

目录

前言	1.1
简介	1.2
搭建环境	1.3
常见问题和心得	1.3.1
开发心得	1.4
iOS的各种坑	1.5
获取源码xml	1.5.1
常用代码段	1.6
wda	1.6.1
crifan版wda	1.6.1.1
元素处理	1.6.2
iOS设备	1.6.3
app	1.6.4
app管理	1.6.4.1
首次登录引导页	1.6.4.2
微信	1.6.4.3
安全模式	1.6.4.3.1
微信公众号	1.6.4.3.2
设置	1.6.4.4
更新WiFi代理	1.6.4.4.1
AppStore	1.6.4.5
自动安装app	1.6.4.5.1
弹框处理	1.6.5
调试辅助	1.6.6
源码分析	1.7
附录	1.8
iOS内置app的bundle id	1.8.1
文档	1.8.2
参考资料	1.8.3

iOS自动化测试利器：facebook-wda

- 最新版本: v1.3
- 更新时间: 20200625

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。

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，尤其是iOS的各种坑，以及常用代码段，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。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-25 17:21:11

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](#)

- 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 XCUI Test
 - 是什么：苹果的测试框架
 - 官网文档
 - 之前：
 - XCUI Test
 - (貌似) 最新
 - [XCTest | Apple Developer Documentation](#)
- 关于：Appium
 - [Appium: Mobile App Automation](#)

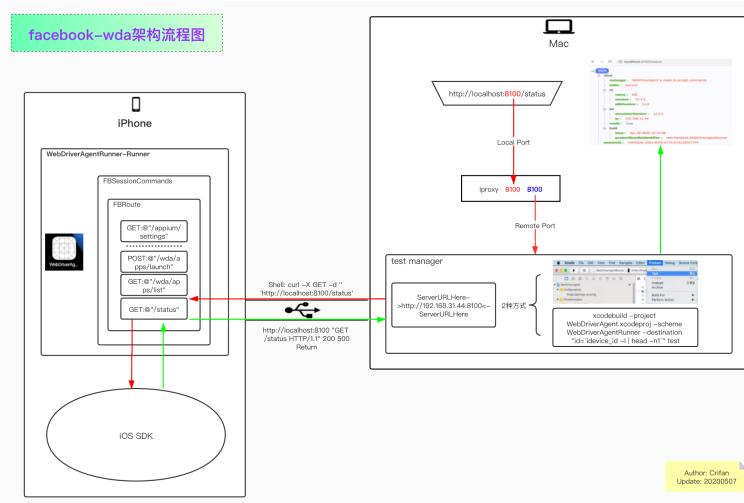
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-01 18:33:14

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
```

(2) 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
```

看到相关文件：

```
□ ll
total 96
-rw-r--r-- 1 limao CORP\Domain Users 2.0K 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 111B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 75B 2 13 17:40
drwxr-xr-x 3 limao CORP\Domain Users 96B 2 13 17:40
drwxr-xr-x 4 limao CORP\Domain Users 128B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 177B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 4.3K 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 1.5K 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 1.9K 2 13 17:40
drwxr-xr-x 7 limao CORP\Domain Users 224B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 2.5K 2 13 17:40
drwxr-xr-x 5 limao CORP\Domain Users 160B 2 13 17:40
drwxr-xr-x 5 limao CORP\Domain Users 160B 2 13 17:40
drwxr-xr-x 16 limao CORP\Domain Users 512B 2 13 17:40
drwxr-xr-x 4 limao CORP\Domain Users 128B 2 13 17:40
drwxr-xr-x 6 limao CORP\Domain Users 192B 2 13 17:40
drwxr-xr-x 5 limao CORP\Domain Users 160B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 666B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 896B 2 13 17:40
drwxr-xr-x 9 limao CORP\Domain Users 288B 2 13 17:40
-rw-r--r-- 1 limao CORP\Domain Users 2.5K 2 13 17:40
drwxr-xr-x 5 limao CORP\Domain Users 160B 2 13 17:40
```

其中核心的入口文件，即Xcode项目文件
是：`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`
- 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.xcodeproj -scheme WebDriverAgentRunner -destination "id=`idevice_id -l | head -n1`" test`
- 或分2步
 - 先获取iOS设备的UDID:
 - `CUR_UDID=$(idevice_id -l | head -n1)`
 - 再运行:
 - `xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination "id=$CUR_UDID" test`
- 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)

最后能看到输出 `ServerURLHere` 和 `Using singleton test manager` :

```
...
Test Case '-[UITestingUITests testRunner]' started.
t = 0.01s Start Test at 2020-02-20 10:50:59.818
t = 0.01s Set Up
2020-02-20 10:50:59.968359+0800 WebDriverAgentRunner-Runner
2020-02-20 10:51:00.119667+0800 WebDriverAgentRunner-Runner
2020-02-20 10:51:00.123946+0800 WebDriverAgentRunner-Runner
```

即表示正常启动了 `test manager = WDA的server` 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：

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
WebDriverAgentRunner -destination "id=`idevice_id -l | head
-n1`" test
```

会报错：

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

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] IDETestAgent[2440]: Device: 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
  deviceIsActive: YES
  deviceActivationState: Activated
  isPasscodeLocked: NO
  deviceType: <DVTDeviceType:0x7f8456...
  supportedDeviceFamilies: (
  )
  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
```

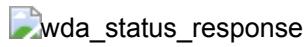
如果已端口转发则可以把IP换localhost

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 commands.",  
        "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.WebDriverAgent"  
        }  
    },  
    "sessionId": "38289A64-E467-4458-A0F1-8A3B2A6AAECE"  
}
```



crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-18 10:05:52

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：

```
gWdaClient = wda.Client('http://localhost:8100')  
gWdaClient = wda.Client('http://localhost:8101')  
gWdaClient = 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

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-02 09:53:29

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.appStoreSearchResultIntoDetail,
        "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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeCollectionView type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeButton type="XCUIElementTypeCell">
                        </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementTypeCell">
                        <XCUIElementTypeButton type="XCUIElementTypeCell">
                            </XCUIElementTypeCell>
                        </XCUIElementTypeCell>
                    </XCUIElementTypeCollectionView>
                </XCUIElementTypeCell>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    """

    搜索结果列表页 美团 获取:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeCollectionView type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeButton type="XCUIElementTypeCell">
                        </XCUIElementTypeCell>
                    <XCUIElementTypeCell type="XCUIElementTypeCell">
                        <XCUIElementTypeButton type="XCUIElementTypeCell">
                            </XCUIElementTypeCell>
                        </XCUIElementTypeCell>
                    </XCUIElementTypeCollectionView>
                </XCUIElementTypeCell>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

```

```
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    .....
parentCollectionViewClassChain = "/XCUIElementTypeCollection"
firstMatchCellQuery = {"type":"XCUIElementTypeCell", "label": "更多"}
firstMatchCellQuery["parent_class_chains"] = [ parentCollectionViewClassChain ]
foundAndClicked = self.findAndClickElement(query=firstMatchCellQuery)
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 element
```

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：

蜗牛移动 15:52

个人所得税
国家税务总局

国家税务总局发布

2.4 ★★★☆☆ 7.97万个评分 #1 财务 4+ 年龄

新功能 版本历史记录

版本 1.4.4 6 天前

1、相关专项附加扣除（大病医疗、赡养老人、继续教育）填报功能中，增加了填报提示信息。

预览

年度汇算
减税降费 让利于民

业务办理
足不出户 掌上搞定

Today 游戏 App 更新 搜索

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

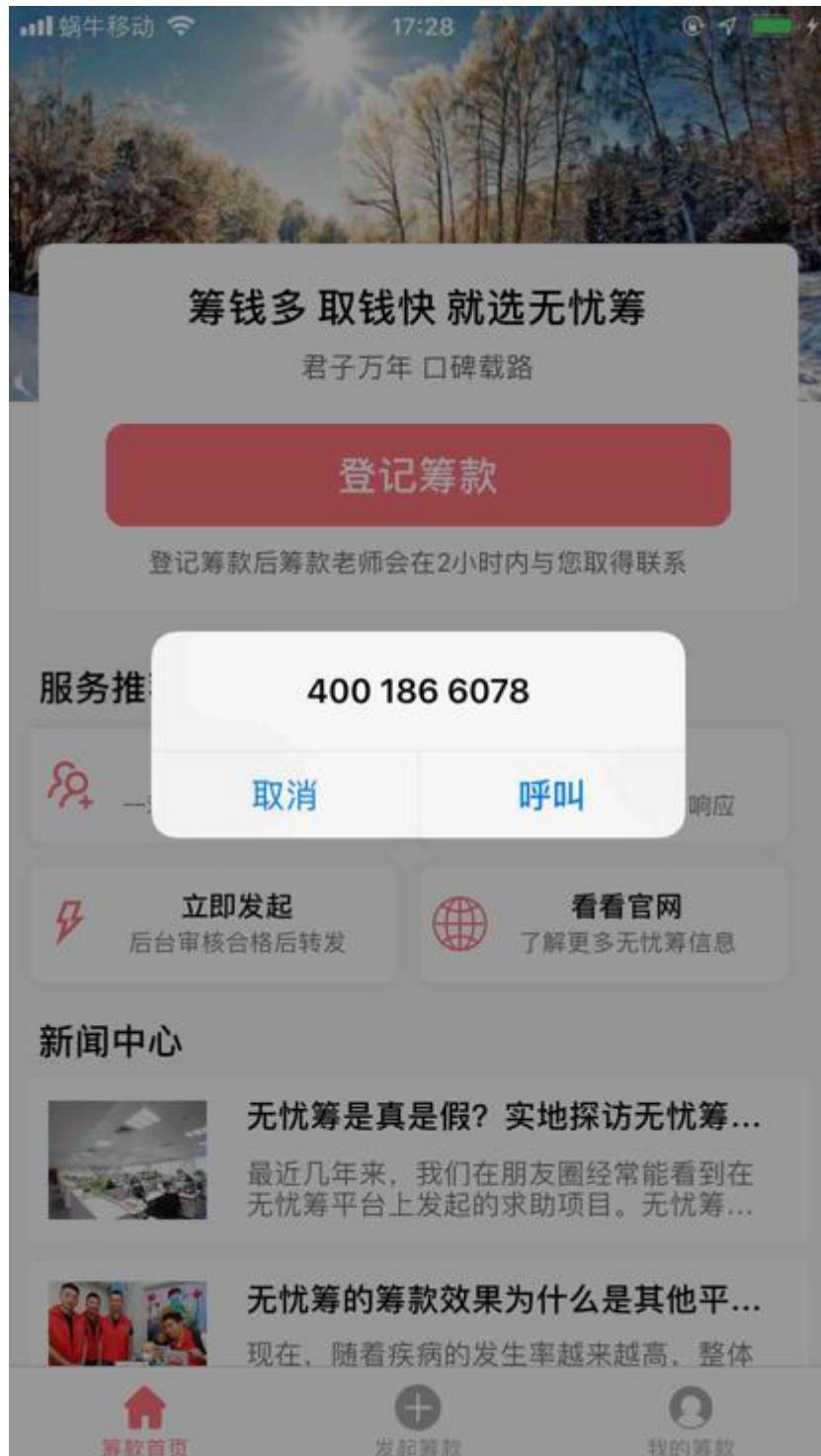
希望，或许能稳定些，或许能规避此问题？

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：



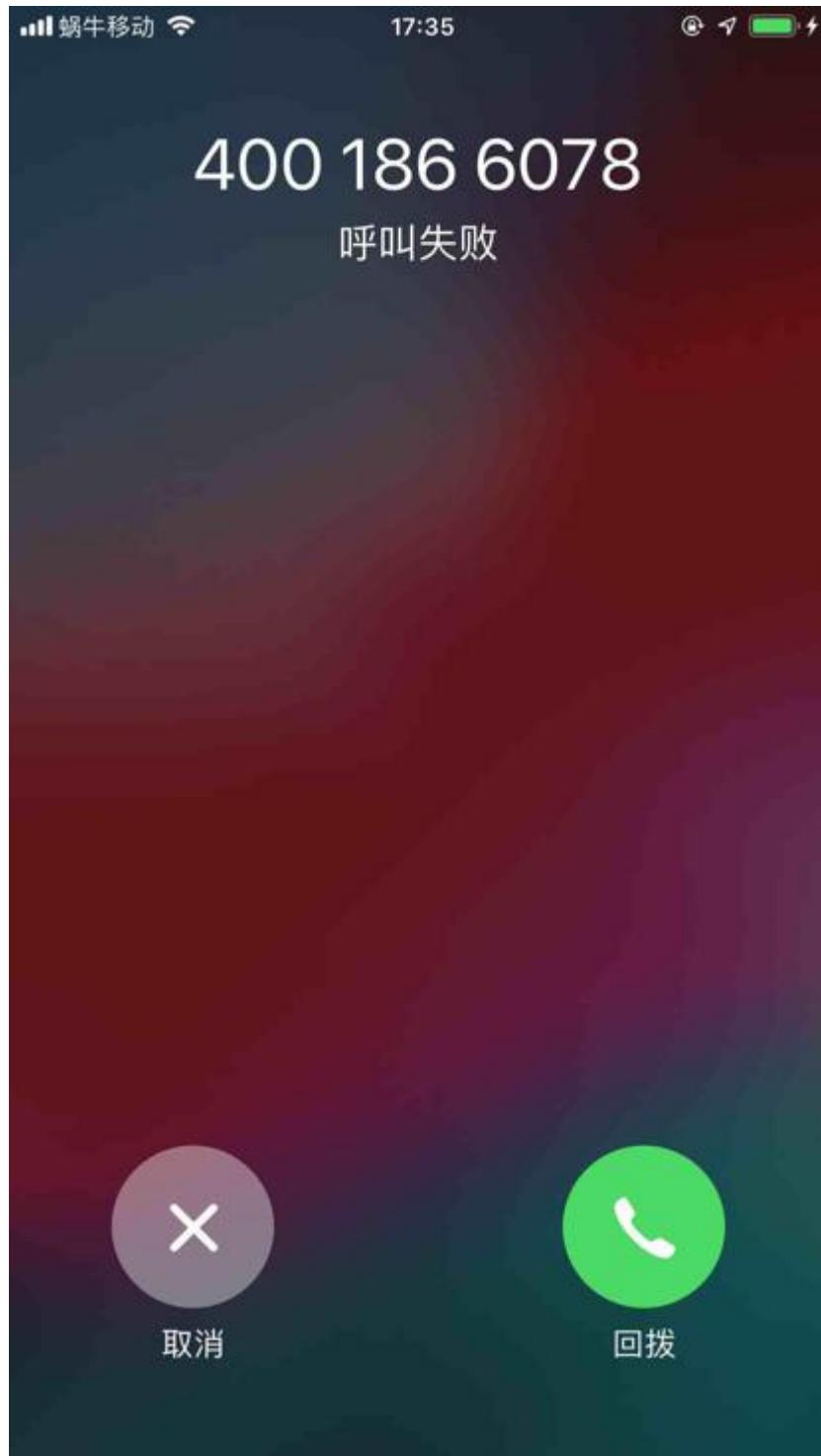
代码已找到了 取消 按钮，然后去点击 其中间坐标位置

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

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

cancelButtonQuery = {"type":"XCUIElementTypeButton"}
foundAndClicked = findAndClickElement(curSession, cancelButtonQuery)
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：



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", "name": "无线局域网与蜂窝移动网络按钮中间位置"}
    },
]
wifiCellularSoup = utils.bsChainFind(soup, wifiCellularChainList)
if wifiCellularSoup:
    clickCenterPosition(curSession, wifiCellularSoup.attrs["name"])
    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, wifiCellularChainList)
if wifiCellularSoup:
    # self.clickElementCenterPosition(wifiCellularSoup)
    # 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 element
    curName = wifiCellularSoup.attrs["name"] # 好
    wifiCellularButtonQuery = {"type": "XCUIElement", "name": curName}
    foundAndClicked = self.findAndClickElement(wifiCellularButtonQuery)
    foundAndProcessedPopup = foundAndClicked
return foundAndProcessedPopup
```

才可以。

【后记3】

由于经常遇到此问题，所以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：

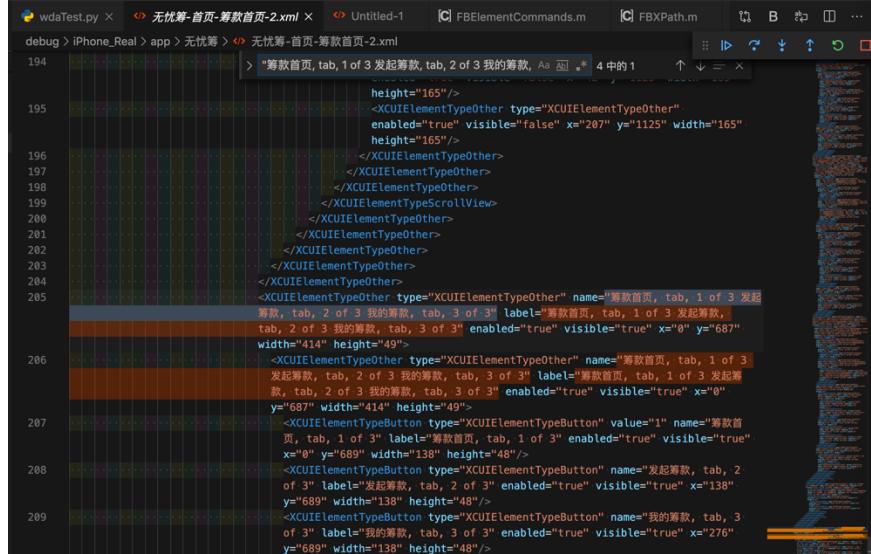


左下角的 3个tab页的父级元素，对应 locator，调试出现警告：

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

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="
```



```

194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209

```

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" >
    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true" >
        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="0" >
        <XCUIElementTypeSwitch type="XCUIElementTypeSwitch" value="1" >
    </XCUIElementTypeOther>
</XCUIElementTypeCell>
```

已经可以用代码：

```
newAuthenticateValue = newManualProxyValue["authenticate"]
authSwitchQuery = {"type": "XCUIElementTypeSwitch", "name": "authenticate"}
authSwitchQuery["parent_class_chains"] = [ parentCellClass ]
foundAuth, respInfo = self.findElement(authSwitchQuery, timer=1)
```

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": "0", "y": "0", "width": "100%", "height": "100%"}, "value": "0"
    },
    {
        "tag": "XCUIElementTypeSwitch",
        "attrs": {"enabled": "true", "visible": "true", "name": "authSwitch", "value": "0"}, "value": "0"
    }
]
authSwitchSoup = CommonUtils.bsChainFind(soup, authSwitchChainList)
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", "name": "server"}
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 way

    Args:
        curElement (Element): wda element
    Returns:
    Raises:
    """
    # curElement.click()
    # curElement.clear_text()
    # curElement.tap_hold(2.0) # then try select All -> Delete
    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。

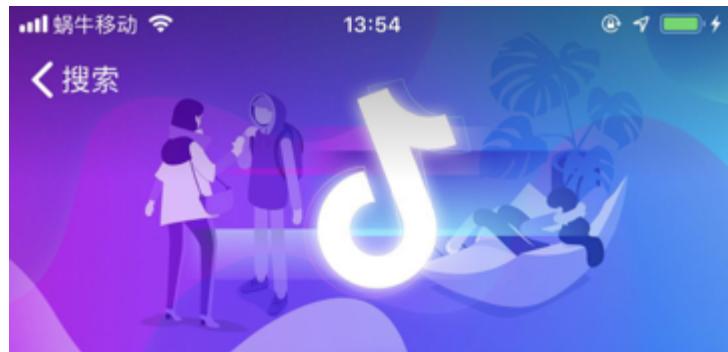
详见：

[【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



4.9 ★★★★★

2,800万个评分

#1

摄影与录像

17+

年龄

新功能

版本 11.3.0

版本历史记录

6 天前

评论区体验优化：现在你可以更好地与朋友进行互动了

预览



Today

游戏

App

更新

搜索

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"] responder]
+ (id<FBResponsePayload>)handleGetAttribute:(FBRouteRequest*)
{
    FBElementCache *elementCache = request.session.elementCache;
    XCUIElement *element = [elementCache elementForUUID:request.UUID];
    if (nil == element) {
        return FBResponseStatus([FBCommandStatus staleElement]);
    }
    id attributeValue = [element fb_valueForWDAttributeName:name];
    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://lo
20200609 01:49:34 __init__.py:178 DEBUG Return (213ms):
    "value" : {
        "element-6066-11e4-a52e-4f735466cecf" : "32000000-0
        "attribute\visible" : true,
        "attribute\name" : "正在下载",
        "attribute\value" : null,
        "attribute\accessible" : true,
        "text" : "正在下载",
        "label" : "正在下载",
        "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://lo
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="true">
    <XCUIElementTypeButton type="XCUIElementTypeButton" name="菜单" enabled="true">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            </XCUIElementTypeOther>
    </XCUIElementTypeButton>
</XCUIElementTypeCell>
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="true">
    <XCUIElementTypeButton type="XCUIElementTypeButton" name="菜单" enabled="true">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            </XCUIElementTypeOther>
    </XCUIElementTypeButton>
</XCUIElementTypeCell>
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="true">
    <XCUIElementTypeButton type="XCUIElementTypeButton" name="菜单" enabled="true">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            </XCUIElementTypeOther>
    </XCUIElementTypeButton>
</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：

```
x limao@limao:~/dev/crawler/appAutoCrawler/AppCrawler/iOSAutomation$ master$ env PTVS_LAUNCHER_PORT=50892 /Users/limao/.pyenv/versions/3.8.0/Python.framework/Versions/3.8/bin/python /Users/Limao/.vscode/extensions/ms-python.python-2020.2.64397/pythonFiles/libpython/new_ptvsd/no_wheels/ptvsd/launcher /Users/limao/dev/Fibot/crawler/appAutoCrawler/iOSAutomation/wdaTestServer$ http://192.168.31.43:8100
wdaClient=<wda.Client object at 0x10b0c41b20>
Shell: curl -X POST -d '{"desiredCapabilities": {"bundleId": "com.tencent.xin", "shouldWaitForQuiescence": false}, "capabilities": {"alwaysMatch": {"bundleId": "com.tencent.xin", "shouldWaitForQuiescence": false}}}' 'http://192.168.31.43:8100/session'
```

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 protocol error)
** TEST FAILED **
```

```
*** If you believe this error represents a bug, please attach the result bundle at /Users/limao/Library/Developer/Xcode/DerivedData/WebDriverAgent-doxykunuiuxsdzeisbkcuadwyab/Logs/Test-WebDriverAgentRunner-2020.03.13_16-31-13→0800.xcresult
2020-03-13 17:16:35.851 xcdbuild[3621:41314] [MT] IDETestOperationsDebug: 2720,400 elapsed -- Testing started completed.
2020-03-13 17:16:35.851 xcdbuild[3621:41314] [MT] IDETestOperationsObserverDebug: 0,000 sec, +0,000 sec -- start
2020-03-13 17:16:35.852 xcdbuild[3621:41314] [MT] IDETestOperationsObserverDebug: 2720,400 sec, +2720,400 sec -- end
2020-03-13 17:16:35.853 xcdbuild[3621:41314] Error Domain=com.apple.platform.iphones Code=-13 "Lost connection to DTServiceHub" UserInfo={NSLocalizedDescription=Lost connection to DTServiceHub}

Test session results, code coverage, and logs:
    /Users/limao/Library/Developer/Xcode/DerivedData/WebDriverAgent-doxykunuiuxsdzeisbkcuadwyab/Logs/Test/Test-WebDriverAgentRunner-2020.03.13_16-31-13→0800.xcresult

Testing failed:
    WebDriverAgentRunner:
        testRunner encountered an error (Encountered a problem with the test runner after launch. (Underlying error: Lost connection to DTServiceHub))

** TEST FAILED **
```

所以去重启服务：

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

```
[x limao ~] ➤ ~/dev [crawler/appAutoCrawler/AppCrawler/IDSAutomation/refer/webDriverAgent] ↵ master ➤ xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination "id=6824579634" test
2020-03-13 17:17:45.172 [xmldoc] xcdbuild[6824579634] DDDeviceKit: deviceType from ed94089f3e346538965a695b7df03d7fb3c5579 was NULL
2020-03-13 17:17:45.172 [xmldoc] xcdbuild[6824579634] DDDeviceKit: deviceType from ed94089f3e346538965a695b7df03d7fb3c5579 was NULL
note: Using new build system
note: Planning build
note: Using build description from disk
Testing started at 2020-03-13 17:17:45 +0800
```

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。

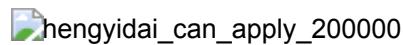
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新： 2020-06-21 22:24:35

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="1">
    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="1">
        <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
            <XCUIElementTypeImage type="XCUIElementTypeImage" name="立即进入" value="立即进入" x="350" y="450" width="200" height="50" x2="550" y2="500" alt="A large orange button labeled '立即进入' (Enter Now) with a white outline." data-bbox="350 450 550 500" data-type="XCUIElementTypeImage" data-value="立即进入" data-x="350" data-y="450" data-width="200" data-height="50" data-x2="550" data-y2="500"/>
        </XCUIElementTypeScrollView>
        <XCUIElementTypeButton type="XCUIElementTypeButton" name="立即进入" value="立即进入" x="350" y="450" width="200" height="50" x2="550" y2="500" alt="A large orange button labeled '立即进入' (Enter Now) with a white outline." data-bbox="350 450 550 500" data-type="XCUIElementTypeButton" data-value="立即进入" data-x="350" data-y="450" data-width="200" data-height="50" data-x2="550" data-y2="500"/>
    </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：



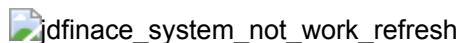
但是对应xml：

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
    <XCUIElementTypeTable type="XCUIElementTypeTable" name="刷新试试">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            <XCUIElementTypeImage type="XCUIElementTypeImage" name="刷新试试" value="刷新试试" />
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" name="刷新试试" value="刷新试试" />
            <XCUIElementTypeButton type="XCUIElementTypeButton" name="刷新试试" value="刷新试试" />
        </XCUIElementTypeOther>
    </XCUIElementTypeTable>
</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="刷新试试">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            <XCUIElementTypeImage type="XCUIElementTypeImage" name="刷新试试" value="刷新试试" />
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" name="刷新试试" value="刷新试试" />
            <XCUIElementTypeButton type="XCUIElementTypeButton" name="刷新试试" value="刷新试试" />
        </XCUIElementTypeOther>
    <XCUIElementTypeOther type="XCUIElementTypeOther" name="刷新试试" value="刷新试试" />
    <XCUIElementTypeOther type="XCUIElementTypeOther" name="刷新试试" value="刷新试试" />
</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="XCUIElementTypeApplication">
    <XCUIElementTypeWindow type="XCUIElementTypeWindow" enabled="true">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
        <XCUIElementTypeWindow type="XCUIElementTypeWindow" enabled="true">
            <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                </XCUIElementTypeWindow>
            <XCUIElementTypeWindow type="XCUIElementTypeWindow" enabled="true">
                <XCUIElementTypeStatusBar type="XCUIElementTypeStatusBar" enabled="true">
                    </XCUIElementTypeWindow>
                <XCUIElementTypeWindow type="XCUIElementTypeWindow" enabled="true">
                    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                        </XCUIElementTypeWindow>
                    </XCUIElementTypeOther>
                </XCUIElementTypeWindow>
            </XCUIElementTypeWindow>
        </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="XCUIElementTypeCell">
    <XCUIElementTypeImage type="XCUIElementTypeImage" enabled="true" visible="false" value="康爱公社-二级页面-百万医保补充互助社.jpg" />
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="康爱公社" />
    <XCUIElementTypeProgressIndicator type="XCUIElementTypeProgressIndicator" value="100%" />
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="康爱公社" />
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="康爱公社" />
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="康爱公社" />
</XCUIElementTypeCell>
```

其中

```
<XCUIElementTypeImage type="XCUIElementTypeImage" enabled="true" visible="false" value="康爱公社-二级页面-百万医保补充互助社.jpg" />
```

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

kags_second_level_page

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

kags_third_level_page

竟然其中xml源码中，还有 前一页的页面元素：

kags_third_level_xml

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
    <XCUIElementTypeStaticText type="XCUIElementTypeText" value="..."/>
</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。

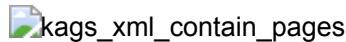
即：在第三级页面，也能找到第二级，甚至第一级页面的元素，以为是在第二级或第一级页面呢，无需返回，即可找到并点击相关元素，而实际上页面上，并不是第二级或第一级页面，屏幕上并没有这些元素。

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="page">
    ...
</XCUIElementTypeOther>
```



且不止一个：

