook中book.json中能否把公共部分提取出来
- 端口转发 · 苹果相关开发总结

- ATX 文档 - iOS 真机如何安装 WebDriverAgent · TesterHome
- ATX 系列-如何测试网易云音乐 (iOS 篇) · TesterHome
- 使用 Python 库 facebook-wda 完成网易云音乐 iOS 客户端的自动化测试 (示例) · TesterHome
- ATX 文档 - iOS 控件操作 API · TesterHome
- iOS 自动化测试 · TesterHome
- 【IOS测试】一篇读懂自动化框架WebDriverAgent – Python量化投资
- iOS真机安装WebDriverAgent | Vicの博客
-