```
<XCUIElementTypeOther type="XCUIElementTypeOther" name="page">
    ...
</XCUIElementTypeOther>
```



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：

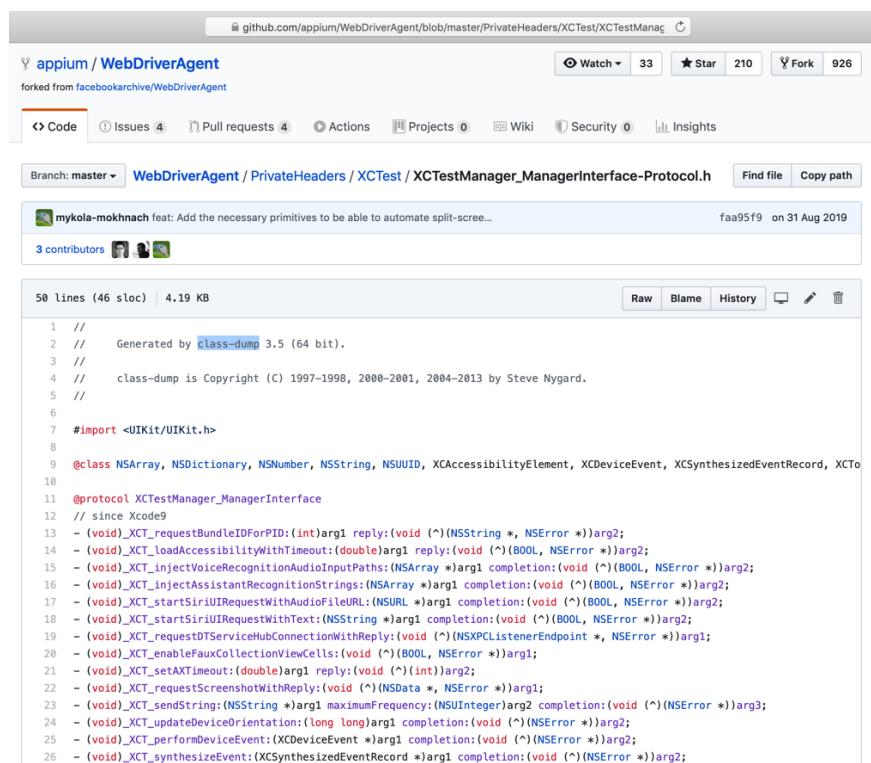
Apple方面，对于iOS设备的自动化测试，本身就是：从底层不太愿意支持

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="1">
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="1">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="非自营商品本店铺提供购买链接，若出现商品质量问...">
</XCUIElementTypeOther>
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="1">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="牛尔全球精选" style="font-size: 14px; font-weight: bold; color: #000; margin-bottom: 10px;">
</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：

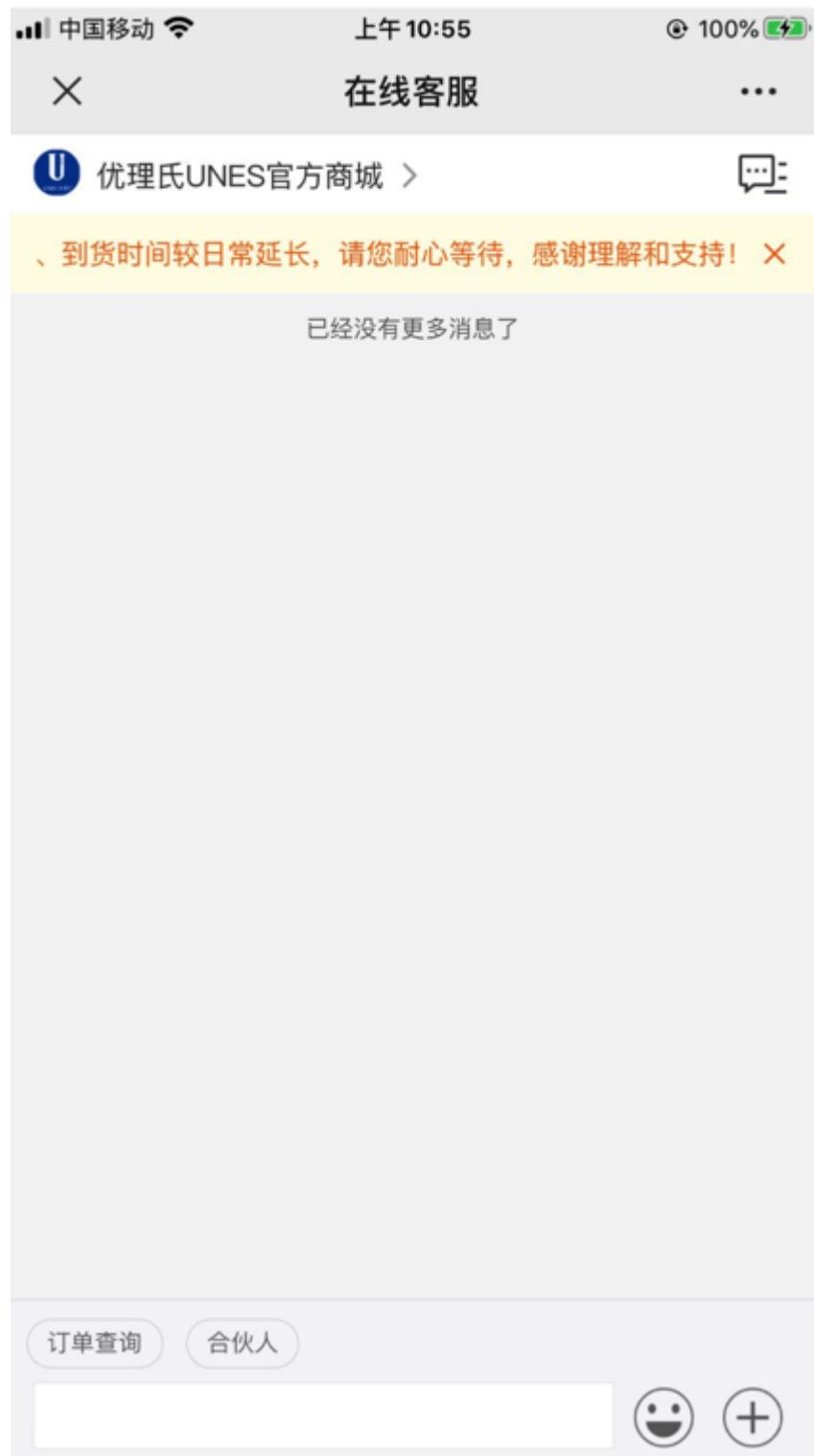
【记录】更新WebDriverAgent后测试iOS 12的iPhone6抓包微信公众号

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           0x00000001ba05e  
    1  libobjc.A.dylib         0x00000001b9d7c  
    2  Foundation             0x00000001ba54c  
    3  Foundation             0x00000001ba54c  
    4  Foundation             0x00000001ba31t  
    5  XCTAutomationSupport   0x000000010503c  
    6  Foundation             0x00000001ba54c  
    7  Foundation             0x00000001ba54c  
    8  Foundation             0x00000001ba31t  
    9  XCTAutomationSupport   0x000000010501t  
    10 Foundation            0x00000001ba54c  
    11 Foundation            0x00000001ba54c  
    12 Foundation            0x00000001ba55c  
    13 Foundation            0x00000001ba32c  
    14 Foundation            0x00000001ba33c  
    15 Foundation            0x00000001ba54c  
    16 Foundation            0x00000001ba54c  
    17 Foundation            0x00000001ba31t  
    18 XCTAutomationSupport   0x000000010501c  
    19 Foundation            0x00000001ba54c  
    20 Foundation            0x00000001ba54c  
    21 Foundation            0x00000001ba55c  
    22 Foundation            0x00000001ba32c  
    23 Foundation            0x00000001ba33c  
    24 Foundation            0x00000001ba54c  
    25 Foundation            0x00000001ba54c  
    26 Foundation            0x00000001ba31t
```

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/FBXCodeCom
patibility.m
```

中的

```
- (XCElementSnapshot *)fb_cachedSnapshot
{
    static dispatch_once_t onceToken;
    static BOOL isUniqueMatchingSnapshotAvailable;
    dispatch_once(&onceToken, ^{
        isUniqueMatchingSnapshotAvailable = [self respondsToSelector:@selector(fb_isUniqueMatchingSnapshotAvailable)];
    });
    ...
}
```

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NSXPCCConnection的
_XCT_fetchSnapshotForElement:attributes:parameters:reply错误

【规避解决】 WebDriverAgent获取页面源码报错：xpc.exceptions
NSXPCCConnection 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+

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 22:23:40

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，供需要的参考：

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, shell=True)
        try:
            resultStr = consoleOutputByte.decode("utf-8")
        except UnicodeDecodeError:
            # TODO: use chardet auto detect encoding
            # consoleOutputStr = consoleOutputByte.decode('gb18030')
            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, sleepInterval=0.5, isShowErrWhenFail=False):
    """
    do something, retry mutiple time if fail
    """

    Args:
        functionInfoDict (dict): function info dict contains function name and parameters
        maxRetryNum (int): max retry number
        sleepInterval (float): sleep time of each interval
        isShowErrWhenFail (bool): show error when fail

    Returns:
        bool
    Raises:
        None

    doSuccess = False
    functionCallback = functionInfoDict["functionCallback"]
    functionParaDict = functionInfoDict.get("functionParaDict", {})

    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.switchToApp of <DevicesMethods object at 0x107f3d0>'
            logging.error("Still fail after %d retry for %s" % (maxRetryNum, functionName))
    return doSuccess

```

isPageHasNaviBar_iOS 是否包含导航栏

```

def isPageHasNavBar_iOS(self, page):
    """Check whether current page has XCUIElementTypeNavigationBar
    """
    hasNavBar = False
    naviBarName = ""
    .....
    has:
        <XCUIElementTypeNavigationBar type="XCUIElementType">
            <XCUIElementTypeOther type="XCUIElementType">
                <XCUIElementTypeButton type="XCUIElementType">
                    ...
                </XCUIElementTypeButton>
            </XCUIElementTypeOther>
        <XCUIElementTypeNavigationBar type="XCUIElementType">
            <XCUIElementTypeOther type="XCUIElementType">
                <XCUIElementTypeButton type="XCUIElementType">
                    ...
                </XCUIElementTypeButton>
            </XCUIElementTypeOther>
        <XCUIElementTypeNavigationBar type="XCUIElementType">
            <XCUIElementTypeButton type="XCUIElementType">
                <XCUIElementTypeOther type="XCUIElementType">
                    <XCUIElementTypeButton type="XCUIElementType">
                        ...
                    </XCUIElementTypeButton>
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
        </XCUIElementTypeNavigationBar>

    某个公众号: 动卡空间
    <XCUIElementTypeNavigationBar type="XCUIElementType">
        <XCUIElementTypeButton type="XCUIElementType">
            <XCUIElementTypeOther type="XCUIElementType">
                <XCUIElementTypeButton type="XCUIElementType">
                    ...
                </XCUIElementTypeButton>
            </XCUIElementTypeOther>
        </XCUIElementTypeButton>
    </XCUIElementTypeNavigationBar>
    not:
        <XCUIElementTypeNavigationBar type="XCUIElementType">
            <XCUIElementTypeButton type="XCUIElementType">
                <XCUIElementTypeOther type="XCUIElementType">
                    <XCUIElementTypeButton type="XCUIElementType">
                        ...
                    </XCUIElementTypeButton>
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
        </XCUIElementTypeNavigationBar>
    .....
    soup = CommonUtils.xmlToSoup(page)
    foundNavBar = soup.find(
        'XCUIElementTypeNavigationBar',
        attrs={"type": "XCUIElementTypeNavigationBar", "end": "1"})
    if foundNavBar:
        # maybeFakeNavBarName = foundNavBar.attrs["name"]
        maybeFakeNavBarName = foundNavBar.attrs.get("name")
        typeOtherNameP = re.compile("%s,?" % maybeFakeNavBarName)
        foundTypeOther = foundNavBar.find("XCUIElementTypeOther")
        if foundTypeOther:
            hasNavBar = True
            naviBarName = maybeFakeNavBarName

```

```
return hasNavBar, naviBarName
```

调用：

```
OfflinePageNavBarNameList = [
    "微信",
    "通讯录",
    "公众号",
]
hasNavBar, naviBarName = self.isPageHasNavBar_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", sourceFormat)
    afterGetTime = time.time()
    getSourceTime = afterGetTime - beforeGetTime
    logging.info("Cost %.2f seconds to get iOS page source" % getSourceTime)
    return pageSource
```

调用：

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

back_iOS 返回前一页

```

def back_iOS(self):
    isFoundAndClicked = False
    # iOS: NO physical back button
    # so try support following case to find and click to it

    backQueryList = []

    parentNavBarClassChain = "/XCUIElementTypeNavigationBar"

    # case 1: 左上角 导航栏 返回
    .....
    <XCUIElementTypeNavigationBar type="XCUIElementTypeNavigationBar">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeOther type="XCUIElementTypeOther" value="返回" />
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            </XCUIElementTypeNavigationBar>
    .....
    NaviBarReturnText = "返回"
    returnButtonQuery = {"type": "XCUIElementTypeButton", "value": "返回"}
    returnButtonQuery["parent_class_chains"] = [parentNavBarClassChain]
    backQueryList.append(returnButtonQuery)

    .....
    善友筹 二级页面 小满 左上角返回按钮:
    <XCUIElementTypeNavigationBar type="XCUIElementTypeNavigationBar">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeOther type="XCUIElementTypeOther" value="返回" />
        </XCUIElementTypeNavigationBar>
    .....
    NaviBarFanhuiText = "fanhui"
    fanhuiButtonQuery = {"type": "XCUIElementTypeButton", "value": "返回"}
    fanhuiButtonQuery["parent_class_chains"] = [parentNavBarClassChain]
    backQueryList.append(fanhuiButtonQuery)

    .....
    恒易贷 验证码登录页面 左上角 返回:
    <XCUIElementTypeNavigationBar type="XCUIElementTypeNavigationBar">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeOther type="XCUIElementTypeOther" value="返回" />
        </XCUIElementTypeNavigationBar>
    .....
    NaviBarBackText = "back"
    naviContainBackQuery = {"type": "XCUIElementTypeButton", "value": "返回"}
    naviContainBackQuery["parent_class_chains"] = [parentNavBarClassChain]
    backQueryList.append(naviContainBackQuery)

    # case 2: 左上角 导航栏 关闭
    .....
    <XCUIElementTypeNavigationBar type="XCUIElementTypeNavigationBar">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeOther type="XCUIElementTypeOther" value="关闭" />
        </XCUIElementTypeNavigationBar>

```

```

        <XCUIElementTypeButton type="XCUIElementTypeButton"
        </XCUIElementTypeNavigationBar>
        ....
NavBarCloseText = "关闭"
closeButtonQuery = {"type": "XCUIElementTypeButton", "name": "关闭"}
closeButtonQuery["parent_class_chains"] = [ parentNavBar]
backQueryList.append(closeButtonQuery)

# case 3: 公众号搜索页的左上角的返回
.....
<XCUIElementTypeImage type="XCUIElementTypeImage" >
    <XCUIElementTypeButton type="XCUIElementTypeButton"
    <XCUIElementTypeOther type="XCUIElementTypeOther"
    .....
    </XCUIElementTypeOther>
</XCUIElementTypeImage>
.....
SearchReturnName = "返回"
parentImageClassChain = "/XCUIElementTypeImage[`rect.w:"
searchReturnQuery = {"type": "XCUIElementTypeButton", "name": "返回"}
searchReturnQuery["parent_class_chains"] = [ parentImage]
backQueryList.append(searchReturnQuery)

# case 3.1: app 益路通行 左上角 返回 按钮, name中包含back
.....
<XCUIElementTypeOther type="XCUIElementTypeOther"
    <XCUIElementTypeButton type="XCUIElementTypeButton"
    <XCUIElementTypeOther type="XCUIElementTypeOther"
        <XCUIElementTypeImage type="XCUIElementTypeImage"
            <XCUIElementTypeTextField type="XCUIElementTypeTextFie
        </XCUIElementTypeOther>
        <XCUIElementTypeButton type="XCUIElementTypeButton"
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
.....
<XCUIElementTypeOther type="XCUIElementTypeOther"
    <XCUIElementTypeButton type="XCUIElementTypeButton"
    <XCUIElementTypeOther type="XCUIElementTypeOther"
        <XCUIElementTypeImage type="XCUIElementTypeImage"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeImage type="XCUIElementTypeImage"
                    </XCUIElementTypeOther>
    </XCUIElementTypeOther>
</XCUIElementTypeOther>
.....
NameContainBackText = "back"
parentOtherClassChain = "/XCUIElementTypeOther[`rect.w:"
backReturnQuery = {"type": "XCUIElementTypeButton", "name": "返回"}
backReturnQuery["parent_class_chains"] = [ parentOther]
backQueryList.append(backReturnQuery)

# case 4: 全屏显示的图片 full screen image, click any pos
.....
<XCUIElementTypeScrollView type="XCUIElementTypeScrollView"

```

```

<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
    <XCUIElementTypeImage type="XCUIElementTypeImage">
        </XCUIElementTypeScrollView>
</XCUIElementTypeScrollView>

<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
    <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
        <XCUIElementTypeImage type="XCUIElementTypeImage">
            </XCUIElementTypeScrollView>
        </XCUIElementTypeScrollView>
    </XCUIElementTypeScrollView>

```

.....

```

<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
    <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
        <XCUIElementTypeImage type="XCUIElementTypeImage">
            </XCUIElementTypeScrollView>
        </XCUIElementTypeScrollView>
    </XCUIElementTypeScrollView>

```

.....

```

FullScreenImageText = "关闭"
parentParentScrollViewClassChain = "/XCUIElementTypeScrollView[`enabled`]"
parentScrollViewClassChain = "/XCUIElementTypeScrollView[`enabled`][`type`='XCUIElementTypeImage']"
fullScreenImageQuery = {"type": "XCUIElementTypeImage", "label": "关闭"}
fullScreenImageQuery["parent_class_chains"] = [parentParentScrollViewClassChain]
backQueryList.append(fullScreenImageQuery)

# case 5: 小程序页面 左上角 返回 按钮


.....


<XCUIElementTypeOther type="XCUIElementTypeOther">
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            </XCUIElementTypeOther>
        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeButton type="XCUIElementTypeButton">
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                        </XCUIElementTypeOther>
                    </XCUIElementTypeButton>
                </XCUIElementTypeOther>
            </XCUIElementTypeStaticText>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

```

.....

```

MiniprogramBackButtonText = "返回"
parentParentOtherClassChain = "/XCUIElementTypeOther[`enabled`]"
parentOtherClassChain = "/XCUIElementTypeOther[`enabled`][`type`='XCUIElementTypeImage']"
miniprogramBackButtonQuery = {"type": "XCUIElementTypeImage", "label": "返回"}
miniprogramBackButtonQuery["parent_class_chains"] = [parentParentOtherClassChain]
backQueryList.append(miniprogramBackButtonQuery)

# case 6: 小程序页面 右上角 关闭 按钮


.....


<XCUIElementTypeOther type="XCUIElementTypeOther">
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

```

```

</XCUIElementTypeOther>
.....
MiniprogramCloseButtonText = "关闭"
parentParentOtherClassChain = "/XCUIElementTypeOther[`enabled`"
parentOtherClassChain = "/XCUIElementTypeOther[`enabled`"
miniprogramCloseButtonQuery = {"type": "XCUIElementType"
miniprogramCloseButtonQuery["parent_class_chains"] = [
backQueryList.append(miniprogramCloseButtonQuery)

# case 6.1: 京东金融 瑞幸咖啡页 图片全屏
.....
<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
    <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
        <XCUIElementTypeButton type="XCUIElementTypeImage">
        <XCUIElementTypeButton type="XCUIElementTypeImage">
        <XCUIElementTypeImage type="XCUIElementTypeImage">
        <XCUIElementTypeImage type="XCUIElementTypeImage">
            <XCUIElementTypeOther type="XCUIElementTypeImage">
        </XCUIElementTypeImage>
    </XCUIElementTypeScrollView>
</XCUIElementTypeScrollView>
.....
FullScreenImagePartName = "centerSlider"
parentParentScrollViewClassChain = "/XCUIElementTypeScrollView[`isAccessibilityElement`"
parentScrollViewClassChain = "/XCUIElementTypeScrollView[`isAccessibilityElement`"
fullScreenImageQuery = {"type": "XCUIElementTypeImage"}
fullScreenImageQuery["parent_class_chains"] = [ parentScrollViewClassChain ]
backQueryList.append(fullScreenImageQuery)

# find and click
for eachBackQuery in backQueryList:
    # isCurFound, respInfo = self.findElement(query=eachBackQuery)
    SearchBackTimeout = 0.5
    isCurFound, respInfo = self.findElement(query=eachBackQuery)
    logging.debug("eachBackQuery=%s -> isCurFound=%s", eachBackQuery, isCurFound)
    if isCurFound:
        curElement = respInfo
        clickOk = self.clickElement(curElement)
        isFoundAndClicked = clickOk

        if isFoundAndClicked:
            break

    if not isFoundAndClicked:
        isFoundAndClicked = self.iOS_special_back_BiYao_Tat

if not isFoundAndClicked:
    # try find back area element then click
    page = self.get_page_source()
    # headElementList = self.get_BaseElements(page, head)
    # for eachHeadElement in headElementList:
    #     isInBackArea = self.is_element_InCertainArea(eachHeadElement)

```

```
#     isVisible = self.is_element_visible(eachHead)
#     if isInBackArea and isVisible:
#         self.clickElementCenterPosition(eachHead)
#         isFoundAndClicked = True
#         break

# backElement = self.findUpperLeftBackTypeButtonElement()
backElement, nextPage = self.findRealBackElement(page)
# if backElement:
if backElement is not None:
    isFoundAndClicked = self.clickElementCenterPosition(backElement)

return isFoundAndClicked
```

说明：

正常来说，iOS的app的页面，返回前一页，都是点击左上角的返回

但是很多页面比较特殊，其xml内容有各种类型，所以就有了上述的代码，为了兼容各种页面的返回，支持各种特殊格式

相关代码：

```

def findRealBackElement(self, page):
    """
    对于正常情况下，就是找左上角返回按钮区域内容的按钮，就是真正的返回按钮
    对于个别特殊页面，比如 必要app中商品详情页(其中带 下拉返回商品)
    左上角按钮，第一次点击，只是 相当于top按钮，返回页面顶部
    第二次点击返回按钮，才是真正的返回
    """

    newPage = page
    backElement = self.findUpperLeftBackTypeButtonElement()
    if self.isiOS:
        # if backElement:
        # foundBack = bool(backElement) # if text=None, will be False
        # foundBack = hasattr(backElement, "attrib") # lxml
        foundBack = backElement is not None
    if foundBack:
        # 检查是否是特殊的情况：必要app中商品详情页(其中带 下拉返回商品)
        DropDownReturnDetailStr = "下拉返回商品详情"
        if DropDownReturnDetailStr in page:
            # <XCUIElementTypeStaticText type="XCUIElementTypeTextElement">
            dropDownReturnDetailQuery = {"type": "XCUIElementTypeTextElement", "label": DropDownReturnDetailStr}
            isFound, foundElement = self.findElement(dropDownReturnDetailQuery)
            if isFound:
                isFoundAndClicked = self.clickElement(foundElement)
                if isFoundAndClicked:
                    newPage = self.get_page_source()
                    backElement = self.findUpperLeftBackTypeButtonElement()
    return backElement, newPage

```

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" id="..."/>
    # <XCUIElementTypeButton type="XCUIElementTypeButton" id="..."/>
    # searchButtonQuery = {"name": "Search"}
    searchButtonQuery = {"name": "Search", "type": "XCUIElementTypeButton"}
    # Note: occasionally not found Search, change to find more
    MaxRetryNumber = 5
    curRetryNumber = MaxRetryNumber
    while curRetryNumber > 0:
        foundAndClickedDoSearch = self.findAndClickElement(
            searchButtonQuery)
        if foundAndClickedDoSearch:
            break
        curRetryNumber -= 1

    if not foundAndClickedDoSearch:
        logging.error("Not found and/or clicked for %s", self)
    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 clear
    Returns:
    Raises:
    """
    isInputOk = False
    isFound, respInfo = self.findElement(query=query)
    logging.debug("isFound=%s, respInfo=%s", isFound, respInfo)
    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 way
    """
    Args:
        curElement (Element): wda element
    Returns:
    Raises:
    """
    # curElement.click()
    # curElement.clear_text()
    # curElement.tap_hold(2.0) # then try select All -> Delete
    backspaceChar = '\b'
    maxDeleteNum = 50
    curElement.set_text(maxDeleteNum * backspaceChar)
    return

```

调用举例：

(1)

对于xml

```
<XCUIElementTypeCell type="XCUIElementTypeCell" enabled="true">
    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
        <XCUIElementTypeTextField type="XCUIElementTypeTextField" name="url">
            <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                    </XCUIElementTypeOther>
    </XCUIElementTypeCell>
```

调用：

```
newUrlValue = newAutoProxyValue["url"]
parentCellClassChain = "/XCUIElementTypeCell[`rect.x = 0 And `rect.y = 0 And `label = 'URL' And `type = 'XCUIElementTypeText Field']"
urlFieldQuery = {"type": "XCUIElementTypeTextField", "name": "url"}
urlFieldQuery["parent_class_chains"] = [parentCellClassChain]
# foundUrl, respInfo = self.findElement(urlFieldQuery, timeOut=10)
# if not foundUrl:
#     return False
isInputUrlOk = self.wait_element_setText_iOS(urlFieldQuery, newUrlValue)
```

(2)

```
newAuthUserValue = newManualProxyValue["authUser"]
userFieldQuery = {"type": "XCUIElementTypeTextField", "name": "username"}
userFieldQuery["parent_class_chains"] = [parentCellClassChain]
isInputUserOk = self.wait_element_setText_iOS(userFieldQuery, newAuthUserValue)
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 13:42:45

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["WdaServerPort"], self.config["WdaServerPort"])
    wdaServerUrl = 'http://%s:%s' % (self.config["wda"]["serverPort"], self.config["wda"]["serverPort"])
    logging.info("wdaServerUrl=%s", wdaServerUrl) # 'http://127.0.0.1:8100'
    self.wdaClient = wda.Client(wdaServerUrl)
    logging.info("self.wdaClient=%s", self.wdaClient)

    self.curSession = self.wdaClient

    # init settings
    # newSettings = {
    #     "snapshotTimeout": self.config["WdaSnapshotTimeout"],
    #     "snapshotQuality": self.config["WdaScreenshotQuality"],
    #     "mjpecfgScalingFactor": self.config["WdaScalingFactor"],
    #     "includeNonModalElements": self.config["WdaIncludeNonModalElements"],
    #     "shouldUseCompactResponses": self.config["WdaShouldUseCompactResponses"],
    #     "elementResponseAttributes": self.config["WdaElementResponseAttributes"]
    # }
    newSettings = {
        "snapshotTimeout": self.config["wda"]["settings"]["snapshotTimeout"],
        "snapshotQuality": self.config["wda"]["settings"]["snapshotQuality"],
        "mjpecfgScalingFactor": self.config["wda"]["settings"]["mjpecfgScalingFactor"],
        "includeNonModalElements": self.config["wda"]["settings"]["includeNonModalElements"],
        "shouldUseCompactResponses": self.config["wda"]["settings"]["shouldUseCompactResponses"],
        "elementResponseAttributes": self.config["wda"]["settings"]["elementResponseAttributes"]
    }
    respNewSettings = self.curSession.set_settings(newSettings)
    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:
        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, respSessionId=%s", respStatus, respValue, respSessionId)
    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=%s", respStatus, respValue, respSessionId)
# if respStatus == 0:
#     isTerminalOk = True
#     respInfo = None
# else:
#     errInfo = {
#         "status": respStatus,
#         "value": respValue,
#     }
#     respInfo = errInfo
isTerminalOk, respInfo = self.processWdaResponse(terminateRes
```

(3)

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

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-21 11:11:38

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_UPSIDE_DOWN'

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/xcuiapplication
class ApplicationState(Enum):
    Unknown = 0
    NotRunning = 1
    RunningBackgroundSuspended = 2
    RunningBackground = 3
    RunningForeground = 4

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

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

    def __str__(self):
        return 'WDAResponseError(status=%d, value=%s)' % (self.status, self.value)

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 urls]).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}'"
              .format(method=method.upper(), body=body or '', url=url))

    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))

    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("error"):
            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://localhost:4200
        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,
        #               data)

    def _fetch_no_alert(self, method, url, data=None, depth=0):
        target_url = urljoin(self.address, url)
        try:
            return httpdo(target_url, method, data)
        except WDAResponseError 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, depth+1)

    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})'.format(
            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 + self.width / 2,
                                              self.y + self.height / 2)

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

@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://localhost:4723')
        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')
    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:
        WDARRequestError
    """
    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_filename:
        raise WDARRequestError(-1, "screenshot png format error")

    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.com']
        - enviroment (dict): {"KEY": "VAL"}
        - alert_action (str): "accept" or "dismiss"

    WDA Return json like

    {
        "value": {
            "sessionId": "69E6FDBA-8D59-4349-B7DE-A9CA"
    }

```

```

    "capabilities": {
        "device": "iphone",
        "browserName": "部落冲突",
        "sdkVersion": "9.3.2",
        "CFBundleIdentifier": "com.supercell.ma
    },
},
"sessionId": "69E6FDBA-8D59-4349-B7DE-A9CA41A9",
"status": 0
}

```

To create a new session, send json data like

```

{
    "desiredCapabilities": {
        "bundleId": "your-bundle-id",
        "app": "your-app-path"
        "shouldUseCompactResponses": (bool),
        "shouldUseTestManagerForVisibilityDetection": (bool),
        "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 showAlerts to True
    # 'useJSONSource': True,
    # 'simpleIsVisibleCheck': True,
    # 'shouldUseTestManagerForVisibilityDetection': True
}
# Remove empty value to prevent WDAResponseError
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_act

    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 response
        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.sessionId)

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

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.__session_id)

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

    Refs:
        https://developer.apple.com/library/archive/documentation/General/Conceptual/Devpedia-CocoaAppKit/UIKit.html#//apple_ref/doc/uid/TP40009379-CH1-SW1
    There is another way to get scale
        self.http.get("/wda/screen").value returns {"scale": 1.0}
    """
    v = max(self.screenshot().size) / max(self.window_size())
    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=3
        https://developer.apple.com/documentation/ios_ipados/reference/wda/batteryinfo
    """
    return self._session_http.get("/wda/batteryInfo").value

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': 'Asia/Shanghai'}
    """
    return self._session_http.get("/wda/device/info").value

def screen_info(self):
    """get screen info
       https://developer.apple.com/library/archive/documentation/DeviceInformation/Reference/iOSDeviceCompatibility/Screen/Screen.html#//apple_ref/doc/uid/TP40009631-SW1
    Returns dict:
        eg:
            {'statusBarSize': {'width': 375, 'height': 20}}
            -> updated wda code, can return more info:
            {'statusBarSize': {'width': 375, 'height': 20},
             '...': ...}
    """
    if DEBUG:
        print("self._session_http=%s" % self._session_http)

    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/settings",
                                      {"settings": newSettings,
                                       })
    respNewSettings = postResp.value
    if DEBUG:
        print("respNewSettings=%s" % respNewSettings)

```

```

        return respNewSettings

    def setSnapshotTimeout(self, newSnapshotTimeout):
        """set new snapshotTimeout value for current session
        newSettings = {
            "snapshotTimeout": newSnapshotTimeout
        }
        return self.set_settings(newSettings)
    def set_clipboard(self, content, content_type="plain_text"):
        """ set clipboard """
        self._session_http.post(
            "/wda/setPasteboard", {
                "content": base64.b64encode(content.encode("utf-8")),
                "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(callback)
        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.com']
            - environment (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_quiescence
    })
}

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-F7CCF...
    }

    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").value

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

```

```

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=x, y=y))

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

    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', data={'x': x, 'y': y})

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

    Args:
        - x, y(int, float): float(percent) or int(absolutely)
        - duration(float): seconds of hold time
    """
    [[FBRoute POST:@"/wda/touchAndHold"] respondWithTap]
    """
    x, y = self._percent2pos(x, y)
    data = {'x': x, 'y': y, 'duration': duration}
    return self._session_http.post('/wda/touchAndHold', data)

def swipe(self, x1, y1, x2, y2, duration=0):
    """
    Args:
        x1, y1, x2, y2 (int, float): float(percent), if
        duration (float): start coordinate press duration
    """
    [[FBRoute POST:@"/wda/dragfromtoforduration"] respondWithSwipe]
    """
    if any(isinstance(v, float) for v in [x1, y1, x2, y2]):
        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, duration=duration)
    return self._session_http.post('/wda/dragfromtoforduration', data)

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 | UIA_DEVICE_ORIENTATION_PORTRAIT_UPSIDE
    """
    return self._session_http.post('orientation', data=value)

def window_size(self):
    """
    Returns:
        namedtuple: eg
            Size(width=320, height=568)
    """
    value = self._session_http.get('/window/size').value
    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, **kwargs)

@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 library")

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

Session = Client # for compatibility

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

```

```

@property
def exists(self):
    try:
        self.text
    except WDAResponseError 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').value

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/buttons')
# 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": button_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/WebD
        # rect - Element's rectangle as a dictio
        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 t:
    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 dictio
    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, e
    xpath (str): xpath string, a little slow, but v

```

```

        timeout (float): maximum wait element time, default 10
        index (int): index of founded elements

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

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, session=%s")
    print("Selector __init__: className=%s, type=%s")
    print("Selector __init__: text=%s, textContains=%s")
    print("Selector __init__: label=%s, labelContains=%s")
    print("Selector __init__: rect=%s, x=%s, y=%s, width=%s, height=%s")
    print("Selector __init__: classChain=%s, xpath=%s")

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_parameter(
    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.startswith("XCUIElementType"):
            self.class_name = "XCUIElementType" + self.class_name[1:]
    
```

```

        'XCUIElementType'):
    self.class_name = 'XCUIElementType' + self.class_name
    if self.name_regex:
        if not self.name_regex.startswith(
            '^') and not self.name_regex.startswith(
                '$') and not self.name_regex.endswith(
                    '$') and not self.name_regex.endswith(
                        '$'):
            self.name_regex = '.*' + self.name_regex
        if not self.name_regex.endswith(
            '$') and not self.name_regex.endswith(
                '$'):
            self.name_regex = self.name_regex + '.*'
    if DEBUG:
        print("after update: self.name_regex=%s" % self.name_regex)
    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" % (inputValue, respBool))

    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 + ')', '/XCUIElement$',
                 s)

def _add_escape_character_for_quote_prime_character(self, text):
    """Fix for https://github.com/openatx/facebook-wda/issues/100
    Returns:
        string with properly formated quotes, or non char
    """
    if text is not None:
        if '"' in text:
            return text.replace('"', '\\"')
        elif '\'' in text:
            return text.replace('\'', '\\\'')
        else:
            return text
    else:
        return None

```

```
        return text

def _wdasearch_single(self, using, value):
    """
    Returns:
        bool, str
        True, element info:
        {
            "id": "0F000000-0000-0000-D703-000000000000",
            "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 element",
            "traceback" : "(\n\t0 WebDriverA...
        }
    """

    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 WDAResponseError as wdaReqErr:
        isException = True
        if DEBUG:
            print("wdaReqErr=%s" % wdaReqErr)

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



```

EmptyId = "00000000-0000-0000-0000-000000000000"

if elementId == EmptyId:
    isFound = False
    respInfo = {
        "error": "empty id",
        "message": "found element but id is empty"
    }
else:
    .....
    {
        "value": {
            "element-6066-11e4-a52e-4f7354c766d1": {
                "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": "19010000-0000-0000-0000-000000000000"
            },
            "sessionId": "DCF1A843-9FB9-4415-8E8D-000000000000"
        }
        .....
        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] = eachValue

```

```

        respInfo = elementInfo

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

    # return elementId
    return isFound, respInfo

def _wdasearch(self, using, value):
"""
    Returns:
        element_ids (list(string)): example ['id1', 'id2']

    HTTP example response:
    [
        {"ELEMENT": "E2FF5B2A-DBDF-4E67-9179-91609480D6C8"},
        {"ELEMENT": "597B1A1E-70B9-4CBE-ACAD-40943B0A6C8"}
    ]
"""

    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 already 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('urllib.parse.quote')
    # if DEBUG:
    #     print("self.name_regex=%s, escapedNameRegex=%s" % (self.name_regex, escapedNameRegex))
    # qs.append("name MATCHES '%s'" % escapedNameRegex)
    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_part)
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_part)
if self.value_regex:
    # escapedValueRegex = self.value_regex.encode('urllib.parse.quote')
    # if DEBUG:
    #     print("self.value_regex=%s, escapedValueRegex=%s" % (self.value_regex, escapedValueRegex))
    # qs.append("value MATCHES '%s'" % escapedValueRegex)
    qs.append("value MATCHES '%s'" % self.value_regex)
    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 else BoolFalseValue
    enableKey = "enabled"
    # enableKey = "wdEnabled"
    qs.append("%s == %s" % (enableKey, enabledValue))

# 20200427: after merge latest WebDriverAgent, consider visible
if DEBUG:
    print("self.visible=%s" % self.visible)
if self.visible is not None:
    visibleKey = "visible"
    # Note: WebDriverAgent says support visible, but not accessible
    # so temp change to (maybe similar meaning) & make it consistent with visible
    # 20200402: sometime use visible or accessible
    # visibleKey = "accessible"

```

```

# 20200408: debug displayed
# visibleKey = "displayed"
visibleValue = BoolTrueValue if self.visible else BoolFalseValue
visibleConditionStr = "%s == %s" % (visibleKey, value)
qs.append(visibleConditionStr)
if DEBUG:
    print("visibleConditionStr=%s" % visibleConditionStr)
    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" % (eachRectKey, eachRectValue)
            qs.append(curMatchRule)

predicate = ' AND '.join(qs)
chain = '/'.join([self.class_name or 'XCUIElementType',
                  predicate])

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.predicate)
    if self.xpath:
        return self._wdasearch('xpath', self.xpath)
    if self.class_chain:
        return self._wdasearch('class chain', self.class_chain)

    chain = '**' + ''.join(
        self.parent_class_chains) + self._gen_class_chain()
    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', self.predicate)

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

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

    chain = '**' + ''.join(
        self.parent_class_chains) + self._gen_class_chain()
    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, respInfo))

    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(''), element)
element = Element(self.http.new_client(''), element)

    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' % element_id)
        ele = Element(self.http.new_client(''), element_id)
        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" % elementIdList)

    return elementIdNum

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

    Returns:
        True, Element
        False, error info

    Raises:
    """

```

```

isFound, respInfo = False, "Unknown error"

if DEBUG:
    # print("get: timeout=%s, raise_error=%s, isRes=%s"
    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(
            time.time()))
        print("curDatetime=%s -> Not timeout, continue" % curDatetime)

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

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

    # 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 found")
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, respInfo))

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_chains
    return Selector(self.http, self.session, *args, **kwargs)

@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=raise_error)
    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

```

```

# dragfromtoreduration
# twoFingerTap

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

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_ID
        """
        self.http = httpclient
        self._id = id
        # add cached rect=bounds, to avoid later get rect
        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._id + url, data)

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

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

    def _wda_prop(self, key):
        ret = self._request('GET', 'wda/element/%s/%s' % (self._id, key))
        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: FBElementClassName
    # so, for:
    # <XCUIElementTypeButton type="XCUIElementTypeButton" name="Name"/>
    # name is XCUIElementTypeButton in type="XCUIElementTypeButton"
    # 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 download
#   <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 w/
    Raises:
        ValueError
    distance=1.0 means, element (width or height) mult:
    """
    if direction == 'visible':
        self._wda_req('post', '/scroll', {'toVisible': 1})
    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 zero
        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, **kwargs)

    # 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 list")
    # elif isinstance(locator, dict):
    # {'name': '公众号'}
    # query = {}
    # query = locator
    query = copy.deepcopy(locator)

    # Note: not auto convert key for compatible old Android
    # {'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", locator)

    # 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 (somehow)
    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=query)  
    return isFound, foundElement
```

findElement 查找元素

```

# def findElement(self, query={}, timeout=WaitFind):
# def findElement(self, query={}, timeout=WaitFind, isRetryNoYWhenFail=False):
def findElement(self, query={}, timeout=WaitFind, isRetryNoYWhenFail=False):
    """Find element from query

    Args:
        query (dict): query condition
        timeout(float): max wait time
        isRetryNoYWhenFail(bool): True to do retry but remove y position
    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=timeout)
    logging.debug("elementSelector=%s", elementSelector)
    isFound, respInfo = elementSelector.get()
    logging.debug("query=%s -> isFound=%s, respInfo=%s", query, isFound, respInfo)
    if not isFound:
        if isRetryNoYWhenFail and ("y" in query):
            # Note:
            # for sometime / many cases:
            #     not found element due to y position is normal
            #     for these cases, retry to find without y
            noYQuery = copy.deepcopy(query)
            del noYQuery["y"]
            elementSelectorNoY = self.curSession(**noYQuery)
            logging.debug("elementSelectorNoY=%s", elementSelectorNoY)
            isFound, respInfo = elementSelectorNoY.get()
            logging.debug("retry no y: noYQuery=%s -> isFound=%s", noYQuery, isFound)
            # sometime here found out element but y is negative
            # <XCUIElementTypeImage type="XCUIElementTypeImage">
            # actually is not visible -> not our expected result
        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", isReallyFound)
        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="XCUIElementTypeNavigation">  
    <XCUIElementTypeButton type="XCUIElementTypeButton" name="Back" />  
    <XCUIElementTypeOther type="XCUIElementTypeOther" name="Placeholder" />  
</XCUIElementTypeNavigationBar>
```

写法是：

```
wifiName = "无线局域网"
parentNavBarClassChain = "/XCUIElementTypeNavigationBar[1]"
wifiQuery = {"type": "XCUIElementTypeOther", "name": wifiName}
wifiQuery["parent_class_chains"] = [parentNavBarClassChain]
isFoundWifi, respInfo = self.findElement(query=wifiQuery, t=
```

和：

代码 判断iOS中页面是否处于 微信:

```
weixinTabQuery = {"type": "XCUIElementTypeButton", "name": "WeChat"}  
weixinTabQuery["parent_class_chains"] = [ tabBarClassChain  
isFoundWeixin, respInfo = self.findElement(query=weixinTabQuery)  
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": "XCUIElementTypeStaticText", "name": "下拉返回商品详情", "isFound": true}
isFound, foundElement = self.findElement(dropDownReturnDetailQuery)
```

- AppStore 详情页 不折叠输入法 带金额的购买按钮 ¥1.00:

```
<XCUIElementTypeCollectionView type="XCUIElementTypeCollection"
    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
        <XCUIElementTypeCell type="XCUIElementTypeCell" enabled="true">
            <XCUIElementTypeImage type="XCUIElementTypeImage" value="1.00" />
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="¥1.00" />
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="购买" />
            <XCUIElementTypeButton type="XCUIElementTypeButton" value="购买" />
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="¥1.00" />
            <XCUIElementTypeButton type="XCUIElementTypeButton" value="¥1.00" />
            <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                </XCUIElementTypeCell>
```

查找写法是:

```
parentCellClassChain = "/XCUIElementTypeCell[`rect.x = 0 And `rect.y = 0 And `rect.width = 100 And `rect.height = 100] And XCUIElementTypeImage[`rect.x = 0 And `rect.y = 0 And `rect.width = 100 And `rect.height = 100] And XCUIElementTypeStaticText[`rect.x = 0 And `rect.y = 0 And `rect.width = 100 And `rect.height = 100] And XCUIElementTypeStaticText[`rect.x = 0 And `rect.y = 0 And `rect.width = 100 And `rect.height = 100] And XCUIElementTypeButton[`rect.x = 0 And `rect.y = 0 And `rect.width = 100 And `rect.height = 100] And XCUIElementTypeStaticText[`rect.x = 0 And `rect.y = 0 And `rect.width = 100 And `rect.height = 100]"'
moneyButtonQuery = {"type": "XCUIElementTypeButton", "name": "购买", "isFound": true}
moneyButtonQuery["parent_class_chains"] = [parentCellClassChain]
foundMoneyButton, respInfo = self.findElement(query=moneyButtonQuery)
```

(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="E">
        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText" value="J">
    </XCUIElementTypeOther>
</XCUIElementTypeOther>

```

代码:

```

floatUpperLetterElement = None
# floatUpperLetterQuery = {"type": "XCUIElementTypeStaticText", "value": "J"}
# isFound, foundElement = findElement(curSession, floatUpperLetterQuery)
floatUpperLetterQuery = {"type": "XCUIElementTypeOther", "name": "D"}
isFound, foundElement = self.findElement(floatUpperLetterQuery)

```

(5) 复杂的, value包含的

```

curNoticeQuery = {"type": "XCUIElementTypeStaticText", "value": "立即体验"}
curNoticeQuery["parent_class_chains"] = [scrollViewClassChain]
isFoundCurNotice, respInfo = self.findElement(query=curNoticeQuery)

```

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 element
    enableDebug (bool): if enable debug, then draw click rectangle
    isShowErrLog(bool): True to show error log, False to hide
Returns:
    bool
Raises:
    .....
foundAnClicked = False

# isFound, respInfo = self.findElement(query=query, timeout=timeout)
# isFound, respInfo = self.findElement(query=query, timeout=timeout)
isFound, respInfo = self.findElement(query=query, timeout=timeout)
logging.debug("isFound=%s, respInfo=%s", isFound, respInfo)
if isFound:
    curElement = respInfo

    # if enableDebug:
    #     # for debug
    #     curScreenFile = debugSaveScreenshot(curScale=1.0)
    #     utils.imageDrawRectangle(curScreenFile, rectList)

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

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

foundAndClickedDoSearch = self.findAndClickElement(searchButtonQuery)
```

(2) 场景：设置 WiFi 配置代理 从手动切换到 关闭 存储

xml:

```
<XCUIElementTypeNavigationBar type="XCUIElementTypeNavigationBar">
<XCUIElementTypeButton type="XCUIElementTypeButton" name="存储">
<XCUIElementTypeOther type="XCUIElementTypeOther" name="关闭">
<XCUIElementTypeButton type="XCUIElementTypeButton" name="关闭">
</XCUIElementTypeNavigationBar>
```

代码:

```
storeName = "存储"
parentNavBarClassChain = "//XCUIElementTypeNavigationBar[1]"
storeButtonQuery = {"type": "XCUIElementTypeButton", "name": storeName}
storeButtonQuery["parent_class_chains"] = [parentNavBarClassChain]
foundAndClickedStore = self.findAndClickElement(query=storeButtonQuery)
```

(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="true">
    <XCUIElementTypeTable type="XCUIElementTypeTable" enabled="true">
        <XCUIElementTypeCell type="XCUIElementTypeCell" enabled="true">
            <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                <XCUIElementTypeImage type="XCUIElementTypeImage" enabled="true">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText" enabled="true">
                        </XCUIElementTypeText>
                    </XCUIElementTypeImage>
                </XCUIElementTypeOther>
            </XCUIElementTypeCell>
        </XCUIElementTypeTable>
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
                    <XCUIElementTypeButton type="XCUIElementTypeButton" enabled="true">
                        </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeTable>
</XCUIElementTypeOther>
```

代码：

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

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 element
    # if not curButtonName:
    #     curSoupAttrs = curButtonSoup.attrs
    #     curButtonName = curSoupAttrs["name"]
    # # rights close white
    # # login close
    # curButtonQuery = {"type": "XCUIElementTypeButton",
    # # foundAndClicked = self.findAndClickElement(curButtonQuery)

    curButtonQuery = {"type": "XCUIElementTypeButton", "name": "OK"}
    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',
        else:
            # merge query
            # curButtonQuery = {**curButtonQuery, **extraQuery}
            curButtonQuery.update(extraQuery) # {'enabled': 'true', 'height': '32'}

    foundAndClicked = self.findAndClickElement(curButtonQuery)
    return foundAndClicked

```

不同的地方的各种调用：

```
foundAndProcessedPopup = self.findAndClickButtonElementByS
```

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 click it

    Args:
        bsChainList (list): dict list for dict of tag and attr
        soup (Soup): BeautifulSoup soup
        isUseWdaQueryAndClick (bool): for special node bs click
    Returns:
        bool: found and clicked 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.findAndClickButtonElementByS
        else:
            self.clickElementCenterPosition(foundSoup)
            foundAndClicked = True
    return foundAndClicked
```

场景：米家 弹框 立即体验 xml

```
<XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
    <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
        <XCUIElementTypeOther type="XCUIElementTypeOther" enabled="true">
            <XCUIElementTypeImage type="XCUIElementTypeImage" name="立即体验" value="立即体验" x="480" y="150" width="100" height="100" x2="580" y2="250" alt="立即体验" data-bbox="480 150 580 250"/>
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="立即体验" x="480" y="150" width="100" height="100" alt="立即体验" data-bbox="480 150 580 250"/>
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="立即体验" x="480" y="150" width="100" height="100" alt="立即体验" data-bbox="480 150 580 250"/>
            <XCUIElementTypeButton type="XCUIElementTypeButton" value="立即体验" x="480" y="150" width="100" height="100" alt="立即体验" data-bbox="480 150 580 250"/>
        </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": "立即体验", "value": "立即体验", "x": 480, "y": 150, "width": 100, "height": 100}
    }
]
foundAndProcessedPopup = self.findAndClickCenterPosition(immediatelyExperienceChainList)
return foundAndProcessedPopup
```

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

```

def findAndClickButtonElementBySoup(self, curButtonSoup=None):
    """
        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", "enab...
    # # foundAndClicked = self.findAndClickElement(curButtonName)

    curButtonQuery = {"type": "XCUIElementTypeButton", "enable...
    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': 'XCUIEl...

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

    foundAndClicked = self.findAndClickElement(curButtonQuery)
    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) # so change to wda query element then click
    # 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 attrDict)
            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", center)
    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
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 22:51:37

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) # curBattrystate=Charging
    curBattrystateName = curBattrystate.name
    logging.debug("curBattrystateName=%s", curBattrystateName)
    logging.info("Battery state: %s", curBattrystateName)
    return batteryLevelPercentInt
```

相关定义：

```
from wda import ScreenshotQuality, BatteryState, Application
```

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) [fc9e1f9f2e1339328b9580f826a30fded3bc69ff]
        # iPhone 11 Pro Max (13.3) + Apple Watch Series 5 - [509BC7C7-9C0E-42FA-8AB2-F5220E1A8D9F]
        foundDevId = re.search("\[(?P<iOSdevId>[\w\-.]+)\]^(?P<deviceStr>.+)$", eachDevStr)
        if foundDevId:
            iOSdevId = foundDevId.group("iOSdevId")
            iOSDevIdList.append(iOSdevId)
    # ['fc9e1f9f2e1339328b9580f826a30fded3bc69ff', 'D86F0B1A-8D9F-42FA-BE8A-9C0E509BC7C7']
    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) [fc9e1f9f2e1339328b9580f826
        Apple TV (13.3) [6680F059-4DE1-430C-B696-228AC
        Apple TV 4K (13.3) [048E58E8-6A27-4D81-BDEB-88
        Apple TV 4K (at 1080p) (13.3) [384D5E60-B6B1-4
        Apple Watch Series 4 - 40mm (6.1.1) [1B98415B-1
        Apple Watch Series 4 - 44mm (6.1.1) [661838E9-1
        iPad (7th generation) (13.3) [7F8EDE89-74E0-4B
        iPad Air (3rd generation) (13.3) [BBC48526-392
        iPad Pro (11-inch) (13.3) [04DD3B8A-5B78-48E8-8
        iPad Pro (12.9-inch) (3rd generation) (13.3) [I
        iPad Pro (9.7-inch) (13.3) [B11D5D40-FEA2-4114
        iPhone 11 (13.3) [509BC7C7-9C0E-42FA-8AB2-F522
        iPhone 11 Pro (13.3) [3E8E7E92-66F2-4AF3-A405-7
        iPhone 11 Pro (13.3) + Apple Watch Series 5 - 4
        iPhone 11 Pro Max (13.3) [50C15135-1532-44C5-B6
        iPhone 11 Pro Max (13.3) + Apple Watch Series 5
        iPhone 8 (13.3) [54589698-0C9F-407D-B21A-83432
        iPhone 8 Plus (13.3) [509B7103-97DB-4AB9-B829-6
        """
        # CoreData: annotation: Failed to load optimized r
        # InvalidCoreData = "CoreData"
        # Known Devices:
        # InvalidKnownDevices = "Known Devices"
        for eachDeviceStr in deviceRawList:
            eachDeviceStr = eachDeviceStr.strip()
            # isNotCoreData = InvalidCoreData not in eachDe
            # isNotKnownDevices = InvalidKnownDevices not :
            # if isNotCoreData and isNotKnownDevices:
            # if isNotKnownDevices:
            # Note: current only support iPhone devices, be
            #       with device name contain 'iphone'
            foundiPhone = re.search("iPhone", eachDeviceStr)
            if foundiPhone:
                deviceStrList.append(eachDeviceStr)
            """
            Mo iPhone (11.4.1) [fc9e1f9f2e1339328b9580f826
            iPhone 11 (13.3) [509BC7C7-9C0E-42FA-8AB2-F522
            iPhone 11 Pro (13.3) [3E8E7E92-66F2-4AF3-A405-7
            iPhone 11 Pro (13.3) + Apple Watch Series 5 - 4
            iPhone 11 Pro Max (13.3) [50C15135-1532-44C5-B6
            iPhone 11 Pro Max (13.3) + Apple Watch Series 5
            iPhone 8 (13.3) [54589698-0C9F-407D-B21A-83432
            iPhone 8 Plus (13.3) [509B7103-97DB-4AB9-B829-6
            """

```

```
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+\((?P<iOSVersion>[\w .]+)\)"  
    for eachDeviceStr in deviceStrList:  
        foundCurDev = re.search(curDevP, eachDeviceStr)  
        if foundCurDev:  
            deviceName = foundCurDev.group("deviceName")  
            iOSVersion = foundCurDev.group("iOSVersion")  
            break  
  
    return deviceName
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 11:09:06

app

此处整理和app相关的常用代码段。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-21 12:12:44

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-A5E3-244A74CDF5C1"
    }
    """

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

```

调用:

```

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

```


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="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                </XCUIElementTypeScrollView>
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                <XCUIElementTypeButton type="XCUIElementTypeButton">
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                </XCUIElementTypeButton>
            </XCUIElementTypeButton>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

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

        途虎养车 左滑引导页 3页/3页 立即进入：
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                </XCUIElementTypeScrollView>
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                <XCUIElementTypeButton type="XCUIElementTypeButton">
                </XCUIElementTypeButton>
            </XCUIElementTypeButton>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    """

    # GuideText = "guide"
    # parentScrollViewClassChain = "/XCUIElementTypeScrollView"
    # guideImgeQuery = {"type": "XCUIElementTypeImage", "name": "guideText"}
    # guideImgeQuery["parent_class_chains"] = [parentScrollViewClassChain]
    # isFound, guideImgeElement = findElement(curSession, guideImgeQuery)
    # if isFound:
    #     # foundAndClicked = clickElement(curSession, guideImgeElement)
    #     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.FullScreenAttrDict}
    },
]
]

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

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

    parentScrollViewSoup = guideImageSoup.parent
    if not parentScrollViewSoup:
        break

    validButtonSoupList = []

    nextSiblingList = parentScrollViewSoup.next_siblings
    for eachNextSiblingSoup in nextSiblingList:
        if hasattr(eachNextSiblingSoup, "attrs"):
            soupAttrDict = eachNextSiblingSoup.attrs #
            soupType = soupAttrDict.get("type") # 'XCUIElementTypeImage'
            soupName = soupAttrDict.get("name") # 'loading'
            soupVisible = soupAttrDict.get("visible") #
            isButton = soupType == "XCUIElementTypeButton"
            # isLoadingName = bool(re.search(soupName,
            # isLoadingName = bool(re.search("loading", soupName))
            isVisible = soupVisible == "true"
            isValid = isButton and isLoadingName and isVisible
            if isValid:
                validButtonSoupList.append(eachNextSiblingSoup)

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

```

```
# end page, click button, into main page
lastPageButtonSoup = validButtonSoupList[0]
self.clickElementCenterPosition(lastPageButton)
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:
 - 
 - 最后一页:
 - 
- 途虎养车
 - 第一页
 - 
 - 第二页
 - 
 - 第三页
 - 

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 22:52:24

微信相关

判断是否处于微信

```
def iOSisInWeixin(self):
    tabBarClassChain = "/XCUIElementTypeTabBar[`rect.width
    ....
        <XCUIElementTypeTabBar type="XCUIElementTypeTabBar"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeOther type="XCUIElementTypeOther"
                    <XCUIElementTypeOther type="XCUIElementTypeOther"
                        <XCUIElementTypeOther type="XCUIElementTypeOther"
                            <XCUIElementTypeOther type="XCUIElementTypeOther"
                                <XCUIElementTypeOther type="XCUIElementTypeOther"
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeButton type="XCUIElementTypeButton"
                            <XCUIElementTypeButton type="XCUIElementTypeButton"
                            <XCUIElementTypeButton type="XCUIElementTypeButton"
                                <XCUIElementTypeButton type="XCUIElementTypeButton"
                            </XCUIElementTypeButton>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeTabBar>
    ....
    weixinTabQuery = {"type":"XCUIElementTypeButton", "name": "发现", "label": "发现"}
    weixinTabQuery["parent_class_chains"] = [ tabBarClassChain ]
    isFoundWeixin, respInfo = self.findElement(query=weixinTabQuery)
    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="XCUIElementTypeNavigationBar">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
        </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

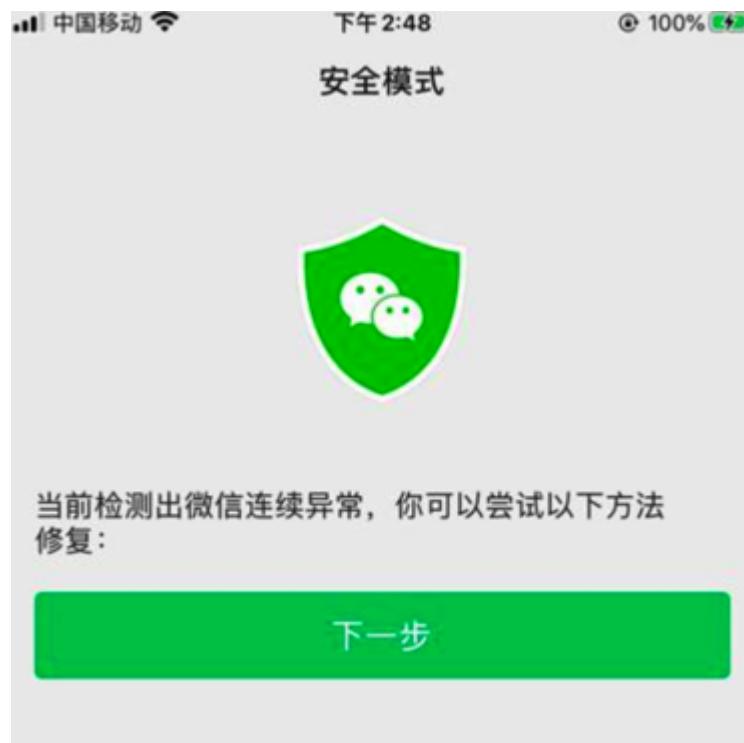
```

安全模式

背景

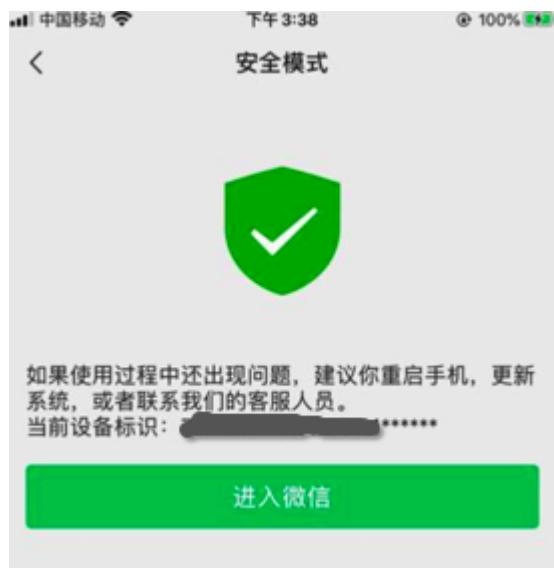
在 facebook-wda 调试期间，通过代码控制微信app的退出和启动，多次如此调试后就会导致微信检测到以为发生了多次的异常，所以会进入安全模式。

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：









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 iOSMakessureIntoWeixin(self):
    """Makessure 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.iOSWeixinException()
            if foundAndProcessedException:
                beInWeixin = self.iOSIsInWeixin()

        maxRetryNum -= 1

    return beInWeixin

def iOSWeixinExceptionNextStep(self):
    foundAndClicked = False

    scrollViewClassChain = "/XCUIElementTypeScrollView[`reco...`]"
    scrollViewElement = self.findElement(scrollViewClassChain)

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

    # nextStepButtonQuery = {"type":"XCUIElementTypeButton"}
    # nextStepButtonQuery["parent_class_chains"] = [scrollViewElement]

    # intoWeixinButtonQuery = {"type":"XCUIElementTypeButton"}
    # intoWeixinButtonQuery["parent_class_chains"] = [scrollViewElement]

    .....
    <XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
        <XCUIElementTypeImage type="XCUIElementTypeImage">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                <XCUIElementTypeButton type="XCUIElementTypeButton">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        </XCUIElementTypeButton>
                    </XCUIElementTypeStaticText>
                </XCUIElementTypeButton>
            </XCUIElementTypeStaticText>
        </XCUIElementTypeImage>
    </XCUIElementTypeScrollView>
    .....
    # continuousExceptionQuery = {"type":"XCUIElementTypeStaticText"}
    # continuousExceptionQuery["parent_class_chains"] = [scrollViewElement]
    # isFoundContinuousException, respInfo = self.findElement(continuousExceptionQuery)
    # if isFoundContinuousException:
    #     isFoundNextStep, respInfo = self.findElement(nextStepButtonQuery)
    #     if isFoundNextStep:
    #         nextStepElement = respInfo

```

```

#         foundAndClicked = self.clickElement(nextStepElement)

#####
<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
<XCUIElementTypeImage type="XCUIElementTypeImage">
<XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
<XCUIElementTypeButton type="XCUIElementTypeButton">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
    </XCUIElementTypeButton>
<XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
<XCUIElementTypeButton type="XCUIElementTypeButton">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
    </XCUIElementTypeButton>
<XCUIElementTypeOther type="XCUIElementTypeOther">
<XCUIElementTypeOther type="XCUIElementTypeOther">
</XCUIElementTypeScrollView>
#####

# # if not foundAndClicked:
# rebootWeixinQuery = {"type":"XCUIElementTypeStaticText"}
# rebootWeixinQuery["parent_class_chains"] = [scrollView]
# isFoundRebootWeixin, respInfo = self.findElement(query)
# if isFoundRebootWeixin:
#     isFoundNextStep, respInfo = self.findElement(query)
#     if isFoundNextStep:
#         nextStepElement = respInfo
#         foundAndClicked = self.clickElement(nextStepElement)

#####
<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
<XCUIElementTypeOther type="XCUIElementTypeOther">
    <XCUIElementTypeImage type="XCUIElementTypeImage">
        <XCUIElementTypeImage type="XCUIElementTypeImage">
</XCUIElementTypeOther>
<XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
<XCUIElementTypeButton type="XCUIElementTypeButton">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
    </XCUIElementTypeButton>
<XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
<XCUIElementTypeButton type="XCUIElementTypeButton">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
    </XCUIElementTypeButton>
<XCUIElementTypeOther type="XCUIElementTypeOther">
<XCUIElementTypeOther type="XCUIElementTypeOther">
</XCUIElementTypeScrollView>
#####

# # if not foundAndClicked:
# clearCacheQuery = {"type":"XCUIElementTypeStaticText"}
# clearCacheQuery["parent_class_chains"] = [scrollView]
# isFoundClearCache, respInfo = self.findElement(query)
# if isFoundClearCache:
#     isFoundNextStep, respInfo = self.findElement(query)

```

```

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

.....
<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
    <XCUIElementTypeImage type="XCUIElementTypeImage">
        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                    </XCUIElementTypeButton>
                <XCUIElementTypeButton type="XCUIElementTypeButton">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        </XCUIElementTypeButton>
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                </XCUIElementTypeButton>
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                    </XCUIElementTypeScrollView>
.....
.....
<XCUIElementTypeScrollView type="XCUIElementTypeScrollView">
    <XCUIElementTypeImage type="XCUIElementTypeImage">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                <XCUIElementTypeButton type="XCUIElementTypeButton">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        </XCUIElementTypeButton>
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            </XCUIElementTypeScrollView>
.....
.....
EachStepNoticeList = [
    # ("当前检测出微信连续异常，你可以尝试以下方法修复：", ButtonLabelINextStep),
    # ("当前检测出微信连续异常", ButtonLabelINextStep),
    # ("建议你重启手机，避免微信启动异常", ButtonLabelINextStep),
    # ("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": "XCUIElementTypeStaticText"}
    # curNoticeQuery = {"type": "XCUIElementTypeStaticText", "label": "建议你重启手机，避免微信启动异常"}
    curNoticeQuery = {"type": "XCUIElementTypeStaticText", "label": "当前检测出微信连续异常"}
    curNoticeQuery["parent_class_chains"] = [scrollView]
    isFoundCurNotice, respInfo = self.findElement(query=curNoticeQuery)
    if isFoundCurNotice:

```

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

return foundAndClicked
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2020-06-25 15:04:46

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="XCUIElementTypeImage">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeImage">
                        <XCUIElementTypeImage type="XCUIElementTypeImage">
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeSearchField type="XCUIElementTypeSearchField">
                                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                            </XCUIElementTypeButton>
                                        </XCUIElementTypeStaticText>
                                    </XCUIElementTypeButton>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeSearchField>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeOther>
                </XCUIElementTypeImage>
            </XCUIElementTypeButton>
        </XCUIElementTypeImage>
    </XCUIElementTypeImage>
    <XCUIElementTypeImage type="XCUIElementTypeImage">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeImage">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeSearchField type="XCUIElementTypeSearchField">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                        </XCUIElementTypeButton>
                                    </XCUIElementTypeStaticText>
                                </XCUIElementTypeButton>
                            </XCUIElementTypeSearchField>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeOther>
                </XCUIElementTypeImage>
            </XCUIElementTypeButton>
        </XCUIElementTypeImage>
    </XCUIElementTypeImage>
    <XCUIElementTypeImage type="XCUIElementTypeImage">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeImage">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeSearchField type="XCUIElementTypeSearchField">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                        </XCUIElementTypeButton>
                                    </XCUIElementTypeStaticText>
                                </XCUIElementTypeButton>
                            </XCUIElementTypeSearchField>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeOther>
                </XCUIElementTypeImage>
            </XCUIElementTypeButton>
        </XCUIElementTypeImage>
    </XCUIElementTypeImage>
```

```
        </XCUIElementTypeButton>
        </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    </XCUIElementTypeImage>
    .....
    soup = CommonUtils.xmlToSoup(page)
    widthStr = str(self.X)
    foundImage = soup.find(
        'XCUIElementTypeImage',
        attrs={"type": "XCUIElementTypeImage", "enabled": "true"})
    if foundImage:
        foundSearchField = foundImage.find("XCUIElementTypeSearchField",
            attrs={"type": "XCUIElementTypeSearchField", "isPlaceholder": "false"})
        if foundSearchField:
            isPublicAccountSearch = True
            # curInputValue = foundSearchField.attrs["value"]
            curInputValue = foundSearchField.attrs.get("value")
    return isPublicAccountSearch, curInputValue
```

isPublicAccountFocusOrIntoPage_iOS


```

<XCUIElementTypeButton type="XCUIElementTypeButton">
    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
        </XCUIElementTypeButton>
    <XCUIElementTypeButton type="XCUIElementTypeButton">
        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
            </XCUIElementTypeButton>
    </XCUIElementTypeButton>

    .....

soup = CommonUtils.xmlToSoup(page)
foundNavBar = soup.find(
    'XCUIElementTypeNavigationBar',
    attrs={"type": "XCUIElementTypeNavigationBar", "enabled": "true"})
# logging.debug("foundNavBar=%s", foundNavBar)
if foundNavBar:
    foundTypeOther = foundNavBar.find("XCUIElementTypeOther",
                                       attrs={"type": "XCUIElementTypeOther", "enabled": "true"})
    # 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 isTypeOtherNameNotEmpty
        logging.debug("isTypeOtherNameEmpty=%s", isTypeOtherNameEmpty)
        if isTypeOtherNameEmpty:
            foundIntoAccount = soup.find(
                'XCUIElementTypeButton',
                attrs={"type": "XCUIElementTypeButton", "enabled": "true"})
            # logging.debug("foundIntoAccount=%s", foundIntoAccount)
            if foundIntoAccount:
                prevSiblingList = foundIntoAccount.prev_siblings
                logging.debug("prevSiblingList=%s", prevSiblingList)
                curAccountZhcnName = self.account_zw
                TypeStaticText = "XCUIElementTypeStaticText"
                for eachPrevSibling in prevSiblingList:
                    # curType = eachPrevSibling.attrs['type']
                    # curName = eachPrevSibling.attrs['name']
                    # curType = eachPrevSibling.attrs.get('type')
                    # curName = eachPrevSibling.attrs.get('name')
                    if hasattr(eachPrevSibling, 'attrs'):
                        curType = eachPrevSibling.attrs['type']
                        curName = eachPrevSibling.attrs['name']
                        if (curType == TypeStaticText):
                            isPublicAccountFocusOrIntoAccount = True
                        else:
                            logging.debug("eachPrevSibling=%s", eachPrevSibling)
                            if (curName == curAccountZhcnName):
                                isPublicAccountFocusOrIntoAccount = True
                if isPublicAccountFocusOrIntoAccount:
                    logging.debug("isPublicAccountFocusOrIntoAccount=%s", isPublicAccountFocusOrIntoAccount)
                    return True
            else:
                logging.debug("foundIntoAccount=%s", foundIntoAccount)
        else:
            logging.debug("foundTypeOther=%s", foundTypeOther)
    else:
        logging.debug("foundNavBar=%s", foundNavBar)
else:
    logging.debug("foundNavBar=%s", foundNavBar)

```

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

调用：

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

    OfflinePageNavBarNameList = [
        "微信",
        "通讯录",
        "公众号",
    ]
    hasNavBar, naviBarName = self.isPageHasNavBar_iOS(page)
    if hasNavBar:
        if naviBarName in OfflinePageNavBarNameList:
            # is in some weixin page
            isOffline = True

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

    if not isOffline:
        isPublicAccountFocusOrIntoPage = self.isPublicAccountFocusOrIntoPage_iOS()
        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：





从中查找出 公众号的中文名（的全称）

```

# def findWeixinPublicAccountZhcnSoup(self, soup, curAccou
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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            <XCUIElementTypeOther type="XCUIElementTypeText">
                <XCUIElementTypeOther type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                        </XCUIElementTypeOther>
                    </XCUIElementTypeText>
                </XCUIElementTypeText>
            </XCUIElementTypeText>
            <XCUIElementTypeText>
                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    </XCUIElementTypeText>
                </XCUIElementTypeText>
            </XCUIElementTypeText>
        </XCUIElementTypeText>
    </XCUIElementTypeText>
    <XCUIElementTypeText>
        <XCUIElementTypeText>
            <XCUIElementTypeText>
                <XCUIElementTypeText>
                    <XCUIElementTypeText>
                        <XCUIElementTypeText>
                            <XCUIElementTypeText>
                                <XCUIElementTypeText>
                                    <XCUIElementTypeText>
                                        <XCUIElementTypeText>
                                            <XCUIElementTypeText>
                                                <XCUIElementTypeText>
                                                    <XCUIElementTypeText>
                                                        <XCUIElementTypeText>
                                                            <XCUIElementTypeText>
                                                                <XCUIElementTypeText>
                                                                    <XCUIElementTypeText>
                                                                        <XCUIElementTypeText>
                                                                            <XCUIElementTypeText>
                                                                                <XCUIElementTypeText>
                                                                                    <XCUIElementTypeText>
                                                                                        <XCUIElementTypeText>
                                                                                            <XCUIElementTypeText>
                                                                                                <XCUIElementTypeText>
                                                                                                 <XCUIElementTypeText>
                                                                                                     <XCUIElementTypeText>
                                         </XCUIElementTypeText>
                                     </XCUIElementTypeText>
                                 </XCUIElementTypeText>
                             </XCUIElementTypeText>
                         </XCUIElementTypeText>
                     </XCUIElementTypeText>
                 </XCUIElementTypeText>
             </XCUIElementTypeText>
         </XCUIElementTypeText>
     </XCUIElementTypeText>

```

搜索结果中文名节点是Other, 其下是多个Text节点:

```

<XCUIElementTypeOther type="XCUIElementTypeText">
    <XCUIElementTypeText>
        <XCUIElementTypeText>
            <XCUIElementTypeText>
                <XCUIElementTypeText>
                    <XCUIElementTypeText>
                        <XCUIElementTypeText>
                            <XCUIElementTypeText>
                                <XCUIElementTypeText>
                                    <XCUIElementTypeText>
                                        <XCUIElementTypeText>
                                            <XCUIElementTypeText>
                                                <XCUIElementTypeText>
                                                    <XCUIElementTypeText>
                                                        <XCUIElementTypeText>
                                                            <XCUIElementTypeText>
                                                                <XCUIElementTypeText>
                                                                    <XCUIElementTypeText>
                                                                        <XCUIElementTypeText>
                                </XCUIElementTypeText>
                            </XCUIElementTypeText>
                        </XCUIElementTypeText>
                    </XCUIElementTypeText>
                </XCUIElementTypeText>
            </XCUIElementTypeText>
        </XCUIElementTypeText>
    </XCUIElementTypeText>

```



```

# idParentPrevSiblingList = idParent.previous_
# accountDescNode = None
# accountZhcnNode = None

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

# for eachPrevSiblingNode in idParentPrevSiblingList:
#     curNodeName = eachPrevSiblingNode.name
#     isTypeOtherNode = curNodeName == TypeOther
#     if isTypeOtherNode:
#         typeOtherNodeCurIdx += 1

#         if AccountDescNodeIdx == typeOtherNodeCurIdx:
#             accountDescNode = eachPrevSiblingNode
#             elif AccountZhcnNodeIdx == typeOtherNodeCurIdx:
#                 accountZhcnNode = eachPrevSiblingNode

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

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

# if accountZhcnNode:
#     accountZhcnTextSoup = accountZhcnNode.find_element_by_type(
#         'XCUIElementTypeStaticText',
#         attrs={"type": "XCUIElementTypeStaticText"})
#     )

# method 3: parent.parent is 搜一搜, direct child
idParentParent = idParent.parent
if idParentParent:
    otherSoupList = idParentParent.find_all(
        "XCUIElementTypeOther",
        attrs={"type": "XCUIElementTypeOther"},
        recursive=False,
    )
    if otherSoupList and (len(otherSoupList) >= 2):
        firstOtherSoup = otherSoupList[0]
        if firstOtherSoup.attrs["name"] == "公
            secondOtherSoup = otherSoupList[1]
            zhcnNameSoupList = secondOtherSoup.find_element_by_type(
                "XCUIElementTypeStaticText",
                attrs={"type": "XCUIElementTypeStaticText"})
            )
        if zhcnNameSoupList:

```

```
        for eachTextSoup in zhcnNameSoup:
            curPartName = eachTextSoup
            accountZhcnFullName += curPartName

        if accountZhcnFullName:
            secondOtherAttrDict = secondOtherAttrDict
            parentX = secondOtherAttrDict['x']
            parentY = secondOtherAttrDict['y']
            parentWidth = secondOtherAttrDict['width']
            parentHeight = secondOtherAttrDict['height']
            parentNodeLocator = {
                "type": "XCUIElementTypeText",
                "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_cfcfcfcee032cc"
curPageXml = self.get_page_source()
soup = CommonUtils.xmlToSoup(curPageXml)

accountZhcnName, parentNodeLocator = self.findWeixinPublicAccountName(
    curAccountId, curPageXml, soup)
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 22:53:50

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, isDebugEnabled=False):
    """iOS terminal and re-launch Settings=Preferences app

    Args:
        isTerminateFirst (bool): force terminal Settings before
        isDebugEnabled (bool): enable debug for app state
    Returns:
        bool
    Raises:
        ...

iOS_AppId_Settings = "com.apple.Preferences"

if isTerminateFirst:
    if isDebugEnabled:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_Settings)
        logging.info("before terminal: curState=%s", curState)
        # stop before start to avoid current page is not handled
        isTerminalOk, respInfo = self.iOSTerminateApp(iOS_AppId_Settings)
        logging.info("%s: isTerminalOk=%s, respInfo=%s", __name__, isTerminalOk, respInfo)
    if isDebugEnabled:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_Settings)
        logging.info("after terminal: curState=%s", curState)

    # settingsSession = self.wdaClient.session(iOS_AppId_Settings)
    # logging.debug("settingsSession=%s" % settingsSession)
    # launchResult = self.wdaClient.app_launch(iOS_AppId_Settings)
    # logging.debug("launchResult=%s", launchResult)
    isLaunchOk, respInfo = self.iOSLaunchApp(iOS_AppId_Settings)
    logging.info("isLaunchOk=%s, respInfo=%s", isLaunchOk, respInfo)
    # logging.info("launchResult: value=%s, status=%s, sessionId=%s", launchResult.value, launchResult.status, launchResult.sessionId)
    # logging.info("launchResult=%s", str(launchResult))
    # launchResult=GenericDict(value=None, sessionId='79A39B')
    if isDebugEnabled:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_Settings)
        logging.info("after launch: curState=%s", curState)
return isLaunchOk, respInfo

```

调用：

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

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 12:18:00

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(newPro
```

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(proxyConfigInfo)
    isRestoreOk, respInfo = self.iOSWifiDetailUpdateProxy(proxyConfigInfo)
    logStr = "%s to restore previous proxy %s" % (isRestoreOk, respInfo)
    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 current proxy
       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.iOSLaunchSettingsAndFindCurrentProxy()
    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.iOSProxyConfigUpdate()
        logging.info("Update proxy result: %s, %s", isUpdateProxyOk, respInfo)
        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, click
    Args:
    Returns:
        bool, soup/str
            True, proxy type soup
            False, error message str
    Raises:
    """
    isGetTypeOk = False
    respInfo = None

    isIntoWiFiListOk, respInfo = self.iOSLaunchSettingsAndGetProxyType()
    if not isIntoWiFiListOk:
        errMsg = respInfo
        respInfo = "Fail into WiFi list page for %s" % errMsg
        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 already being in WiFi list page, so
    isInWifiList = self.iOSIsInWiFiList()
    if isInWifiList:
        isIntoWiFiListOk = True
        respInfo = None
        return isIntoWiFiListOk, respInfo
    else:
        # foundAndClickedWifi = self.iOSFromSettingsIntoWiFiList()
        foundAndClickedWifi = CommonUtils.multipleRetry(lambda: self.iOSFromSettingsIntoWiFiList())
        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, isDebugEnabled=False):
    """iOS terminal and re-launch Settings app

    Args:
        isTerminateFirst (bool): force terminal Settings before
        isDebugEnabled (bool): enable debug for app state
    Returns:
        bool
    Raises:
        ...
    """

    iOS_AppId_Settings = "com.apple.Preferences"

    if isTerminateFirst:
        if isDebugEnabled:
            isGetOk, curState = self.iOSGetAppState(iOS_AppId_Settings)
            logging.info("before terminal: curState=%s", curState)
            # stop before start to avoid current page is not home
            isTerminalOk, respInfo = self.iOSTerminateApp(iOS_AppId_Settings)
            logging.info("%s: isTerminalOk=%s, respInfo=%s", __name__, isTerminalOk, respInfo)
        if isDebugEnabled:
            isGetOk, curState = self.iOSGetAppState(iOS_AppId_Settings)
            logging.info("after terminal: curState=%s", curState)

    # settingsSession = self.wdaClient.session(iOS_AppId_Settings)
    # logging.debug("settingsSession=%s" % settingsSession)
    # launchResult = self.wdaClient.app_launch(iOS_AppId_Settings)
    # logging.debug("launchResult=%s", launchResult)
    isLaunchOk, respInfo = self.iOSLaunchApp(iOS_AppId_Settings)
    logging.info("isLaunchOk=%s, respInfo=%s", isLaunchOk, respInfo)
    # logging.info("launchResult: value=%s, status=%s, sessionId=%s", str(launchResult))
    # launchResult: value=None, status=0, sessionId=79A39B7E
    # logging.info("launchResult=%s", str(launchResult))
    # launchResult=GenericDict(value=None, sessionId='79A39B7E')
    if isDebugEnabled:
        isGetOk, curState = self.iOSGetAppState(iOS_AppId_Settings)
        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：



```

def iOSFromSettingsIntoWifiList(self):
    """from settings page, click 无线局域网 into WiFi list page
    foundAndClickedWifi = False
    """
    设置 顶部 无线局域网:
    <XCUIElementTypeTable type="XCUIElementTypeTable">
        <XCUIElementTypeCell type="XCUIElementTypeCell">
            <XCUIElementTypeImage type="XCUIElementTypeImage">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
        </XCUIElementTypeCell>
    </XCUIElementTypeTable>
    """
    # parentWifiCellClassChain = "/XCUIElementTypeCell[`name` == '无线局域网']"
    parentWifiCellClassChain = "/XCUIElementTypeCell[`name` == '无线局域网' and `type` == 'XCUIElementTypeCell']"
    wifiTextQuery = {"type": "XCUIElementTypeStaticText", "text": "无线局域网"}
    wifiTextQuery["parent_class_chains"] = [parentWifiCellClassChain]
    foundAndClickedWifi = self.findAndClickElement(query=wifiTextQuery)
    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="XCUIElementTypeNavigationBar">
        <XCUIElementTypeButton type="XCUIElementTypeButton">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
    </XCUIElementTypeNavigationBar>
    """
    wifiName = "无线局域网"
    parentNavBarClassChain = "/XCUIElementTypeNavigationBar[`name` == '无线局域网']"
    wifiQuery = {"type": "XCUIElementTypeOther", "name": wifiName}
    wifiQuery["parent_class_chains"] = [parentNavBarClassChain]
    isFoundWifi, respInfo = self.findElement(query=wifiQuery)
    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 WiFi detail page"""
    isIntoDetailOk = False
    .....
    设置 无线局域网 列表页:
    <XCUIElementTypeTable type="XCUIElementTypeTable">
        <XCUIElementTypeCell type="XCUIElementTypeCell">
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="WIFI
            <XCUIElementTypeOther type="XCUIElementTypeImage" value="Wi-Fi
            <XCUIElementTypeOther type="XCUIElementTypeImage" value="Wi-Fi
            <XCUIElementTypeImage type="XCUIElementTypeImage" value="Wi-Fi
            </XCUIElementTypeOther>
            <XCUIElementTypeOther type="XCUIElementTypeImage" value="Wi-Fi
            <XCUIElementTypeImage type="XCUIElementTypeImage" value="Wi-Fi
            <XCUIElementTypeImage type="XCUIElementTypeImage" value="Wi-Fi
            <XCUIElementTypeButton type="XCUIElementTypeButton" value="更多
        </XCUIElementTypeCell>
    .....
    curPageXml = self.get_page_source()
    soup = CommonUtils.xmlToSoup(curPageXml)
    blueCheckChainList = [
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled": "true", "visible": "true", "name": "Wi-Fi
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "name": "Wi-Fi
        },
        {
            "tag": "XCUIElementTypeImage",
            "# attrs": {"enabled": "true", "visible": "true", "name": "Wi-Fi
            "attrs": {"enabled": "true", "name": "UIPreference
        },
    ],
    blueCheckSoup = CommonUtils.bsChainFind(soup, blueCheckChainList)
    if blueCheckSoup:
        parentOtherSoup = blueCheckSoup.parent
        parentCellSoup = parentOtherSoup.parent
        moreInfoSoup = parentCellSoup.find(
            'XCUIElementTypeButton',
            attrs={"type": "XCUIElementTypeButton", "name": "更多
        )
        if moreInfoSoup:
            clickedOk = self.clickElementCenterPosition(moreInfoSoup)
            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="XCUIElementTypeTable">
            <XCUIElementTypeCell type="XCUIElementTypeCell">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        </XCUIElementTypeStaticText>
                </XCUIElementTypeOther>
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    </XCUIElementTypeButton>
                            </XCUIElementTypeStaticText>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeCell>
                </XCUIElementTypeCell>
            </XCUIElementTypeTable>
        </XCUIElementTypeCell>
    </XCUIElementTypeTable>
    """
    curPageXml = self.get_page_source()
    soup = CommonUtils.xmlToSoup(curPageXml)
    configProxyChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled": "true", "visible": "true", "type": "XCUIElementTypeCell"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled": "true", "visible": "true", "type": "XCUIElementTypeStaticText"}
        }
    ]
    configProxySoup = CommonUtils.bsChainFind(soup, configProxyChainList)
    if configProxySoup:
        parentCellSoup = configProxySoup.parent
        # proxyTypeP = re.compile("(手动)|(自动)|(关闭)")
        # proxyTypeSoup = parentCellSoup.find(
        #     'XCUIElementTypeStaticText',
        #     attrs={"type": "XCUIElementTypeStaticText", "value": proxyTypeP})
    
```

```
# )
proxyTypeSoup = self.iOSFindChildProxyType(parentC
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, isReturnSoup):
    """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="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            <XCUIElementTypeButton type="XCUIElementTypeButton">
                                </XCUIElementTypeButton>
                            </XCUIElementTypeStaticText>
                        </XCUIElementTypeStaticText>
                    </XCUIElementTypeStaticText>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeCell>

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

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

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

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

```

```
<XCUIElementTypeOther type="XCUIElementTypeCell">
<XCUIElementTypeStaticText type="XCUIElementTypeText" value="代理类型" />
<XCUIElementTypeButton type="XCUIElementTypeText" value="手动" />
</XCUIElementTypeCell>
....
proxyTypeP = re.compile("(手动)|(自动)|(关闭)")
proxyTypeSoup = parentCellSoup.find(
    'XCUIElementTypeStaticText',
    attrs={"type": "XCUIElementTypeStaticText", "value": proxyTypeP})
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 attrDict)
            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", center)
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 examples:
        (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="XCUIElementTypeTable">
    <XCUIElementTypeCell type="XCUIElementTypeCell">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                        </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeStaticText>
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeCell type="XCUIElementTypeCell">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeCell type="XCUIElementTypeCell">
                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                            </XCUIElementTypeStaticText>
                                        </XCUIElementTypeOther>
                                    </XCUIElementTypeCell>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeCell>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeCell>
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
        </XCUIElementTypeOther>
    </XCUIElementTypeCell>
</XCUIElementTypeTable>
```

设置 无线局域网 配置代理 手动:

```
<XCUIElementTypeTable type="XCUIElementTypeTable">
    <XCUIElementTypeCell type="XCUIElementTypeCell">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                        </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeStaticText>
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeCell type="XCUIElementTypeCell">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeTextField type="XCUIElementTypeTextField">
                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                            <XCUIElementTypeCell type="XCUIElementTypeCell">
                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                    <XCUIElementTypeTextField type="XCUIElementTypeTextField">
                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                <XCUIElementTypeSwitch type="XCUIElementTypeSwitch">
                        </XCUIElementTypeSwitch>
                    </XCUIElementTypeOther>
                </XCUIElementTypeCell>
            </XCUIElementTypeOther>
        </XCUIElementTypeCell>
    </XCUIElementTypeTable>
```



```

    if not curTypeName:
        respInfo = "Fail to find current proxy type name in parentTableSoup"
        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(parentTableSoup)
        elif curTypeName == "自动":
            curProxyValue = self.getAutoProxyValue(parentTableSoup)

    if not curProxyValue:
        respInfo = "Fail to get %s proxy value" % curTypeName
        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 type")

```

```

# common logic process

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

    if isNeedUpdatenewValue:
        if newTypeName == "手动":
            isUpdateValueOk = self.setManualProxyValue(parentTableSoup)
        elif newTypeName == "自动":
            isUpdateValueOk = self.setAutoProxyValue(parentTableSoup)

        if isUpdateValueOk:
            isNeedStore = True
        else:
            respInfo = "Fail to update new %s proxy config" % curType
            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 to %s" % (oldProxyName, newProxyName)
            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


```

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

# <XCUIElementTypeTextField type="XCUIElementTypeTextField"
proxyPortSoup = parentTableSoup.find(
    'XCUIElementTypeTextField',
    attrs={"type": "XCUIElementTypeTextField", "name": "端口"})
if not proxyPortSoup:
    return manualProxyValue
proxyPort = proxyPortSoup.attrs.get("value", None) # '1080'

# <XCUIElementTypeSwitch type="XCUIElementTypeSwitch"
proxyAuthenticateSoup = parentTableSoup.find(
    'XCUIElementTypeSwitch',
    attrs={"type": "XCUIElementTypeSwitch", "name": "鉴权"})
if not proxyAuthenticateSoup:
    return manualProxyValue
proxyAuthenticate = proxyAuthenticateSoup.attrs.get("value", None)

authUser = None
authPassword = None

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

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

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:



```
def getAutoProxyValue(self, parentTableSoup):
    """in 配置代理 page, from parent table soup, find 自动代理
    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="XCUIElementTypeTable">
        <XCUIElementTypeCell type="XCUIElementTypeCell">
            <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                <XCUIElementTypeStaticText type="XCUIElementTypeTextfield">
                    <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                </XCUIElementTypeTextfield>
            </XCUIElementTypeTextfield>
            <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                <XCUIElementTypeStaticText type="XCUIElementTypeTextfield">
                    <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                </XCUIElementTypeTextfield>
            </XCUIElementTypeTextfield>
            <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                    <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                        <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                            <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                                <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                                    <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                                </XCUIElementTypeTextfield>
                            </XCUIElementTypeTextfield>
                        </XCUIElementTypeTextfield>
                    </XCUIElementTypeTextfield>
                </XCUIElementTypeTextfield>
            </XCUIElementTypeTextfield>
        </XCUIElementTypeCell>
    </XCUIElementTypeTable>
    """
    autoUrlSoup = parentTableSoup.find(
        'XCUIElementTypeTextField',
        attrs={"type": "XCUIElementTypeTextField", "name": "value"})
    if not autoUrlSoup:
        return autoProxyValue
    autoProxyValue = autoUrlSoup.attrs.get("value", None)
    return autoProxyValue
```

点击去切换类型：

```

def switchToProxyType(self, parentTableSoup, newProxyTypeName):
    """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="XCUIElementTypeTable">
        <XCUIElementTypeCell type="XCUIElementTypeCell">
            ...
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                ...
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                ...
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                ...
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                ...
        .....
        newProxySoup = parentTableSoup.find(
            'XCUIElementTypeStaticText',
            attrs={"type": "XCUIElementTypeStaticText", "value": newProxyName})
        if newProxySoup:
            clickedNewProxy = self.clickElementCenterPosition(newProxySoup)
            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, newManualProxyValue):
    """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:
        ...
    isUpdateManualOk = False
    ...
    <XCUIElementTypeCell type="XCUIElementTypeCell">
        <XCUIElementTypeOther type="XCUIElementTypeTextfield">
            <XCUIElementTypeTextField type="XCUIElementTypeTextField">
                <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                    <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                        <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                            </XCUIElementTypeTextfield>
                    </XCUIElementTypeTextfield>
                </XCUIElementTypeTextfield>
            </XCUIElementTypeTextfield>
        </XCUIElementTypeTextfield>
    </XCUIElementTypeCell>
    ...
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x = "
    newServerValue = newManualProxyValue["server"]
    serverFieldQuery = {"type": "XCUIElementTypeTextField",
                       "parent_class_chains": [parentCellClassChain]}
    # isFoundServer, respInfo = self.findElement(query=serverFieldQuery)
    # if not isFoundServer:
    #     return False
    isInputServerOk = self.wait_element_setText_iOS(serverFieldQuery)
    if not isInputServerOk:
        return False
    ...
    <XCUIElementTypeCell type="XCUIElementTypeCell">
        <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
            <XCUIElementTypeTextField type="XCUIElementTypeTextField">
                <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                    <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                        </XCUIElementTypeTextfield>
                </XCUIElementTypeTextfield>
            </XCUIElementTypeTextfield>
        </XCUIElementTypeTextfield>
    </XCUIElementTypeCell>
    ...
    newPortValue = newManualProxyValue["port"]
    portFieldQuery = {"type": "XCUIElementTypeTextField", "parent_class_chains": [parentCellClassChain]}
    isInputPortOk = self.wait_element_setText_iOS(portFieldQuery)
    if not isInputPortOk:
        return False
    ...
    <XCUIElementTypeCell type="XCUIElementTypeCell">
        <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
            <XCUIElementTypeTextField type="XCUIElementTypeTextField">
                <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                    </XCUIElementTypeTextfield>
            </XCUIElementTypeTextfield>
        </XCUIElementTypeTextfield>
    </XCUIElementTypeCell>

```

```

        <XCUIElementTypeSwitch type="XCUIElementTypeCell">
    </XCUIElementTypeCell>
    ....
    newAuthenticateValue = newManualProxyValue["authenticate"]
    authSwitchQuery = {"type": "XCUIElementTypeSwitch", "name": "authSwitch"}
    authSwitchQuery["parent_class_chains"] = [ parentCellChain ]
    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", "text": ""}
        },
        {
            "tag": "XCUIElementTypeSwitch",
            "attrs": {"enabled": "true", "visible": "true", "value": "0"}
        },
    ]
    authSwitchSoup = CommonUtils.bsChainFind(soup, authSwitchChainList)
    if authSwitchSoup:
        curAuthValue = authSwitchSoup.attrs.get("value", "0")
        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"] = [ parentClassChain ]  
isInputUserOk = self.wait_element_setText_iOS(userFieldQuery)  
if not isInputUserOk:  
    return False  
  
newAuthPasswordValue = newManualProxyValue["authPassword"]  
passwordFieldQuery = {"type": "XCUIElementTypeSecureTextField"}  
passwordFieldQuery["parent_class_chains"] = [ parentClassChain ]  
isInputPasswordOk = self.wait_element_setText_iOS(passwordFieldQuery)  
if not isInputPasswordOk:  
    return False  
  
isUpdateManualOk = True  
return isUpdateManualOk
```

页面：

支持 开启 鉴定 的值的恢复：



设置 自动 时的值:

```

def setAutoProxyValue(self, parentTableSoup, newAutoProxyValue):
    """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="XCUIElementTypeCell">
        <XCUIElementTypeOther type="XCUIElementTypeTextfield">
            <XCUIElementTypeTextField type="XCUIElementTypeTextField">
                <XCUIElementTypeOther type="XCUIElementTypeTextfield">
                    <XCUIElementTypeTextfield type="XCUIElementTypeTextfield">
                        </XCUIElementTypeTextfield>
                </XCUIElementTypeOther>
            </XCUIElementTypeTextField>
        </XCUIElementTypeTextfield>
    </XCUIElementTypeCell>
    .....
    newValue = newAutoProxyValue["url"]
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x ="
    urlFieldQuery = {"type": "XCUIElementTypeTextField", "name": "url"}
    urlFieldQuery["parent_class_chains"] = [parentCellClassChain]
    # foundUrl, respInfo = self.findElement(urlFieldQuery,
    # if not foundUrl:
    #     return False
    isInputUrlOk = self.wait_element_setText_iOS(urlFieldQuery)
    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：



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="XCUIElementTypeNavigationBar">
        <XCUIElementTypeButton type="XCUIElementTypeButton" value="存储"/>
        <XCUIElementTypeOther type="XCUIElementTypeTextfield" value=""/>
        <XCUIElementTypeButton type="XCUIElementTypeButton" value="关闭"/>
    </XCUIElementTypeNavigationBar>
    """
    storeName = "存储"
    parentNavBarClassChain = "/XCUIElementTypeNavigationBar/XCUIElementTypeTextfield"
    storeButtonQuery = {"type": "XCUIElementTypeButton", "name": "存储"}
    storeButtonQuery["parent_class_chains"] = [parentNavBarClassChain]
    foundAndClickedStore = self.findAndClickElement(query=storeButtonQuery)
    isStoredOk = foundAndClickedStore
    return isStoredOk
```

即可。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-21 22:51:33

AppStore

此处整理用wda自动化操作AppStore的相关常见代码段。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-21 13:24:32

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

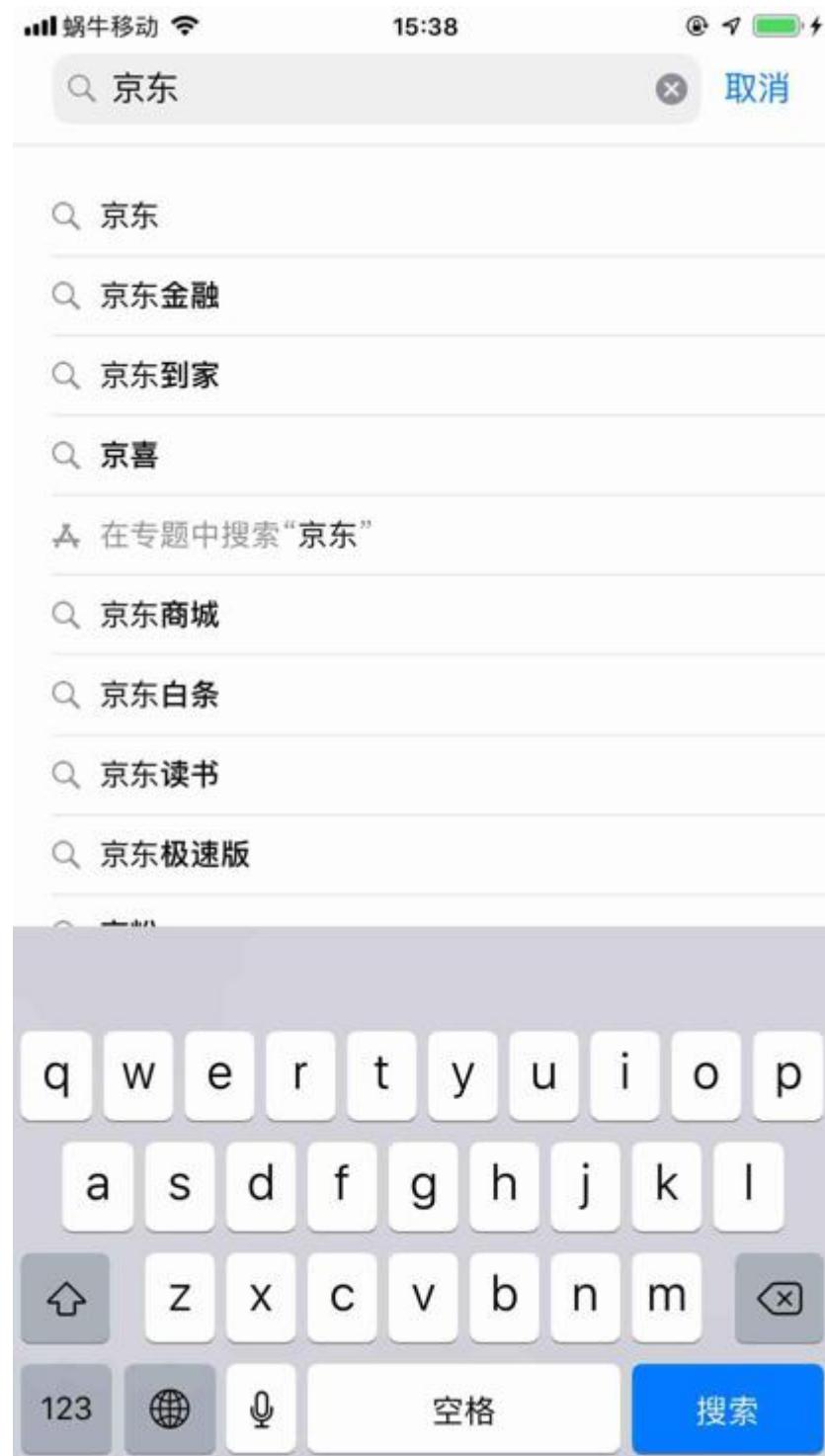
对应着的界面分别是：

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。

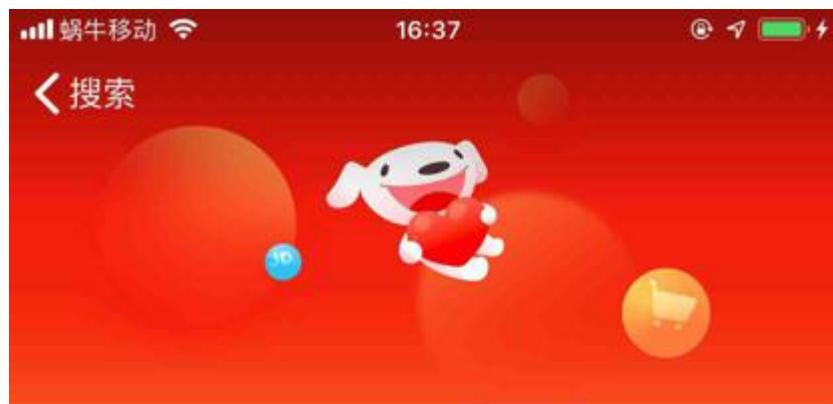
然后再去点击搜索框，输入要搜索的app名字：



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：



4.5 ★★★★★

34.6万个评分

#5

购物

17+

年龄

新功能

版本 9.0.0

京东9.0，全新启航 不负每一份热爱！618 惊喜剧透 火力全开！

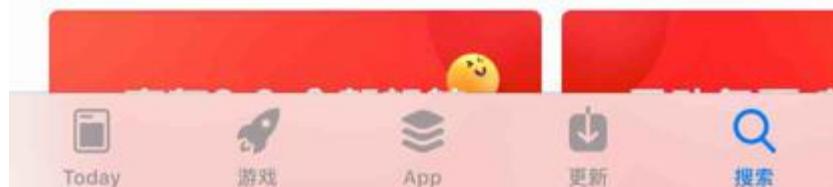
5.29 超嗨预售抢先购

更多

版本历史记录

1 周前

预览



注意：

此处的图标是 云朵 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：

蜗牛移动 16:19

搜索

¥1.00 App 内购买项目

2.9 ★★★★☆
793个评分

#3
工具

4+
年龄

不折叠输入法 —— 充分展现自我

有路 小路 | 优读首席运营
有路笔记：一个一站式智能知识管...
1分钟前 删除

使用前

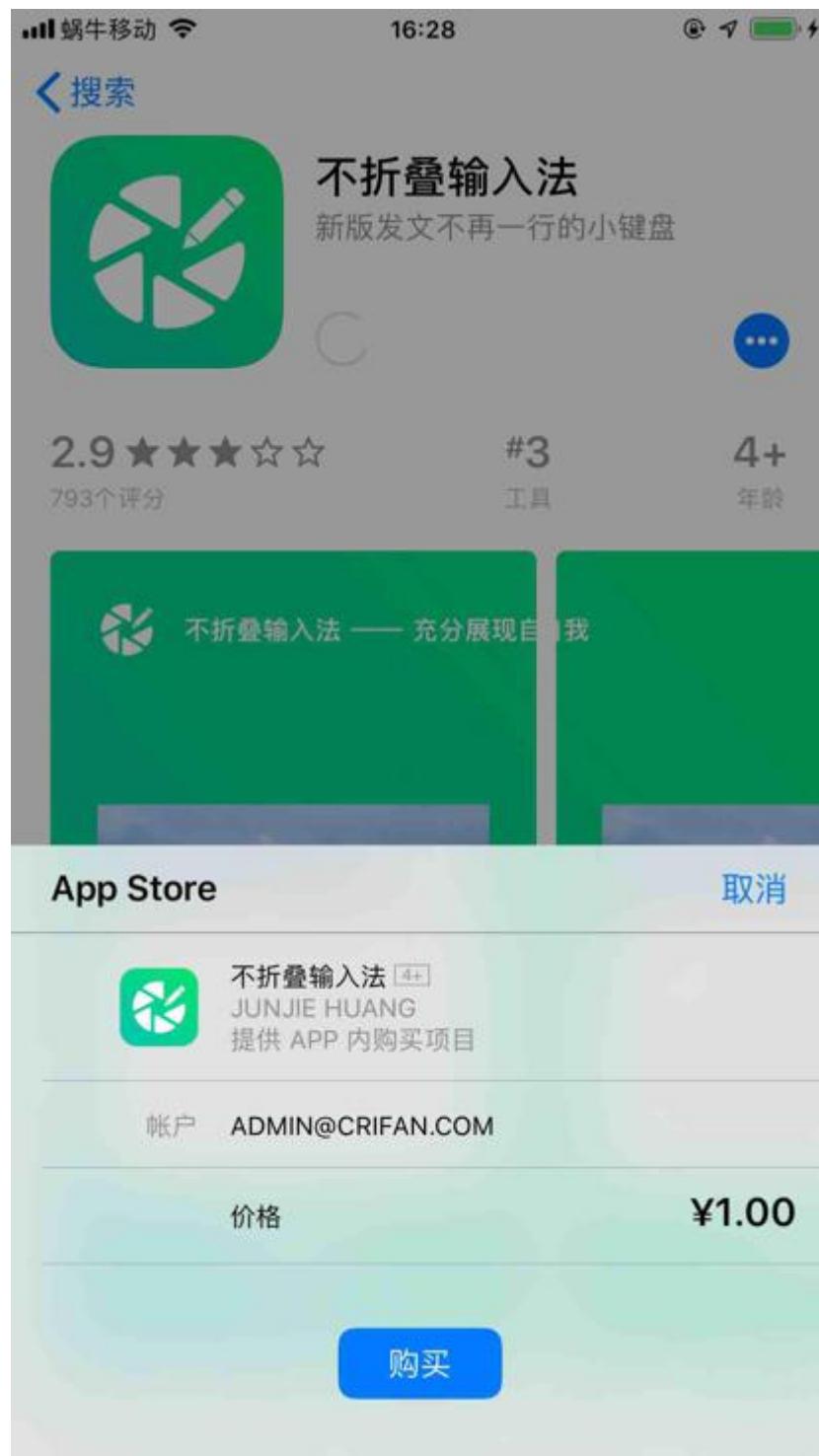
有路 小路 | 优读首席运营
有路笔记：一个一站式的智能平台。通过AI智能算法结合Word、PPT、Excel等维图、流程图等的知识都有价值。
全文
1分钟前 删除

使用后

Today 游戏 App 更新 搜索

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：

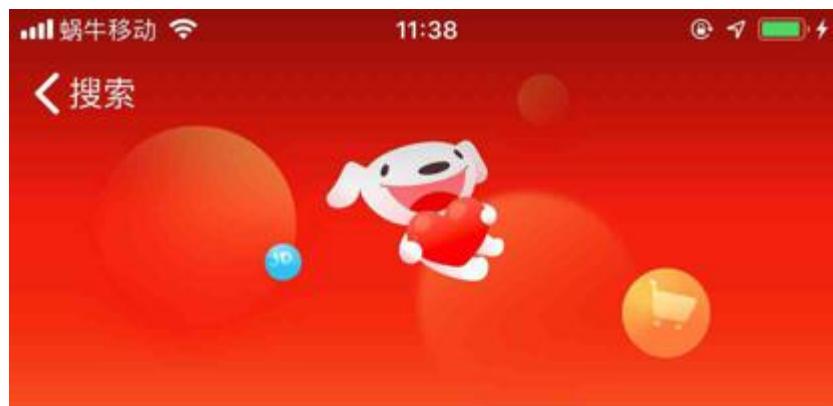


点击购买后，会真的扣费的。

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：



4.5 ★★★★★

35.4万个评分

#4

购物

17+

年龄

新功能

版本 9.0.0

京东9.0，全新启航 不负每一份热爱！618 惊喜剧透 火力全开！

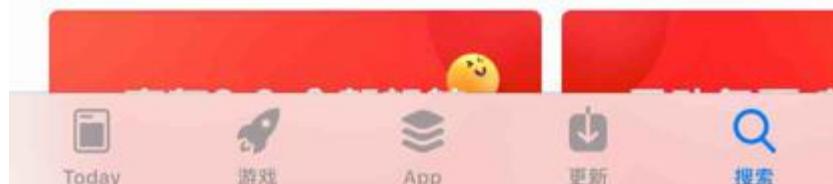
5.29 超嗨预售抢先购

更多

版本历史记录

1 周前

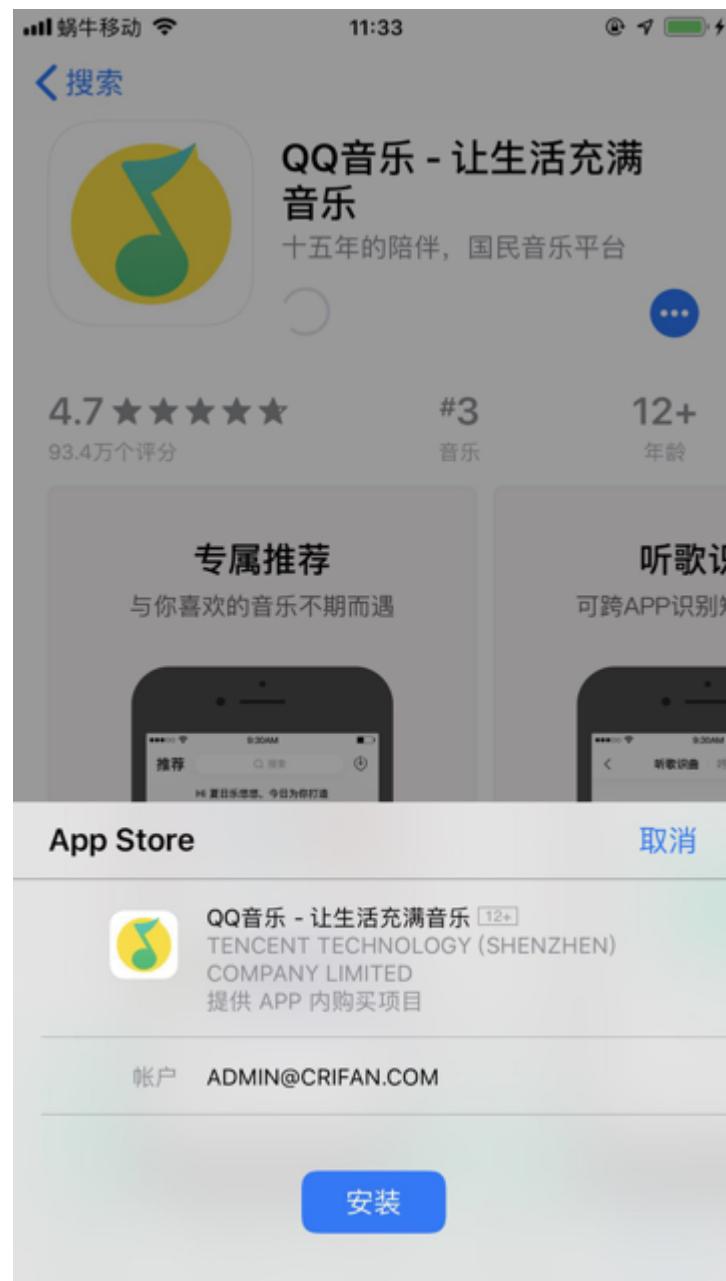
预览



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）





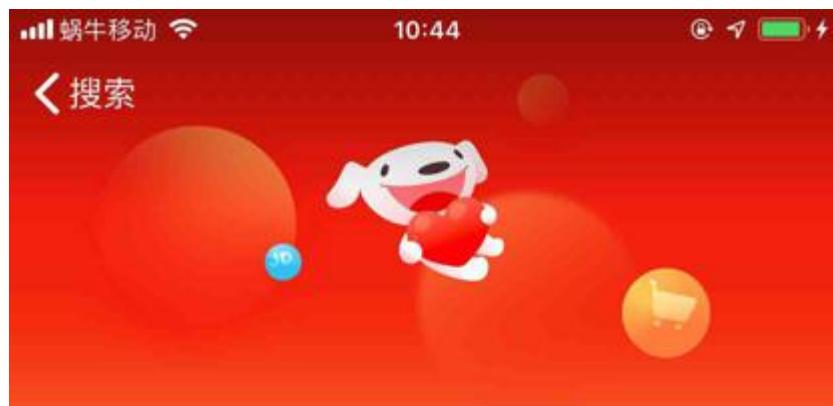
点击安装，弹框会提示完成：

(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：



京东-一起热爱 就现在

新人送188元购物礼包



4.5 ★★★★★

35.4万个评分

#4

购物

17+

年龄

新功能

版本 9.0.0

京东9.0，全新启航 不负每一份热爱！618 惊喜剧透 火力全开！

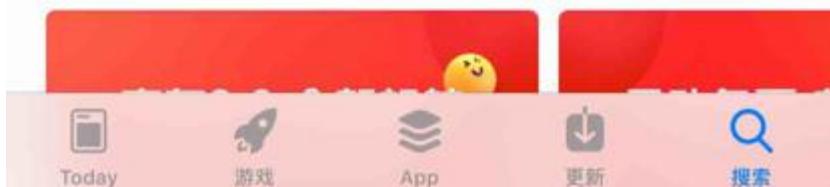
5.29 超嗨预售抢先购

版本历史记录

1 周前

更多

预览



解释：

此处通过调试log日志：

```
[200608 14:22:43] [DevicesMethods.py 1645] start get iOS pac...
[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 pac...
[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 pac...
[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 pac...
[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 pac...
[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 pac...
[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 pac...
[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 pac...
[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：

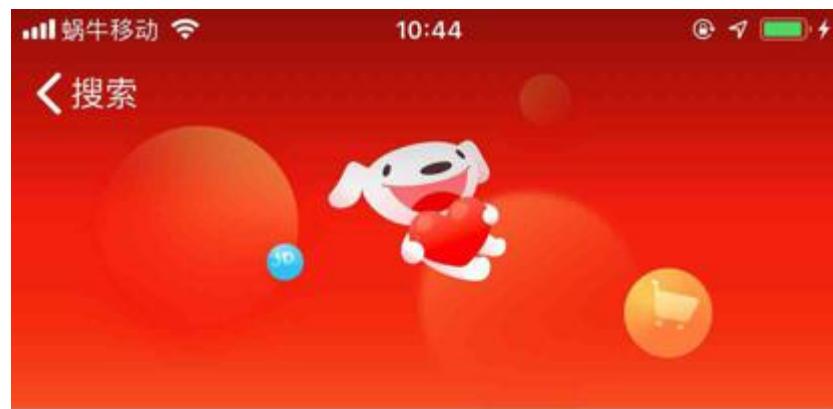
在下载的进度超过76%之后，感觉内部就进入了自动安装过程

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：



4.5 ★★★★★

35.4万个评分

#4

购物

17+

年龄

新功能

版本 9.0.0

京东9.0，全新启航 不负每一份热爱！618 惊喜剧透 火力全开！

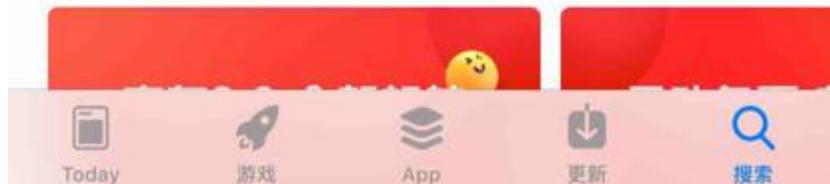
5.29 超嗨预售抢先购

更多

版本历史记录

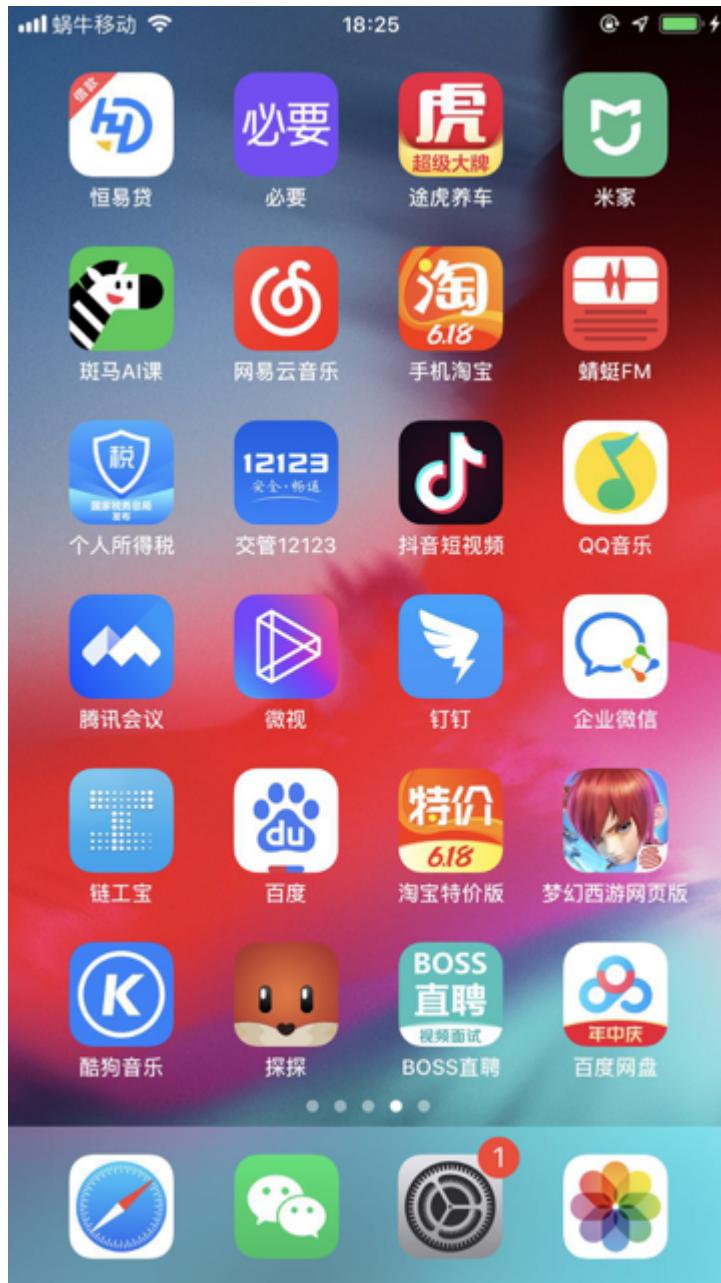
1 周前

预览



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.xiaomi.mihome'}
    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 installed" % appName)
    else:
        # auto install app from AppStore
        isInstallOk, respInfo = self.iOSinstallAppFromAppStore(
            appName=appName)
        logging.debug("appName=%s -> isInstallOk=%s, respInfo=%s" %
                      (appName, isInstallOk, respInfo))
        # {'bundleId': 'com.fenbi.ape.zebstrika', 'name': '斑马AI课'}
        if isInstallOk:
            appInfo = respInfo
        else:
            errMsg = respInfo
            logging.error("Fail to auto install iOS app %s, %s" %
                          (appName, errMsg))

    if appInfo:
        # update bundle id
        installedBundleId = appInfo["bundleId"] # 'com.360safe'
        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_AppStore)
    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 AppStore"
        return isInstallOk, respInfo

    """
    <XCUIElementTypeOther type="XCUIElementTypeOther" >
        <XCUIElementTypeSearchField type="XCUIElementTypeSearchField" >
    </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 field" % appName
        return isInstallOk, respInfo

    isSearchOk = self.search_iOS(wait=0.2)

```

```

    if not isSearchOk:
        respInfo = "Fail to find and click Search button to search"
        return isInstallOk, respInfo

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

    detectRoundNum = 0
    while True:
        detectRoundNum += 1
        logging.info("%s try auto install, round [%d] %s", appName, detectRoundNum, time.strftime("%Y-%m-%d %H:%M:%S"))

        # foundOpen = False
        # isDownloading = False
        # isLoading = False
        # foundAndClickedPopupInstall = False
        # foundAndClickedDownload = False

        # isDownloading, curProgress = self.appStoreDownloading()
        # logging.info("isDownloading=%s, curProgress=%s", isDownloading, curProgress)
        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" % downloadingWaitTime)
            continue

        foundOpen = self.appStoreCheckOpen()
        logging.info("foundOpen=%s", foundOpen)
        if foundOpen:
            logging.info("Found 打开 -> means %s is installed" % appName)
            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", loadingWaitTime)
    continue

foundAndClickedDownload, downloadButtonName = self.getInstallInfo()
logging.info("foundAndClickedDownload=%s, downloadButtonName=%s", foundAndClickedDownload, downloadButtonName)
if foundAndClickedDownload:
    logging.info("Found and clicked button %s to start install", downloadButtonName)
else:
    foundMoneyButton, moneyButtonName = self.appStoreClickPopUp()
    if foundMoneyButton:
        respInfo = "Not support auto install %s for now" % appName
        return isInstallOk, respInfo

foundAndClickedPopupInstall = self.appStoreClickPopUp()
logging.info("foundAndClickedPopupInstall=%s", foundAndClickedPopupInstall)
if foundAndClickedPopupInstall:
    popupInstallWaitTime = 0.2
    time.sleep(popupInstallWaitTime)
    logging.info("After click 安裝 of popup, wait %s seconds", popupInstallWaitTime)

logging.info("Install complete for %s", appName)
installedAppInfo = self.getInstalledAppInfo(appName=appName)
logging.info("installedAppInfo=%s", installedAppInfo)
if not installedAppInfo:
    respInfo = "Fail to extract installed app info for %s" % appName
    return isInstallOk, respInfo

isInstallOk = True
respInfo = installedAppInfo
logging.info("isInstallOk=%s, respInfo=%s", isInstallOk, respInfo)
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=None):
    """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', 'name': '京东手机助手'}
            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 - 嘴嗒出行 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 Off
            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") # '10.1
                curAppInfo = {
                    "bundleId": bundleId,
                    "name": name,
                    "version": version,
                }
                installedAppList.append(curAppInfo)
            else:
                # Total: 9 apps
                if eachAppStr and (not eachAppStr.startswith(
                    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="XCUIElementTypeTabBar">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
        </XCUIElementTypeTabBar>
    .....
    parentTabBarClassChain = "/XCUIElementTypeTabBar[`rect`"
    searchButtonQuery = {"type": "XCUIElementTypeButton", "parent_class_chains": parentTabBarClassChain}
    foundAndClicked = self.findAndClickElement(query=searchButtonQuery)
    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, respInfo)
    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" id="..."/>
    # <XCUIElementTypeButton type="XCUIElementTypeButton" id="..."/>
    # searchButtonQuery = {"name": "Search"}
    searchButtonQuery = {"name": "Search", "type": "XCUIElementTypeButton"}
    # Note: occasionally not found Search, change to find more
    MaxRetryNumber = 5
    curRetryNumber = MaxRetryNumber
    while curRetryNumber > 0:
        foundAndClickedDoSearch = self.findAndClickElement(
            searchButtonQuery)
        if foundAndClickedDoSearch:
            break
        curRetryNumber -= 1

    if not foundAndClickedDoSearch:
        logging.error("Not found and/or clicked for %s", self)
    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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeCollectionView type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeButton type="XCUIElementTypeCell">
                    </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeButton type="XCUIElementTypeCell">
                    </XCUIElementTypeCell>
                </XCUIElementTypeCell>
            </XCUIElementTypeCollectionView>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

    搜索结果列表页 美团 获取:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeCollectionView type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeButton type="XCUIElementTypeCell">
                    </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeButton type="XCUIElementTypeCell">
                    </XCUIElementTypeCell>
                </XCUIElementTypeCell>
            </XCUIElementTypeCollectionView>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    .....
    parentCollectionViewClassChain = "/XCUIElementTypeCollectionView"
    firstMatchCellQuery = {"type": "XCUIElementTypeCell", "isFirstMatch": true}
    firstMatchCellQuery["parent_class_chains"] = [parentCollectionViewClassChain]
    foundAndClicked = self.findAndClickElement(query=firstMatchCellQuery)
    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="XCUIElementTypeCollection"
        <XCUIElementTypeOther type="XCUIElementTypeCell"
            <XCUIElementTypeCell type="XCUIElementTypeCell"
                <XCUIElementTypeImage type="XCUIElementTypeImage"
                <XCUIElementTypeStaticText type="XCUIElementTypeText"
                <XCUIElementTypeStaticText type="XCUIElementTypeText"
                <XCUIElementTypeStaticText type="XCUIElementTypeText"
                <XCUIElementTypeStaticText type="XCUIElementTypeText"
                <XCUIElementTypeButton type="XCUIElementTypeText"
                <XCUIElementTypeStaticText type="XCUIElementTypeText"
                <XCUIElementTypeButton type="XCUIElementTypeText"
            </XCUIElementTypeCell>
        </XCUIElementTypeCell>
    """
    # Note: here change wda query to bs.find, then revert to
    # for later bs.find will need get page source, which takes
    # time
    parentCellClassChain = "/XCUIElementTypeCell[`rect.x ="
    downloadingButtonQuery = {"type": "XCUIElementTypeButton"}
    downloadingButtonQuery["parent_class_chains"] = [parentCellClassChain]
    isfound, respInfo = self.findElement(query=downloadingButtonQuery)
    # 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, then
    # # 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, isDownloading)
# 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 complete to open"""

    Args:
        Returns:
            bool
    Raises:
    """
    foundOpen = False
    """
        AppStore 详情页 京东 打开:
        <XCUIElementTypeCollectionView type="XCUIElementTypeCollection">
            <XCUIElementTypeOther type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeButton type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeButton type="XCUIElementTypeText">
                    <XCUIElementTypeOther type="XCUIElementTypeText">
                </XCUIElementTypeCell>
            </XCUIElementTypeCell>
        </XCUIElementTypeCollectionView>
    """

    parentCellClassChain = "/XCUIElementTypeCell[`rect.x ="
    openButtonQuery = {"type": "XCUIElementTypeButton", "name": "打开"}  # 检查是否有“打开”按钮
    openButtonQuery["parent_class_chains"] = [parentCellClassChain]
    foundOpen, respInfo = self.findElement(query=openButtonQuery)
    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="XCUIElementTypeCollection">
            <XCUIElementTypeOther type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeButton type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeButton type="XCUIElementTypeText">
                    <XCUIElementTypeOther type="XCUIElementTypeText">
                </XCUIElementTypeCell>
            """
        parentCellClassChain = "/XCUIElementTypeCell[`rect.x ="
        loadingButtonQuery = {"type": "XCUIElementTypeButton", "rect": {
            "x": 100, "y": 200, "width": 100, "height": 50}}
        loadingButtonQuery["parent_class_chains"] = [parentCellClassChain]
        foundLoading, respInfo = self.findElement(query=loadingButtonQuery)
    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
    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="XCUIElementTypeCollection">
        <XCUIElementTypeOther type="XCUIElementTypeCell">
            <XCUIElementTypeCell type="XCUIElementTypeCell">
                <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                <XCUIElementTypeButton type="XCUIElementTypeElement">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                        <XCUIElementTypeButton type="XCUIElementTypeElement">
                                            <XCUIElementTypeOther type="XCUIElementTypeElement">
                                                </XCUIElementTypeCell>
                                            <XCUIElementTypeOther type="XCUIElementTypeElement">
                                                <XCUIElementTypeButton type="XCUIElementTypeElement">
                                                    </XCUIElementTypeOther>
                                                <XCUIElementTypeCell type="XCUIElementTypeCell">

```

AppStore 详情页 钉钉 获取:

```

    <XCUIElementTypeCollectionView type="XCUIElementTypeCollection">
        <XCUIElementTypeOther type="XCUIElementTypeCell">
            <XCUIElementTypeCell type="XCUIElementTypeCell">
                <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                <XCUIElementTypeButton type="XCUIElementTypeElement">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                        <XCUIElementTypeButton type="XCUIElementTypeElement">
                                            <XCUIElementTypeOther type="XCUIElementTypeElement">
                                                </XCUIElementTypeCell>
                                            <XCUIElementTypeOther type="XCUIElementTypeElement">
                                                <XCUIElementTypeButton type="XCUIElementTypeElement">
                                                    </XCUIElementTypeOther>
                                                <XCUIElementTypeCell type="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, downloadChainList)
# if downloadSoup:
#     self.clickElementCenterPosition(downloadSoup)
#     foundAndClickedDownload = True
# return foundAndClickedDownload

# Note: to avoid possible later get page slow or even time out
# change to wda query
parentCellClassChain = "/XCUIElementTypeCell[`rect.x = %d` and `rect.y = %d`]"
downloadButtonNameList = ["获取", "重新下载"]
for eachDownloadButtonName in downloadButtonNameList:
    curDownloadButtonQuery = {"type": "XCUIElementTypeButton", "name": eachDownloadButtonName}
    curDownloadButtonQuery["parent_class_chains"] = [parentCellClassChain]
    foundAndClicked = self.findAndClickElement(query=curDownloadButtonQuery)
    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="XCUIElementTypeCollection">
            <XCUIElementTypeOther type="XCUIElementTypeCell">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                <XCUIElementTypeButton type="XCUIElementTypeElement">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                            <XCUIElementTypeButton type="XCUIElementTypeElement">
                                                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                                    <XCUIElementTypeButton type="XCUIElementTypeElement">
                                                        <XCUIElementTypeOther type="XCUIElementTypeElement">
                                                            </XCUIElementTypeCell>
                                                        </XCUIElementTypeElement>
                                                    </XCUIElementTypeButton>
                                                </XCUIElementTypeText>
                                            </XCUIElementTypeText>
                                        </XCUIElementTypeButton>
                                    </XCUIElementTypeText>
                                </XCUIElementTypeText>
                            </XCUIElementTypeButton>
                        </XCUIElementTypeText>
                    </XCUIElementTypeImage>
                </XCUIElementTypeCell>
            </XCUIElementTypeCell>
        </XCUIElementTypeCollectionView>
    """

    parentCellClassChain = "/XCUIElementTypeCell[@"rect.x ="
    moneyButtonQuery = {"type": "XCUIElementTypeButton", "name": ""}
    moneyButtonQuery["parent_class_chains"] = [parentCellClassChain]
    foundMoneyButton, respInfo = self.findElement(query=moneyButtonQuery)
    if foundMoneyButton:
        curElement = respInfo
        # internally use first non-empty of element.wdValue
        # curName = curElement.text
        # so change to name, Note: has changed facebook-wad
        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 install
    Args:
        Returns:
            bool
    Raises:
    """
    .....
    .....
    AppStore 详情页 淘宝 弹框 安装:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeTable type="XCUIElementTypeTable">
            <XCUIElementTypeCell type="XCUIElementTypeCell">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            </XCUIElementTypeCell>
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            </XCUIElementTypeCell>
                </XCUIElementTypeTable>
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                            </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        .....
        parent0therClassChain = "/XCUIElementTypeOther[`rect.x`"
        installButtonQuery = {"type": "XCUIElementTypeButton", "parent0therClassChain": parent0therClassChain}
        installButtonQuery["parent_class_chains"] = [parent0therClassChain]
        foundAndClickedInstall = self.findAndClickElement(query=installButtonQuery)
        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.iOSProcessPopupUpperLeft()
    # if not foundAndProcessedPopup:
    #     foundAndProcessedPopup = self.iOSProcessPopupUpperRight()
    # Note: put common upper right close, for some app, eg
    if not foundAndProcessedPopup:
        # foundAndProcessedPopup = self.iOSProcessPopupUpperRight()
        foundAndProcessedPopup = self.iOSProcessPopupCommon()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupPossibly()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupUnder()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupIKnow()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessCommonUser()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessUserPrivacy()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessAlertUseWith()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessCommonAlert()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessCommonAlert()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessRightUpper()

    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_Finder()

    if self.isSpecialiOSApp_kags:
        if not foundAndProcessedPopup:
            foundAndProcessedPopup = self.iOSProcessKagsPopup()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupPhone()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessSettingsAll()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupCert()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupNetwork()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupNetwork()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupSystem()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupServer()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupWeixin()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupJumpTo()

    if not foundAndProcessedPopup:
        # foundAndProcessedPopup = self.iOSProcessPopupGet()
        foundAndProcessedPopup = self.iOSProcessPopupDoSome()

    if not foundAndProcessedPopup:
        # foundAndProcessedPopup = self.iOSProcessPopupMinimize()
        foundAndProcessedPopup = self.iOSProcessPopupMinimize()

    if not foundAndProcessedPopup:
        foundAndProcessedPopup = self.iOSProcessPopupWeixin()

```

```
if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessPopupMinip

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessPopupMinip

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessWillOpenNote

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessAlertNoteC

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessAlertCancel

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessAlertCommon

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessPopupPhoto

# 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.iOSProcessPopupSover

if not foundAndProcessedPopup:
    foundAndProcessedPopup = self.iOSProcessPopupCommon

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="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeImage">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    </XCUIElementTypeButton>
                                </XCUIElementTypeText>
                            </XCUIElementTypeText>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeImage>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    """

    immediatelyExperienceChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "name": "立即体验"}
        }
    ]
    foundAndProcessedPopup = self.findAndClickCenterPosition(immediatelyExperienceChainList)
    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="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeStaticText type="XCUIElementTypeText" value="米家隐私政策" />
                <XCUIElementTypeTextView type="XCUIElementTypeText" value="确定" />
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    """
    confirmChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "name": "确定"}
        },
    ]
    confirmSoup = CommonUtils.bsChainFind(soup, confirmChainList)
    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="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeTable">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeImage type="XCUIElementTypeImage" alt="保存" />
                        <XCUIElementTypeSearchField type="XCUIElementTypeSearchField" value="保存" />
                    </XCUIElementTypeOther>
                    <XCUIElementTypeOther type="XCUIElementTypeTable">
                        <XCUIElementTypeStaticText type="XCUIElementTypeText" value="地区选择" />
                        <XCUIElementTypeOther type="XCUIElementTypeTable">
                            <XCUIElementTypeOther type="XCUIElementTypeTable">
                                <XCUIElementTypeTable type="XCUIElementTypeTable" />
                                <XCUIElementTypeTable type="XCUIElementTypeTable" />
                            </XCUIElementTypeTable>
                            <XCUIElementTypeTable type="XCUIElementTypeTable" />
                            <XCUIElementTypeTable type="XCUIElementTypeTable" />
                        </XCUIElementTypeTable>
                    </XCUIElementTypeTable>
                    <XCUIElementTypeButton type="XCUIElementTypeButton" value="保存" />
                    <XCUIElementTypeButton type="XCUIElementTypeButton" value="取消" />
                </XCUIElementTypeTable>
            </XCUIElementTypeOther>
        </XCUIElementTypeTable>
    """
    commonSaveChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeTable",
            "attrs": self.FullScreenAttrDict
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "name": "保存"}
        }
    ]
    commonSaveSoup = CommonUtils.bsChainFind(soup, commonSaveChainList)
    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="XCUIElementTypeWindow"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeOther type="XCUIElementTypeImage"
                <XCUIElementTypeOther type="XCUIElementTypeStaticText"
                <XCUIElementTypeButton type="XCUIElementTypeButton"
                <XCUIElementTypeButton type="XCUIElementTypeImage"
            </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, commonCancelChainList)
    if commonCancelSoup:
        self.clickElementCenterPosition(commonCancelSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupSovereignHasRead

```
def iOSProcessPopupSovereignHasRead(self, soup):
    """
        必要 弹框 被已阅
    """
    foundAndProcessedPopup = False
    """
        必要 弹框 被已阅:
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                        <XCUIElementTypeTextValue type="XCUIElementTypeTextValue">
                            <XCUIElementTypeButton type="XCUIElementTypeButton">
                                </XCUIElementTypeImage>
                            </XCUIElementTypeTextValue>
                        </XCUIElementTypeText>
                    </XCUIElementTypeImage>
                </XCUIElementTypeOther>
    """
    hasReadChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "text": "被已阅"}
        },
        {
            "tag": "XCUIElementTypeImage",
            "attrs": {"enabled": "true", "visible": "true", "text": "被已阅"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "text": "被已阅"}
        },
    ]
    hasReadCloseSoup = CommonUtils.bsChainFind(soup, hasReadChainList)
    if hasReadCloseSoup:
        self.clickElementCenterPosition(hasReadCloseSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessAlertCommonGiveup


```
        "tag": "XCUIElementTypeAlert",
        "attrs": {"enabled":"true", "visible":"true"}
    },
{
    "tag": "XCUIElementTypeOther",
    "attrs": {"enabled":"true", "visible":"true"}
},
{
    "tag": "XCUIElementTypeButton",
    "attrs": {"enabled":"true", "visible":"true", "label": "完成"}
},
]
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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeButton type="XCUIElementTypeOther">
                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementTypeImage">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                            </XCUIElementTypeOther>
                        <XCUIElementTypeOther type="XCUIElementTypeImage">
                            <XCUIElementTypeCollectionView type="XCUIElementTypeImage">
                                <XCUIElementTypeButton type="XCUIElementTypeImage">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                        </XCUIElementTypeOther>
                                    </XCUIElementTypeImage>
                                </XCUIElementTypeImage>
                            </XCUIElementTypeCollectionView>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeImage>
                </XCUIElementTypeImage>
            </XCUIElementTypeButton>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    busyRetryChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "text": "京东金融 异常页面 服务器繁忙, 请稍后重试"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "text": "京东金融 异常页面 服务器繁忙, 请稍后重试"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled": "true", "visible": "true", "text": "京东金融 异常页面 服务器繁忙, 请稍后重试"}
        },
    ]
    busyRetrySoup = CommonUtils.bsChainFind(soup, busyRetryChainList)
    if busyRetrySoup:
        # find back button
        parentOtherSoup = busyRetrySoup.parent
        parentParentOtherSoup = parentOtherSoup.parent
        if parentParentOtherSoup:
            backupP = re.compile("backup") # "com icon bac
            backupSoup = parentParentOtherSoup.find(
                "XCUIElementTypeButton",
                attrs={"visible": "true", "enabled": "true", "text": "京东金融 异常页面 服务器繁忙, 请稍后重试"})
            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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeImage">
                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    </XCUIElementTypeImage>
                </XCUIElementTypeStaticText>
            </XCUIElementTypeOther>
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeImage">
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    .....
    retryRefreshChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "text": "再刷新下"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "text": "稍后再试"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "text": "系统正在开小差"}
        },
    ]
    retryRefreshSoup = CommonUtils.bsChainFind(soup, retryRefreshChainList)
    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="XCUIElementTypeWindow">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeImage">
                        <XCUIElementTypeImage type="XCUIElementTypeImage">
                            <XCUIElementTypeButton type="XCUIElementTypeButton">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeButton>
                            </XCUIElementTypeImage>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
    """

    specialChar = "□"
    underCloseChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": self.FullScreenAttrDict,
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": self.FullScreenAttrDict,
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "name": "□"},
        },
    ]
    underCloseSoup = CommonUtils.bsChainFind(soup, underCloseChainList)
    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 smaller than
    isSizeValid = PopupWindowSizeMinRatio <= curSizeRatio <= PopupWindowSizeMaxRatio
    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 <= NormalButtonSizeMax
    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10%
        以及 w和h都在一定（普通关闭按钮的大小）范围，比如 10*10 ~ 30*30
    """

    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，算作关闭按钮
        意思是：引导页往往伴随着一个普通大小的按钮，供关闭用

    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="XCUIElementTypeWindow">
            ...
            <XCUIElementTypeOther type="XCUIElementTypeOther" name="关闭" w="10" h="10" ...>
    
```

```

        <XCUIElementTypeButton type="XCUIElementTypeButton">
            <XCUIElementTypeImage type="XCUIElementTypeImage">
                </XCUIElementTypeImage>
            </XCUIElementTypeButton>
        </XCUIElementTypeWindow>

必要 我的 蒙层弹框 首次使用引导页:
<XCUIElementTypeWindow type="XCUIElementTypeWindow">
    . . .
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeImage type="XCUIElementTypeImage">
                            <XCUIElementTypeButton type="XCUIElementTypeButton">
                                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                                            </XCUIElementTypeButton>
                                        </XCUIElementTypeOther>
                                    </XCUIElementTypeStaticText>
                                </XCUIElementTypeButton>
                            </XCUIElementTypeImage>
                        </XCUIElementTypeImage>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeWindow>

必要 弹框 推荐开通以下授权 关闭按钮在弹框下面:
<XCUIElementTypeApplication type="XCUIElementTypeApplication">
    <XCUIElementTypeWindow type="XCUIElementTypeWindow">
        . . .
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeImage type="XCUIElementTypeImage">
                            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                        </XCUIElementTypeOther>
                                    </XCUIElementTypeStaticText>
                                </XCUIElementTypeStaticText>
                            </XCUIElementTypeImage>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                </XCUIElementTypeButton>
            </XCUIElementTypeWindow>
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                </XCUIElementTypeButton>
        . . .
        # noNameP = re.compile("???")
        noNameP = False
        possibleCloseChainList = [
            {
                "tag": "XCUIElementTypeWindow",
                "attrs": {"enabled": "true", "visible": "true", "name": "蒙层弹框首次使用引导页"}
            },
            {
                "tag": "XCUIElementTypeOther",
                "attrs": {"enabled": "true", "visible": "true", "name": "蒙层弹框首次使用引导页"}
            },
            {
                "tag": "XCUIElementTypeButton",
                "attrs": {"enabled": "true", "visible": "true", "name": "蒙层弹框首次使用引导页"}
            }
        ]
    </XCUIElementTypeApplication>

```

```

        "attrs": {"enabled": "true", "visible": "true", '},
    ],
possibleCloseSoup = CommonUtils.bsChainFind(soup, poss:

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, (
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 # {'enat
        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 <= Button
        isValidHeight = ButtonMinWH <= height <= Button
        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 <= MaxAllowDiffRatio
            isAlmostSameWH = isValidDiff

    isPossibleClose = isAlmostSameWH

    if isPossibleClose:
        # check next sibling is large Image
        # nextSiblingeSoup = possibleCloseSoup.next_sibling
        # nextSiblingeSoupList = possibleCloseSoup.next_siblings
        nextSiblingeSoupGenerator = possibleCloseSoup.next_siblings
        nextSiblingeSoupList = list(nextSiblingeSoupGenerator)

        hasLargeImage = CommonUtils.isContainSpecificSize(nextSiblingeSoupList)
        isPossibleClose = hasLargeImage

    if not isPossibleClose:
        hasNormalButton = CommonUtils.isContainSpecificName(nextSiblingeSoupList)
        isPossibleClose = hasNormalButton

    if not isPossibleClose:
        if isCloseUnderPopup:
            # prevSiblingSoupGenerator = possibleCloseSoup.previous_siblings
            # prevPrevSiblingSoupList = list(prevSiblingSoupGenerator)
            # hasLargeOther = CommonUtils.isContainSpecificSize(nextSiblingeSoupList)
            prevSiblingSoup = possibleCloseSoup.previous_sibling
            if prevSiblingSoup:
                prevPrevSiblingSoup = prevSiblingSoup.previous_sibling
                prevPrevSiblingSoupList = [prevPrevSiblingSoup]
                hasLargeOther = CommonUtils.isContainSpecificSize(prevPrevSiblingSoupList)
                isPossibleClose = hasLargeOther

    if isPossibleClose:
        foundAndProcessedPopup = self.findAndClickButton()

return foundAndProcessedPopup

```

iOSProcessPopupCommonNameContainClose


```
        "tag": "XCUIElementTypeOther",
        # "attrs": {"enabled":"true", "visible":"true"},
        # "attrs": {"enabled":"true", "visible":"true"}
        "attrs": {"enabled":"true", "visible":"true"}
    },
{
    "tag": "XCUIElementTypeButton",
    "attrs": {"enabled":"true", "visible":"true", "label": "关闭", "type": "button", "name": "close", "value": "Close", "color": "#000000", "size": "large", "font": "System", "bold": true}
},
]
commonCloseSoup = CommonUtils.bsChainFind(soup, commonClose)
if commonCloseSoup:
    # self.clickElementCenterPosition(commonCloseSoup)
    # foundAndProcessedPopup = True

    # so change to wda query element then click
    foundAndProcessedPopup = self.findAndClickButtonElement(commonCloseSoup)
return foundAndProcessedPopup
```

iOSProcessPopupUpperRightAlertClose

```

# def iOSProcessPopupUpperRightAlertClose(self, soup):
#     foundAndProcessedPopup = False
#
#     京东金融 tab2财富 弹框 我是Max alertClose:
#         <XCUIElementTypeOther type="XCUIElementTypeOther">
#             ...
#                 <XCUIElementTypeOther type="XCUIElementTypeOther">
#                     <XCUIElementTypeButton type="XCUIElementTypeButton">
#                         ...
#                         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, alertCloseP)
#                         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="XCUIElementTypeOther">
#             <XCUIElementTypeOther type="XCUIElementTypeOther">
#                 <XCUIElementTypeStaticText type="XCUIElementTypeText">
#                     <XCUIElementTypeButton type="XCUIElementTypeButton">
#                         ...
#                         containCloseP = re.compile("close") # com icon black
#                         closeChainList = [
#                             {
#                                 "tag": "XCUIElementTypeOther",
#                                 "attrs": {"enabled": "true", "visible": "true"},
#                             },
#                             {
#                                 "tag": "XCUIElementTypeOther",
#                                 "attrs": {"enabled": "true", "visible": "true"},
#                             },
#                         ]
#                         containCloseSoup = CommonUtils.bsChainFind(soup, containCloseP)
#                         if containCloseSoup:
#                             self.clickElementCenterPosition(containCloseSoup)
#                             foundAndProcessedPopup = True
#                         return foundAndProcessedPopup

```

```

# {
#     "tag": "XCUIElementTypeButton",
#     "attrs": {"enabled":"true", "visible":"true"},
# },
#
# closeSoup = CommonUtils.bsChainFind(soup, closeChain)
# 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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                </XCUIElementTypeButton>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    .....
    jumpOverChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "name": "京东金融"},

        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true", "name": "京东金融"},

        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "name": "京东金融"},

        },
    ]
    jumpOverSoup = CommonUtils.bsChainFind(soup, jumpOverChainList)
    if jumpOverSoup:
        self.clickElementCenterPosition(jumpOverSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup

```

iOSProcessAlertUseWirelessData


```
        </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", "label": "OK"}
        },
    ]
    wifiCellularSoup = CommonUtils.bsChainFind(soup, wifiCellularChainList)
    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 element
        # curName = wifiCellularSoup.attrs["name"] # 好
        # wifiCellularButtonQuery = {"type": "XCUIElementTypeButton", "label": "OK"}
        # foundAndClicked = self.findAndClickElement(wifiCellularSoup, wifiCellularButtonQuery)
        # foundAndProcessedPopup = foundAndClicked
        foundAndProcessedPopup = self.findAndClickButtonElement(wifiCellularSoup)
    return foundAndProcessedPopup
```

iOSProcessCommonAlertOk


```

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(okButti
    # foundAndProcessedPopup = foundAndClicked
    foundAndProcessedPopup = self.findAndClickButtonEle

return foundAndProcessedPopup

```

iOSProcessCommonAlertAllow

恒易贷 想给您发送通知 允许:

```

                <XCUIElementTypeOther type="XCUIElementTypeAlert">
                </XCUIElementTypeOther>
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                                 ...

```

....

```

# 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", "type": "button"}
    },
]
allowSoup = CommonUtils.bsChainFind(soup, allowChainList)
if allowSoup:
    # clickCenterPosition(curSession, allowSoup.attrs)
    # foundAndProcessedPopup = True

    # # above click position not work for 允许 -> actual
    # # change to use wda to find 允许 then click element
    # curName = allowSoup.attrs["name"] # 允许 / 始终允许
    # # allowButtonQuery = {"type": "XCUIElementTypeButton"}
    # allowButtonQuery = {"type": "XCUIElementTypeButton"}
    # foundAndClickedAllow = self.findAndClickElement(a

```

```

    # foundAndProcessedPopup = foundAndClickeAllow
    foundAndProcessedPopup = self.findAndClickButtonElement()
    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 用户意愿保护协议 我接收:
        <XCUIElementTypeWindow type="XCUIElementTypeWindow">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                                            </XCUIElementTypeButton>
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeStaticText>
                        </XCUIElementTypeStaticText>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
    """

    userPrivacyIAcceptChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": {"enabled": "true", "visible": "true", "name": "恒易贷 用户意愿保护协议 我接收:窗口"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "name": "恒易贷 用户意愿保护协议 我接收:子元素1"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "name": "恒易贷 用户意愿保护协议 我接收:按钮元素"}
        }
    ]

    userPrivacyIAcceptSoup = CommonUtils.bsChainFind(soup, userPrivacyIAcceptChainList)
    if userPrivacyIAcceptSoup:
        self.clickElementCenterPosition(userPrivacyIAcceptSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup

```

iOSProcessCommonUserAgreementAgre e

```

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="XCUIElementTypeOther">
            ...
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                            <XCUIElementTypeButton type="XCUIElementTypeButton">
                                </XCUIElementTypeButton>
                            </XCUIElementTypeButton>
                        </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>

        必要 用户服务协议及必要隐私政策 同意:
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                <XCUIElementTypeTextView type="XCUIElementTypeTextView">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                            <XCUIElementTypeButton type="XCUIElementTypeButton">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    </XCUIElementTypeButton>
                                </XCUIElementTypeButton>
                            </XCUIElementTypeButton>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>

        斑马AI课 弹框 个人信息保护政策 同意:
        <XCUIElementTypeWindow type="XCUIElementTypeWindow">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText">
                            <XCUIElementTypeTextView type="XCUIElementTypeTextView">
                                <XCUIElementTypeLink type="XCUIElementTypeLink">
                                <XCUIElementTypeLink type="XCUIElementTypeLink">
                                    <XCUIElementTypeLink type="XCUIElementTypeLink">
                                        </XCUIElementTypeLink>
                                    </XCUIElementTypeLink>
                                </XCUIElementTypeLink>
                            </XCUIElementTypeTextView>
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeButton type="XCUIElementTypeButton">
                                    </XCUIElementTypeButton>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeWindow>
        """
    commonAgreeChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "name": "同意", "type": "button", "label": "同意", "value": "同意", "color": "#000000", "font_size": "14px", "font_weight": "bold", "text_color": "#000000", "background_color": "#F0F0F0", "border_color": "#000000", "border_width": "1px", "border_radius": "5px", "padding": "10px", "margin": "10px", "width": "100px", "height": "40px", "x": "50%", "y": "50%"}, "value": "同意"
        }
    ]

```

```
        },
        {
            "tag": "XCUIElementTypeOther",
            # "attrs": {"enabled":"true", "visible":"true"},
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "type": "button"}
        },
    ]
commonAgreeSoup = CommonUtils.bsChainFind(soup, commonAgreeXpath)
if commonAgreeSoup:
    self.clickElementCenterPosition(commonAgreeSoup)
    foundAndProcessedAgreement = True
return foundAndProcessedAgreement
```

iOSProcessPopupIKnow


```
        </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, iKnowChainList)
    if iKnowSoup:
        self.clickElementCenterPosition(iKnowSoup)
        foundAndProcessedPopup = True
    return foundAndProcessedPopup
```

iOSProcessPopupPhoneCall


```
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled":"true", "visible":"true", "label": "取消"}
        }
    ]
callSoup = CommonUtils.bsChainFind(soup, callChainList)
if callSoup:
    # parentOtherSoup = callSoup.parent
    # if parentOtherSoup:
    #     parentParentOtherSoup = parentOtherSoup.parent
    #     if parentParentOtherSoup:
    #         cancelSoup = parentParentOtherSoup.find(
    #             "XCUIElementTypeButton",
    #             attrs={"enabled":"true", "visible":"true"})
    #         if cancelSoup:
    #             clickCenterPosition(curSession, cancelSoup)
    #             foundAndProcessedPopup = True

    # # above click position not work for 取消 !!!
    # # change to find 取消 then click element
    # cancelButtonQuery = {"type":"XCUIElementTypeButton", "label": "取消"}
    # foundAndClickeCancel = self.findAndClickElement(cancelButtonQuery)
    # foundAndProcessedPopup = foundAndClickeCancel
    foundAndProcessedPopup = self.findAndClickButtonElement("取消")
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="4"
    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="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                                                <XCUIElementTypeOther type="XCUIElementTypeOther">
                                                                
    .....
    allTopOtherSoupList = soup.find_all(
        "XCUIElementTypeOther",
        attrs={"enabled": "true", "visible": "true", "x": "0"
    )
    for eachTopOtherSoup in allTopOtherSoupList:
        popupTopOtherSoup = self.findPopupTopFrameElement(eachTopOtherSoup)
        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 - PossibleCloseButtonWidth
            # possibleCloseButtonY0 = curY + PossibleCloseButtonHeight
            possibleCloseButtonY0 = curY
            possibleCloseButtonX1 = curX1
            possibleCloseButtonY1 = curY + PossibleCloseButtonHeight
            # possibleCloseButtonCenterX = possibleCloseButtonX0 + (possibleCloseButtonWidth / 2)
            # possibleCloseButtonCenterY = possibleCloseButtonY0 + (possibleCloseButtonHeight / 2)
            possibleCloseButtonCenterX = int((possibleCloseButtonX0 + possibleCloseButtonX1) / 2)
            possibleCloseButtonCenterY = int((possibleCloseButtonY0 + possibleCloseButtonY1) / 2)
            possibleCloseButtonCenterPositon = (possibleCloseButtonCenterX, possibleCloseButtonCenterY)
            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[eachToCompareKey]
            curValue = curAttrDict.get(eachToCompareKey)
            if curValue != eachToCompareValue:
                allAttrSame = False
                break
```

```
if not allAttrSame:  
    break  
  
curSoup = firstOnlyOtherSoup  
  
curFullScreenOtherLevel += 1  
  
# if curFullScreenOtherLevel >= MaxFullScreenSizeLevel:  
# isPossiblePopup = FullScreenSizeMinLevel <= curFullScreenOtherLevel  
isPossiblePopup = curFullScreenOtherLevel == FullScreenSizeLevel  
if isPossiblePopup:  
    curChildOtherList = curSoup.find_all(  
        "XCUIElementTypeOther",  
        attrs={"type": "XCUIElementTypeOther", "enabled": "true"},  
        recursive=False,  
    )  
    if curChildOtherList:  
        curChildOtherNum = len(curChildOtherList)  
        if curChildOtherNum == 1:  
            popupTopFrameElement = curChildOtherList[0]  
  
return popupTopFrameElement
```

iOSProcessPopupPhotoCamera


```
        "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, photoC
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_sibl:
        if nextSiblingList:
            nextOtherSoup = None
            for eachNextSibling in nextSiblingList:
                if hasattr(eachNextSibling, "attrs"):
                    curType = eachNextSibling.attrs["t
                    if curType == "XCUIElementTypeOther":
                        # next sibling's first XCUIEl
                        nextOtherSoup = eachNextSibling
                        break

            if nextOtherSoup:
                cancelSoup = nextOtherSoup.find(
                    "XCUIElementTypeButton",
                    attrs={"visible": "true", "enabled": "tr
                )
                if cancelSoup:
                    self.clickElementCenterPosition(cancelS
                    foundAndProcessedPopup = True

return foundAndProcessedPopup
```

iOSProcessAlertNoteConfrim

```
def iOSProcessAlertNoteConfirm(self, soup):
    """只要符合XCUIElementTypeAlert 且其中有 确定 按钮，则点击"""
    foundAndProcessedPopup = False
    .....
    康爱公社 弹框 提醒 您的身份信息不完整 确定：
    <XCUIElementTypeAlert type="XCUIElementTypeAlert">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeOther type="XCUIElementTypeStaticText">
                                    <XCUIElementTypeStaticText>身份信息不完整</XCUIElementTypeStaticText>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeAlert>

    康爱公社 弹框 提示 请登录后再进行操作：
    <XCUIElementTypeAlert type="XCUIElementTypeAlert">
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeOther">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeOther type="XCUIElementTypeOther">
                        <XCUIElementTypeOther type="XCUIElementTypeOther">
                            <XCUIElementTypeOther type="XCUIElementTypeOther">
                                <XCUIElementTypeOther type="XCUIElementTypeStaticText">
                                    <XCUIElementTypeStaticText>请登录后再进行操作</XCUIElementTypeStaticText>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeAlert>
```

iOSProcessAlertCancel


```
{
    "tag": "XCUIElementTypeAlert",
    "attrs": {"enabled":"true", "visible":"true"}
},
{
    "tag": "XCUIElementTypeOther",
    "attrs": {"enabled":"true", "visible":"true"}
},
{
    "tag": "XCUIElementTypeButton",
    "attrs": {"enabled":"true", "visible":"true", "label": "取消", "type": "button", "name": "cancel", "value": "cancel", "color": "#000000", "textColor": "#FFFFFF", "fontStyle": "bold", "fontSize": 17, "fontFamily": "PingFangSC-Medium", "width": 100, "height": 44, "x": 150, "y": 150}
},
]
alertCancelSoup = CommonUtils.bsChainFind(soup, alertCancel)
if alertCancelSoup:
    self.clickElementCenterPosition(alertCancelSoup)
    foundAndProcessedPopup = True
return foundAndProcessedPopup
```

iOSProcessKagsPopupStillNotLogin

```

def iOSProcessKagsPopupStillNotLogin(self, soup):
    foundAndProcessedPopup = False
    .....
    康爱公社 弹框 您还未登录:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        ...
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="您还未登录" />
        </XCUIElementTypeOther>
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="康爱公社" />
        </XCUIElementTypeOther>
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="登录" />
        </XCUIElementTypeOther>
        <XCUIElementTypeOther type="XCUIElementTypeOther">
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="取消" />
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    .....
    stillNotLoginChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "x": "0", "width": "100%"},
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible": "true", "enabled": "true"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible": "true", "enabled": "true", "color": "#000000", "fontStyle": "normal", "fontWeight": "bold", "text": "您还未登录", "size": "14", "textColor": "#000000", "textType": "copyable", "type": "Text", "value": "您还未登录"}, "text": "您还未登录"
        },
    ]
    stillNotLoginSoup = CommonUtils.bsChainFind(soup, stillNotLoginChainList)
    if stillNotLoginSoup:
        parentOtherSoup = stillNotLoginSoup.parent
        parentParentOtherSoup = parentOtherSoup.parent
        if parentParentOtherSoup:
            tempNotLoginSoup = parentParentOtherSoup.find(
                "XCUIElementTypeStaticText",
                attrs={"visible": "true", "enabled": "true", "text": "您还未登录"})
            if tempNotLoginSoup:
                self.clickElementCenterPosition(tempNotLoginSoup)
                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="XCUIElementTypeWebView"
        <XCUIElementTypeOther type="XCUIElementTypeOther"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeOther type="XCUIElementTypeOther"
                    <XCUIElementTypeOther type="XCUIElementTypeOther"
                        <XCUIElementTypeOther type="XCUIElementTypeOther"
                            <XCUIElementTypeOther type="XCUIElementTypeStaticText"
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCUIElementTypeStaticText"
                                <XCUIElementTypeStaticText>
                                </XCUIElementTypeStaticText>
                            </XCUIElementTypeOther>
                            <XCUIElementTypeOther type="XCUIElementTypeButton"
                                <XCUIElementTypeOther type="XCUIElementTypeStaticText"
                                    <XCUIElementTypeStaticText>
                                    </XCUIElementTypeStaticText>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWebView>
    .....
    willOpenChainList = [
        {
            "tag": "XCUIElementTypeWebView",
            "attrs": {"visible": "true", "enabled": "true", "name": "非有赞官方网页, 请确认是否继续访问"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible": "true", "enabled": "true", "name": "非有赞官方网页, 请确认是否继续访问"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible": "true", "enabled": "true", "name": "非有赞官方网页, 请确认是否继续访问"}
        },
    ]
    willOpenSoup = CommonUtils.bsChainFind(soup, willOpenChainList)
    if willOpenSoup:
        parentOtherSoup = willOpenSoup.parent
        parentParentOtherSoup = parentOtherSoup.parent
        if parentParentOtherSoup:

```

```
continueOpenSoup = parentParentOtherSoup.find(  
    "XCUIElementTypeButton",  
    attrs={"visible":"true", "enabled":"true",  
}  
if continueOpenSoup:  
    self.clickElementCenterPosition(continueOpenSoup)  
    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="XCUIElementTypeOther">
        <XCUIElementTypeNavigationBar type="XCUIElementTypeOther">
            <XCUIElementTypeButton type="XCUIElementTypeButton">
                <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    </XCUIElementTypeNavigationBar>
                <XCUIElementTypeOther type="XCUIElementTypeTable">
                    <XCUIElementTypeOther type="XCUIElementTypeCell">
                        <XCUIElementTypeOther type="XCUIElementTypeTable">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                <XCUIElementTypeCell type="XCUIElementTypeCell">
                                    <XCUIElementTypeOther type="XCUIElementTypeCell">
                                        <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                            <XCUIElementTypeCell type="XCUIElementTypeCell">
                                                <XCUIElementTypeSwitch type="XCUIElementTypeSwitch">
                                                    </XCUIElementTypeCell>
                                                <XCUIElementTypeOther type="XCUIElementTypeCell">
                                                    </XCUIElementTypeTable>
                                                </XCUIElementTypeCell>
                                            </XCUIElementTypeTable>
                                        </XCUIElementTypeCell>
                                    </XCUIElementTypeCell>
                                </XCUIElementTypeCell>
                            </XCUIElementTypeTable>
                        </XCUIElementTypeCell>
                    </XCUIElementTypeTable>
                </XCUIElementTypeCell>
            </XCUIElementTypeNavigationBar>
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>
    .....
    foundAndProcessedPopup = False
    useLocationSwitchQuery = {"type": "XCUIElementTypeSwitch"}
    parentCellClassChain = "/XCUIElementTypeCell[`enabled`"
    useLocationSwitchQuery["parent_class_chains"] = [ parentCellClassChain ]
    foundAndClickeUseLocation = self.findAndClickElement(useLocationSwitchQuery)
    logging.debug("foundAndClickeUseLocation=%s", foundAndClickeUseLocation)
    if foundAndClickeUseLocation:
        parentNavBarClassChain = '/XCUIElementTypeNavigationBar[`enabled`'
        backButtonQuery = {"type": "XCUIElementTypeButton"} 
        backButtonQuery["parent_class_chains"] = parentNavBarClassChain
        foundAndClickeBack = self.findAndClickElement(backButtonQuery)
        logging.debug("foundAndClickeBack=%s", foundAndClickeBack)
    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="XCUIElementTypeOther">
            <XCUIElementTypeOther type="XCUIElementTypeTable">
                <XCUIElementTypeButton type="XCUIElementTypeTextFiled">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                        <XCUIElementTypeButton type="XCUIElementTypeText">
                            </XCUIElementTypeText>
                        </XCUIElementTypeTable>
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                        <XCUIElementTypeCell type="XCUIElementTypeImage">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                <XCUIElementTypeImage type="XCUIElementTypeImage">
                                    </XCUIElementTypeImage>
                                </XCUIElementTypeText>
                            </XCUIElementTypeImage>
                        </XCUIElementTypeCell>
                    </XCUIElementTypeImage>
                </XCUIElementTypeTable>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    .....
    networkNotStableChainList = [
        {
            "tag": "XCUIElementTypeTable",
            "attrs": {"enabled": "true", "visible": "true", "text": "京东金融 网络不稳定 请点击重试:"}
        },
        {
            "tag": "XCUIElementTypeCell",
            "attrs": {"enabled": "true", "visible": "true", "text": "京东金融 网络不稳定 请点击重试:"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"enabled": "true", "visible": "true", "text": "京东金融 网络不稳定 请点击重试:"}
        },
    ]
    networkNotStableSoup = CommonUtils.bsChainFind(soup, networkNotStableChainList)
    if networkNotStableSoup:
        self.clickElementCenterPosition(networkNotStableSoup)
        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="XCUIElementTypeOther">
            <XCUIElementTypeTable type="XCUIElementTypeTable">
                <XCUIElementTypeOther type="XCUIElementTypeOther">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeTable>
        </XCUIElementTypeOther>
    .....
    tryRefreshChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "name": "刷新按钮", "type": "button", "label": "刷新"}, "type": "button"
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled": "true", "visible": "true", "name": "刷新按钮", "type": "button", "label": "刷新"}, "type": "button"
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"enabled": "true", "visible": "true", "name": "刷新按钮", "type": "button", "label": "刷新"}, "type": "button"
        },
    ]
    tryRefreshSoup = CommonUtils.bsChainFind(soup, tryRefreshChainList)
    if tryRefreshSoup:
        self.clickElementCenterPosition(tryRefreshSoup)
        foundAndProcessedPopup = True

        # # Special: above click by position not work
        # # so change to wda query element then click
        # tryRefreshButtonQuery = {"type": "XCUIElementTypeButton", "label": "刷新", "name": "刷新按钮", "type": "button", "value": "刷新", "value_type": "text", "x": 500, "y": 500, "x2": 550, "y2": 550}
        # foundAndClicked = findAndClickElement(curSession, tryRefreshButtonQuery)
        # foundAndProcessedPopup = foundAndClicked

    return foundAndProcessedPopup
```

iOSProcessPopupCertificateError

```

def iOSProcessPopupCertificateError(self, soup):
    """Process iOS popup 安全证书存在问题 网络出错, 轻触屏幕重新加载"""
    foundAndProcessed = False

    .....
    <XCUIElementTypeButton type="XCUIElementTypeButton">
        <XCUIElementTypeStaticText type="XCUIElementTypeText">
            </XCUIElementTypeStaticText>
    </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="XCUIElementTypeButton">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                </XCUIElementTypeStaticText>
        </XCUIElementTypeButton>

        <XCUIElementTypeWindow type="XCUIElementTypeWindow">
            <XCUIElementTypeOther type="XCUIElementTypeWindow">
                <XCUIElementTypeOther type="XCUIElementTypeWindow">
                    <XCUIElementTypeOther type="XCUIElementTypeWindow">
                        <XCUIElementTypeButton type="XCUIElementTypeButton">
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                </XCUIElementTypeStaticText>
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                </XCUIElementTypeStaticText>
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                </XCUIElementTypeStaticText>
                            <XCUIElementTypeStaticText type="XCUIElementTypeText">
                                </XCUIElementTypeStaticText>
                        </XCUIElementTypeButton>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeWindow>
            </XCUIElementTypeOther>
            .....

            securityWarningChainList = self.generateCommonPopups()
            securityWarningSoup = CommonUtils.bsChainFind(soup,
            if securityWarningSoup:
                parentButtonSoup = securityWarningSoup.parent
                if parentButtonSoup:
                    certificateProblemP = re.compile("该网站的安全证书存在问题")
                    foundCertProblem = parentButtonSoup.find(
                        "XCUIElementTypeStaticText",
                        attrs={"visible": "true", "enabled": "true"}
                    )
                    if foundCertProblem:

```

```
        foundCancel = parentButtonSoup.find(  
            "XCUIElementTypeStaticText",  
            attrs={"visible":"true", "enabled":  
        })  
        if foundCancel:  
            self.clickElementCenterPosition(foundCancel)  
            foundAndProcessed = True  
  
    return foundAndProcessed
```

iOSProcessPopupWeixinLogin

```
def iOSProcessPopupWeixinLogin(self, soup):
    """Process iOS popup 微信登录"""
    foundAndProcessedPopup = False

    """
    <XCUIElementTypeWindow type="XCUIElementTypeWindow"
        <XCUIElementTypeOther type="XCUIElementTypeOther"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeOther type="XCUIElementTypeOther"
                    <XCUIElementTypeOther type="XCUIElementTypeTable"
                        <XCUIElementTypeCell type="XCUIElementTypeCell"
                            <XCUIElementTypeImage type="XCUIElementTypeImage"
                                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"
                                    <XCUIElementTypeOther type="XCUIElementTypeOther"/>
                                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
                                <XCUIElementTypeOther type="XCUIElementTypeOther"/>
                            </XCUIElementTypeCell>
                        </XCUIElementTypeTable>
                    <XCUIElementTypeOther type="XCUIElementTypeOther"/>
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
                    </XCUIElementTypeButton>
                    <XCUIElementTypeOther type="XCUIElementTypeOther"/>
                    <XCUIElementTypeButton type="XCUIElementTypeButton">
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
                    </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeOther>
    </XCUIElementTypeWindow>
    """

    weixinLoginChainList = self.generateCommonPopupItemChainList(
        secondLevelTag="XCUIElementTypeOther",
        thirdLevelValue="微信登录"
    )
    foundWeixinLogin = CommonUtils.bsChainFind(soup, weixinLoginChainList)

    tableChainList = self.generateCommonPopupItemChainList(
        secondLevelTag="XCUIElementTypeOther",
        thirdLevelTag="XCUIElementTypeTable",
    )
    foundTable = CommonUtils.bsChainFind(soup, tableChainList)

    if foundWeixinLogin and foundTable:
```

```
parentOtherSoup = foundWeixinLogin.parent
if parentOtherSoup:
    foundAllowButton = parentOtherSoup.find(
        "XCUIElementTypeButton",
        attrs={"visible":"true", "enabled":"true",
    })
    if foundAllowButton:
        self.clickElementCenterPosition(foundAllowButton)
        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="XCUIElementTypeOther"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeOther type="XCUIElementTypeOther"
                    <XCUIElementTypeButton type="XCUIElementTypeButton"
                        <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"
                            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"
                                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"
                                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"
                                        </XCUIElementTypeButton>
                                    </XCUIElementTypeOther>
                                </XCUIElementTypeOther>
                            </XCUIElementTypeOther>
                        </XCUIElementTypeButton>
                    </XCUIElementTypeOther>
                </XCUIElementTypeOther>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>
    .....
    leaveWeixinChainList = self.generateCommonPopupItemChain()
    foundLeaveWeixin = CommonUtils.bsChainFind(soup, leaveWeixinChainList)
    if foundLeaveWeixin:
        parentButtonSoup = foundLeaveWeixin.parent
        if parentButtonSoup:
            foundCancel = parentButtonSoup.find(
                "XCUIElementTypeStaticText",
                attrs={"visible": "true", "enabled": "true",
            )
            if foundCancel:
                self.clickElementCenterPosition(foundCancel)
                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="XCUIElementType
                <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="XCUIElement
                    <XCUIElementTypeOther type="XCUIEl
                        <XCUIElementTypeImage type="XCI
                        <XCUIElementTypeStaticText type=
                        <XCUIElementTypeStaticText type=
                        <XCUIElementTypeButton type="XC
                    </XCUIElementTypeOther>
                </XCUIElementTypeButton>
            </XCUIElementTypeOther>
        </XCUIElementTypeWindow>

    小程序:
        获取信息失败:
        <XCUIElementTypeButton type="XCUIElementTypeBu
            <XCUIElementTypeOther type="XCUIElementType
                <XCUIElementTypeImage type="XCUIEl
                <XCUIElementTypeStaticText type="XCUIE
                <XCUIElementTypeTextView type="XCUIEl
                <XCUIElementTypeButton type="XCUIEl
                <XCUIElementTypeButton type="XCUIEl
            </XCUIElementTypeOther>
        </XCUIElementTypeButton>
    .....
    # getInfoFailChainList = self.generateCommonPopupItemC

```

```
# getInfoFailSoup = CommonUtils.bsChainFind(soup, getInfoFailSoup)
# if getInfoFailSoup:
#     parentButtonOrOtherSoup = getInfoFailSoup.parent
doSomeFailP = re.compile("\S+失败$") # 获取信息失败, 微信弹框
doSomeFailChainList = self.generateCommonPopupItemChain()
doSomeFailSoup = CommonUtils.bsChainFind(soup, doSomeFailChainList)
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": "true",
            )

        if foundConfirm:
            self.clickElementCenterPosition(foundConfirm)
            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="XCUIElementTypeButton"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeImage type="XCUIElementTypeImage"/>
                <XCUIElementTypeStaticText type="XCUIElementTypeText"/>
                <XCUIElementTypeText View type="XCUIElementTypeText"/>
                <XCUIElementTypeButton type="XCUIElementTypeButton"/>
                <XCUIElementTypeButton type="XCUIElementTypeButton"/>
            </XCUIElementTypeOther>
        </XCUIElementTypeButton>
    .....
    authorizeLocationChainList = [
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"visible": "true", "enabled": "true", "name": "同意"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible": "true", "enabled": "true", "name": "取消"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible": "true", "enabled": "true", "name": "授权位置权限，以便更精准地定位附近的服务。同意后，将无法取消。"}
        },
    ]
    authorizeLocationSoup = CommonUtils.bsChainFind(soup, authorizeLocationChainList)
    if authorizeLocationSoup:
        parentOtherSoup = authorizeLocationSoup.parent
        confirmSoup = parentOtherSoup.find(
            "XCUIElementTypeButton",
            attrs={"visible": "true", "enabled": "true", "name": "同意"})
        if confirmSoup:
            self.clickElementCenterPosition(confirmSoup)
            foundAndProcessedPopup = True
    return foundAndProcessedPopup

```

iOSProcessPopupMiniprogramWarning

```

def iOSProcessPopupMiniprogramWarning(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="XCUIElementTypeButton"
            <XCUIElementTypeOther type="XCUIElementTypeOther"
                <XCUIElementTypeImage type="XCUIElementTypeImage"/>
                <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
                <XCUIElementTypeText View type="XCUIElementTypeText"/>
                <XCUIElementTypeButton type="XCUIElementTypeButton"/>
                <XCUIElementTypeButton type="XCUIElementTypeButton"/>
            </XCUIElementTypeOther>
        </XCUIElementTypeButton>
    """
    warningChainList = [
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"visible": "true", "enabled": "true", "name": "确定"}
        },
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"visible": "true", "enabled": "true", "name": "弹框背景遮罩层"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible": "true", "enabled": "true", "name": "授权说明文字"}
        }
    ]
    warningSoup = CommonUtils.bsChainFind(soup, warningChainList)
    if warningSoup:
        parentOtherSoup = warningSoup.parent
        confirmSoup = parentOtherSoup.find(
            "XCUIElementTypeButton",
            attrs={"visible": "true", "enabled": "true", "name": "确定"})
        if confirmSoup:
            self.clickElementCenterPosition(confirmSoup)
            foundAndProcessedPopup = True
    return foundAndProcessedPopup

```

iOSProcessPopupMiniprogramGetYour

```

# def iOSProcessPopupMiniprogramAuthority(self, soup):
def iOSProcessPopupMiniprogramGetYour(self, soup):
    """Process iOS popup 获取你的昵称、头像、地区及性别 / 获取你的好友列表
    foundAndProcessedPopup = False
    .....
    弹框 获取你的昵称、头像、地区及性别:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeImage">
        <XCUIElementTypeOther type="XCUIElementTypeText">
            <XCUIElementTypeImage type="XCUIElementTypeImage">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
            <XCUIElementTypeTable type="XCUIElementTypeTable">
                <XCUIElementTypeCell type="XCUIElementTypeCell">
                    <XCUIElementTypeOther type="XCUIElementTypeImage">
                    <XCUIElementTypeOther type="XCUIElementTypeImage">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                    <XCUIElementTypeOther type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeStaticText type="XCUIElementTypeText">
                    <XCUIElementTypeImage type="XCUIElementTypeImage">
                </XCUIElementTypeCell>
            </XCUIElementTypeTable>
            <XCUIElementTypeButton type="XCUIElementTypeText">
            <XCUIElementTypeButton type="XCUIElementTypeText">
            <XCUIElementTypeButton type="XCUIElementTypeText">
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

    弹框 获取你的通讯地址:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeImage">
        <XCUIElementTypeOther type="XCUIElementTypeText">
            <XCUIElementTypeImage type="XCUIElementTypeImage">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
            <XCUIElementTypeStaticText type="XCUIElementTypeText">
            <XCUIElementTypeButton type="XCUIElementTypeText">
            <XCUIElementTypeButton type="XCUIElementTypeText">
        </XCUIElementTypeOther>
    </XCUIElementTypeOther>

    小程序 弹框 需要获取你的地理位置:
    <XCUIElementTypeOther type="XCUIElementTypeOther">
        <XCUIElementTypeOther type="XCUIElementTypeImage">
        <XCUIElementTypeOther type="XCUIElementTypeText">
            <XCUIElementTypeImage type="XCUIElementTypeImage">
            <XCUIElementTypeOther type="XCUIElementTypeText">
            <XCUIElementTypeText type="XCUIElementTypeText">
            <XCUIElementTypeText type="XCUIElementTypeText">
            <XCUIElementTypeAlert type="XCUIElementTypeText">

```

```

<XCUIElementTypeOther type="XCUIElementTypeAlert">
    <XCUIElementTypeOther type="XCUIElementTypeText">
        <XCUIElementTypeOther type="XCUIElementTypeText">
            <XCUIElementTypeOther type="XCUIElementTypeText">
                <XCUIElementTypeOther type="XCUIElementTypeText">
                    <XCUIElementTypeOther type="XCUIElementTypeText">
                        <XCUIElementTypeOther type="XCUIElementTypeText">
                            <XCUIElementTypeText>...</XCUIElementTypeText>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeText>
            </XCUIElementTypeText>
        </XCUIElementTypeText>
    </XCUIElementTypeText>
</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+") # 获取你的通讯地址
getYourChainList = [
    {
        "tag": "XCUIElementTypeOther",
        "attrs": {"visible": "true", "enabled": "true", "text": "..."},
    },
    {
        "tag": "XCUIElementTypeOther",
        "attrs": {"visible": "true", "enabled": "true", "text": "..."}, ...
    }
]

```

```

        "attrs": {"visible": "true", "enabled": "true", },
    },
    {
        "tag": "XCUIElementTypeStaticText",
        "attrs": {"visible": "true", "enabled": "true", },
    },
]
getYourSoup = CommonUtils.bsChainFind(soup, getByCharName)
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=allowAttrs)

    if not foundAllow:
        # for support future possible case
        parentParentOther = parentOther.parent
        foundAllow = parentParentOther.find(allowTag, attrs=allowAttrs)

    if not foundAllow:
        # "贝德玛会员中心" 需要获取你的地理位置
        parentParentParentOther = parentParentOther.parent
        foundAllow = parentParentParentOther.find(allowTag, attrs=allowAttrs)

    if foundAllow:
        self.clickElementCenterPosition(foundAllow)
        foundAndProcessedPopup = True

return foundAndProcessedPopup

```

iOSProcessPopupMiniprogramDisclaimer

```

def iOSProcessPopupMiniprogramDisclaimer(self, soup):
    """Process iOS popup 免责声明"""
    foundAndProcessedPopup = False
    .....
    <XCUIElementTypeWindow type="XCUIElementTypeWindow"
        <XCUIElementTypeOther type="XCUIElementTypeOther"
            <XCUIElementTypeButton type="XCUIElementTypeButton"
                <XCUIElementTypeOther type="XCUIElementTypeOther"
                    <XCUIElementTypeImage type="XCUIElementTypeImage"/>
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
                    <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
                    <XCUIElementTypeButton type="XCUIElementTypeButton"/>
                </XCUIElementTypeOther>
            </XCUIElementTypeButton>
        </XCUIElementTypeOther>
    </XCUIElementTypeWindow>
    .....
    disclaimerChainList = [
        {
            "tag": "XCUIElementTypeWindow",
            "attrs": {"visible": "true", "enabled": "true", "type": "XCUIElementTypeWindow"}
        },
        {
            "tag": "XCUIElementTypeButton",
            "attrs": {"visible": "true", "enabled": "true", "type": "XCUIElementTypeButton"}
        },
        {
            "tag": "XCUIElementTypeStaticText",
            "attrs": {"visible": "true", "enabled": "true", "type": "XCUIElementTypeStaticText"}
        }
    ]
    disclaimerSoup = CommonUtils.bsChainFind(soup, disclaimerChainList)
    if disclaimerSoup:
        parentOtherSoup = disclaimerSoup.parent
        if parentOtherSoup:
            foundIKnow = parentOtherSoup.find(
                "XCUIElementTypeButton",
                attrs={"visible": "true", "enabled": "true", "type": "XCUIElementTypeButton"})
            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 = {"visible": "true", "enabled": "true", "width": "full", "height": "full"}
    commonItemChainList = [
        {
            "tag": firstLevelTag,
            "attrs": CommonAttrs_VisibleEnabledFullWidthFullHeight
        },
        {
            "tag": secondLevelTag,
            "attrs": CommonAttrs_VisibleEnabledFullWidthFullHeight
        },
    ]
    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, isShow=True):
    """for debug, draw rectangle 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, isShow=True):
    """for debug, to draw rectangle for each element in curList"""
    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], curWidth, curHeight]
        curTimeStr = CommonUtils.getCurDatetimeStr("%H%M%S")
        curSaveTail = "_rect_{0}_{1}{2}|{3}{4}|{5}{6}|{7}{8}|{9}{10}|{11}{12}|{13}{14}|{15}{16}|{17}{18}|{19}{20}|{21}{22}|{23}{24}|{25}{26}|{27}{28}|{29}{30}|{31}{32}|{33}{34}|{35}{36}|{37}{38}|{39}{40}|{41}{42}|{43}{44}|{45}{46}|{47}{48}|{49}{50}|{51}{52}|{53}{54}|{55}{56}|{57}{58}|{59}{59}|{60}{61}|{62}{63}|{64}{65}|{66}{67}|{68}{68}|{69}{69}|{70}{70}|{71}{71}|{72}{72}|{73}{73}|{74}{74}|{75}{75}|{76}{76}|{77}{77}|{78}{78}|{79}{79}|{80}{80}|{81}{81}|{82}{82}|{83}{83}|{84}{84}|{85}{85}|{86}{86}|{87}{87}|{88}{88}|{89}{89}|{90}{90}|{91}{91}|{92}{92}|{93}{93}|{94}{94}|{95}{95}|{96}{96}|{97}{97}|{98}{98}|{99}{99}|{100}{100}|{101}{101}|{102}{102}|{103}{103}|{104}{104}|{105}{105}|{106}{106}|{107}{107}|{108}{108}|{109}{109}|{110}{110}|{111}{111}|{112}{112}|{113}{113}|{114}{114}|{115}{115}|{116}{116}|{117}{117}|{118}{118}|{119}{119}|{120}{120}|{121}{121}|{122}{122}|{123}{123}|{124}{124}|{125}{125}|{126}{126}|{127}{127}|{128}{128}|{129}{129}|{130}{130}|{131}{131}|{132}{132}|{133}{133}|{134}{134}|{135}{135}|{136}{136}|{137}{137}|{138}{138}|{139}{139}|{140}{140}|{141}{141}|{142}{142}|{143}{143}|{144}{144}|{145}{145}|{146}{146}|{147}{147}|{148}{148}|{149}{149}|{150}{150}|{151}{151}|{152}{152}|{153}{153}|{154}{154}|{155}{155}|{156}{156}|{157}{157}|{158}{158}|{159}{159}|{160}{160}|{161}{161}|{162}{162}|{163}{163}|{164}{164}|{165}{165}|{166}{166}|{167}{167}|{168}{168}|{169}{169}|{170}{170}|{171}{171}|{172}{172}|{173}{173}|{174}{174}|{175}{175}|{176}{176}|{177}{177}|{178}{178}|{179}{179}|{180}{180}|{181}{181}|{182}{182}|{183}{183}|{184}{184}|{185}{185}|{186}{186}|{187}{187}|{188}{188}|{189}{189}|{190}{190}|{191}{191}|{192}{192}|{193}{193}|{194}{194}|{195}{195}|{196}{196}|{197}{197}|{198}{198}|{199}{199}|{200}{200}|{201}{201}|{202}{202}|{203}{203}|{204}{204}|{205}{205}|{206}{206}|{207}{207}|{208}{208}|{209}{209}|{210}{210}|{211}{211}|{212}{212}|{213}{213}|{214}{214}|{215}{215}|{216}{216}|{217}{217}|{218}{218}|{219}{219}|{220}{220}|{221}{221}|{222}{222}|{223}{223}|{224}{224}|{225}{225}|{226}{226}|{227}{227}|{228}{228}|{229}{229}|{230}{230}|{231}{231}|{232}{232}|{233}{233}|{234}{234}|{235}{235}|{236}{236}|{237}{237}|{238}{238}|{239}{239}|{240}{240}|{241}{241}|{242}{242}|{243}{243}|{244}{244}|{245}{245}|{246}{246}|{247}{247}|{248}{248}|{249}{249}|{250}{250}|{251}{251}|{252}{252}|{253}{253}|{254}{254}|{255}{255}|{256}{256}|{257}{257}|{258}{258}|{259}{259}|{260}{260}|{261}{261}|{262}{262}|{263}{263}|{264}{264}|{265}{265}|{266}{266}|{267}{267}|{268}{268}|{269}{269}|{270}{270}|{271}{271}|{272}{272}|{273}{273}|{274}{274}|{275}{275}|{276}{276}|{277}{277}|{278}{278}|{279}{279}|{280}{280}|{281}{281}|{282}{282}|{283}{283}|{284}{284}|{285}{285}|{286}{286}|{287}{287}|{288}{288}|{289}{289}|{290}{290}|{291}{291}|{292}{292}|{293}{293}|{294}{294}|{295}{295}|{296}{296}|{297}{297}|{298}{298}|{299}{299}|{300}{300}|{301}{301}|{302}{302}|{303}{303}|{304}{304}|{305}{305}|{306}{306}|{307}{307}|{308}{308}|{309}{309}|{310}{310}|{311}{311}|{312}{312}|{313}{313}|{314}{314}|{315}{315}|{316}{316}|{317}{317}|{318}{318}|{319}{319}|{320}{320}|{321}{321}|{322}{322}|{323}{323}|{324}{324}|{325}{325}|{326}{326}|{327}{327}|{328}{328}|{329}{329}|{330}{330}|{331}{331}|{332}{332}|{333}{333}|{334}{334}|{335}{335}|{336}{336}|{337}{337}|{338}{338}|{339}{339}|{340}{340}|{341}{341}|{342}{342}|{343}{343}|{344}{344}|{345}{345}|{346}{346}|{347}{347}|{348}{348}|{349}{349}|{350}{350}|{351}{351}|{352}{352}|{353}{353}|{354}{354}|{355}{355}|{356}{356}|{357}{357}|{358}{358}|{359}{359}|{360}{360}|{361}{361}|{362}{362}|{363}{363}|{364}{364}|{365}{365}|{366}{366}|{367}{367}|{368}{368}|{369}{369}|{370}{370}|{371}{371}|{372}{372}|{373}{373}|{374}{374}|{375}{375}|{376}{376}|{377}{377}|{378}{378}|{379}{379}|{380}{380}|{381}{381}|{382}{382}|{383}{383}|{384}{384}|{385}{385}|{386}{386}|{387}{387}|{388}{388}|{389}{389}|{390}{390}|{391}{391}|{392}{392}|{393}{393}|{394}{394}|{395}{395}|{396}{396}|{397}{397}|{398}{398}|{399}{399}|{400}{400}|{401}{401}|{402}{402}|{403}{403}|{404}{404}|{405}{405}|{406}{406}|{407}{407}|{408}{408}|{409}{409}|{410}{410}|{411}{411}|{412}{412}|{413}{413}|{414}{414}|{415}{415}|{416}{416}|{417}{417}|{418}{418}|{419}{419}|{420}{420}|{421}{421}|{422}{422}|{423}{423}|{424}{424}|{425}{425}|{426}{426}|{427}{427}|{428}{428}|{429}{429}|{430}{430}|{431}{431}|{432}{432}|{433}{433}|{434}{434}|{435}{435}|{436}{436}|{437}{437}|{438}{438}|{439}{439}|{440}{440}|{441}{441}|{442}{442}|{443}{443}|{444}{444}|{445}{445}|{446}{446}|{447}{447}|{448}{448}|{449}{449}|{450}{450}|{451}{451}|{452}{452}|{453}{453}|{454}{454}|{455}{455}|{456}{456}|{457}{457}|{458}{458}|{459}{459}|{460}{460}|{461}{461}|{462}{462}|{463}{463}|{464}{464}|{465}{465}|{466}{466}|{467}{467}|{468}{468}|{469}{469}|{470}{470}|{471}{471}|{472}{472}|{473}{473}|{474}{474}|{475}{475}|{476}{476}|{477}{477}|{478}{478}|{479}{479}|{480}{480}|{481}{481}|{482}{482}|{483}{483}|{484}{484}|{485}{485}|{486}{486}|{487}{487}|{488}{488}|{489}{489}|{490}{490}|{491}{491}|{492}{492}|{493}{493}|{494}{494}|{495}{495}|{496}{496}|{497}{497}|{498}{498}|{499}{499}|{500}{500}|{501}{501}|{502}{502}|{503}{503}|{504}{504}|{505}{505}|{506}{506}|{507}{507}|{508}{508}|{509}{509}|{510}{510}|{511}{511}|{512}{512}|{513}{513}|{514}{514}|{515}{515}|{516}{516}|{517}{517}|{518}{518}|{519}{519}|{520}{520}|{521}{521}|{522}{522}|{523}{523}|{524}{524}|{525}{525}|{526}{526}|{527}{527}|{528}{528}|{529}{529}|{530}{530}|{531}{531}|{532}{532}|{533}{533}|{534}{534}|{535}{535}|{536}{536}|{537}{537}|{538}{538}|{539}{539}|{540}{540}|{541}{541}|{542}{542}|{543}{543}|{544}{544}|{545}{545}|{546}{546}|{547}{547}|{548}{548}|{549}{549}|{550}{550}|{551}{551}|{552}{552}|{553}{553}|{554}{554}|{555}{555}|{556}{556}|{557}{557}|{558}{558}|{559}{559}|{560}{560}|{561}{561}|{562}{562}|{563}{563}|{564}{564}|{565}{565}|{566}{566}|{567}{567}|{568}{568}|{569}{569}|{570}{570}|{571}{571}|{572}{572}|{573}{573}|{574}{574}|{575}{575}|{576}{576}|{577}{577}|{578}{578}|{579}{579}|{580}{580}|{581}{581}|{582}{582}|{583}{583}|{584}{584}|{585}{585}|{586}{586}|{587}{587}|{588}{588}|{589}{589}|{590}{590}|{591}{591}|{592}{592}|{593}{593}|{594}{594}|{595}{595}|{596}{596}|{597}{597}|{598}{598}|{599}{599}|{600}{600}|{601}{601}|{602}{602}|{603}{603}|{604}{604}|{605}{605}|{606}{606}|{607}{607}|{608}{608}|{609}{609}|{610}{610}|{611}{611}|{612}{612}|{613}{613}|{614}{614}|{615}{615}|{616}{616}|{617}{617}|{618}{618}|{619}{619}|{620}{620}|{621}{621}|{622}{622}|{623}{623}|{624}{624}|{625}{625}|{626}{626}|{627}{627}|{628}{628}|{629}{629}|{630}{630}|{631}{631}|{632}{632}|{633}{633}|{634}{634}|{635}{635}|{636}{636}|{637}{637}|{638}{638}|{639}{639}|{640}{640}|{641}{641}|{642}{642}|{643}{643}|{644}{644}|{645}{645}|{646}{646}|{647}{647}|{648}{648}|{649}{649}|{650}{650}|{651}{651}|{652}{652}|{653}{653}|{654}{654}|{655}{655}|{656}{656}|{657}{657}|{658}{658}|{659}{659}|{660}{660}|{661}{661}|{662}{662}|{663}{663}|{664}{664}|{665}{665}|{666}{666}|{667}{667}|{668}{668}|{669}{669}|{670}{670}|{671}{671}|{672}{672}|{673}{673}|{674}{674}|{675}{675}|{676}{676}|{677}{677}|{678}{678}|{679}{679}|{680}{680}|{681}{681}|{682}{682}|{683}{683}|{684}{684}|{685}{685}|{686}{686}|{687}{687}|{688}{688}|{689}{689}|{690}{690}|{691}{691}|{692}{692}|{693}{693}|{694}{694}|{695}{695}|{696}{696}|{697}{697}|{698}{698}|{699}{699}|{700}{700}|{701}{701}|{702}{702}|{703}{703}|{704}{704}|{705}{705}|{706}{706}|{707}{707}|{708}{708}|{709}{709}|{710}{710}|{711}{711}|{712}{712}|{713}{713}|{714}{714}|{715}{715}|{716}{716}|{717}{717}|{718}{718}|{719}{719}|{720}{720}|{721}{721}|{722}{722}|{723}{723}|{724}{724}|{725}{725}|{726}{726}|{727}{727}|{728}{728}|{729}{729}|{730}{730}|{731}{731}|{732}{732}|{733}{733}|{734}{734}|{735}{735}|{736}{736}|{737}{737}|{738}{738}|{739}{739}|{740}{740}|{741}{741}|{742}{742}|{743}{743}|{744}{744}|{745}{745}|{746}{746}|{747}{747}|{748}{748}|{749}{749}|{750}{750}|{751}{751}|{752}{752}|{753}{753}|{754}{754}|{755}{755}|{756}{756}|{757}{757}|{758}{758}|{759}{759}|{760}{760}|{761}{761}|{762}{762}|{763}{763}|{764}{764}|{765}{765}|{766}{766}|{767}{767}|{768}{768}|{769}{769}|{770}{770}|{771}{771}|{772}{772}|{773}{773}|{774}{774}|{775}{775}|{776}{776}|{777}{777}|{778}{778}|{779}{779}|{780}{780}|{781}{781}|{782}{782}|{783}{783}|{784}{784}|{785}{785}|{786}{786}|{787}{787}|{788}{788}|{789}{789}|{790}{790}|{791}{791}|{792}{792}|{793}{793}|{794}{794}|{795}{795}|{796}{796}|{797}{797}|{798}{798}|{799}{799}|{800}{800}|{801}{801}|{802}{802}|{803}{803}|{804}{804}|{805}{805}|{806}{806}|{807}{807}|{808}{808}|{809}{809}|{810}{810}|{811}{811}|{812}{812}|{813}{813}|{814}{814}|{815}{815}|{816}{816}|{817}{817}|{818}{818}|{819}{819}|{820}{820}|{821}{821}|{822}{822}|{823}{823}|{824}{824}|{825}{825}|{826}{826}|{827}{827}|{828}{828}|{829}{829}|{830}{830}|{831}{831}|{832}{832}|{833}{833}|{834}{834}|{835}{835}|{836}{836}|{837}{837}|{838}{838}|{839}{839}|{840}{840}|{841}{841}|{842}{842}|{843}{843}|{844}{844}|{845}{845}|{846}{846}|{847}{847}|{848}{848}|{849}{849}|{850}{850}|{851}{851}|{852}{852}|{853}{853}|{854}{854}|{855}{855}|{856}{856}|{857}{857}|{858}{858}|{859}{859}|{860}{860}|{861}{861}|{862}{862}|{863}{863}|{864}{864}|{865}{865}|{866}{866}|{867}{867}|{868}{868}|{869}{869}|{870}{870}|{871}{871}|{872}{872}|{873}{873}|{874}{874}|{875}{875}|{876}{876}|{877}{877}|{878}{878}|{879}{879}|{880}{880}|{881}{881}|{882}{882}|{883}{883}|{884}{884}|{885}{885}|{886}{886}|{887}{887}|{888}{888}|{889}{889}|{890}{890}|{891}{891}|{892}{892}|{893}{893}|{894}{894}|{895}{895}|{896}{896}|{897}{897}|{898}{898}|{899}{899}|{900}{900}|{901}{901}|{902}{902}|{903}{903}|{904}{904}|{905}{905}|{906}{906}|{907}{907}|{908}{908}|{909}{909}|{910}{910}|{911}{911}|{912}{912}|{913}{913}|{914}{914}|{915}{915}|{916}{916}|{917}{917}|{918}{918}|{919}{919}|{920}{920}|{921}{921}|{922}{922}|{923}{923}|{924}{924}|{925}{925}|{926}{926}|{927}{927}|{928}{928}|{929}{929}|{930}{930}|{931}{931}|{932}{932}|{933}{933}|{934}{934}|{935}{935}|{936}{936}|{937}{937}|{938}{938}|{939}{939}|{940}{940}|{941}{941}|{942}{942}|{943}{943}|{944}{944}|{945}{945}|{946}{946}|{947}{947}|{948}{948}|{949}{949}|{950}{950}|{951}{951}|{952}{952}|{953}{953}|{954}{954}|{955}{955}|{956}{956}|{957}{957}|{958}{958}|{959}{959}|{960}{960}|{961}{961}|{962}{962}|{963}{963}|{964}{964}|{965}{965}|{966}{966}|{967}{967}|{968}{968}|{969}{969}|{970}{970}|{971}{971}|{972}{972}|{973}{973}|{974}{974}|{975}{975}|{976}{976}|{977}{977}|{978}{978}|{979}{979}|{980}{980}|{981}{981}|{982}{982}|{983}{983}|{984}{984}|{985}{985}|{986}{986}|{987}{987}|{988}{988}|{989}{989}|{990}{990}|{991}{991}|{992}{992}|{993}{993}|{994}{994}|{995}{995}|{996}{996}|{997}{997}|{998}{998}|{999}{999}|{1000}{1000}|{1001}{1001}|{1002}{1002}|{1003}{1003}|{1004}{1004}|{1005}{1005}|{1006}{1006}|{1007}{1007}|{1008}{1008}|{1009}{1009}|{1010}{1010}|{1011}{1011}|{1012}{1012}|{1013}{1013}|{1014}{1014}|{1015}{1015}|{1016}{1016}|{1017}{1017}|{1018}{1018}|{1019}{1019}|{1020}{1020}|{1021}{1021}|{1022}{1022}|{1023}{1023}|{1024}{1024}|{1025}{1025}|{1026}{1026}|{1027}{1027}|{1028}{1028}|{1029}{1029}|{1030}{1030}|{1031}{1031}|{1032}{1032}|{1033}{1033}|{1034}{1034}|{1035}{1035}|{1036}{1036}|{1037}{1037}|{1038}{1038}|{1039}{1039}|{1040}{1040}|{1041}{1041}|{1042}{1042}|{1043}{1043}|{1044}{1044}|{1045}{1045}|{1046}{1046}|{1047}{1047}|{1048}{1048}|{1049}{1049}|{1050}{1050}|{1051}{1051}|{1052}{1052}|{1053}{1053}|{1054}{1054}|{1055}{1055}|{1056}{1056}|{1057}{1057}|{1058}{1058}|{1059}{1059}|{1060}{1060}|{1061}{1061}|{1062}{1062}|{1063}{1063}|{1064}{1064}|{1065}{1065}|{1066}{1066}|{1067}{1067}|{1068}{1068}|{1069}{1069}|{1070}{1070}|{1071}{1071}|{1072}{1072}|{1073}{1073}|{1074}{1074}|{1075}{1075}|{1076}{1076}|{1077}{1077}|{1078}{1078}|{1079}{1079}|{1080}{1080}|{1081}{1081}|{1082}{1082}|{1083}{1083}|{1084}{1084}|{1085}{1085}|{1086}{1086}|{1087}{1087}|{1088}{1088}|{1089}{1089}|{1090}{1090}|{1091}{1091}|{1092}{1092}|{1093}{1093}|{1094}{1094}|{1095}{1095}|{1096}{1096}|{1097}{1097}|{1098}{1098}|{1099}{1099}|{1100}{1100}|{1101}{1101}|{1102}{1102}|{1103}{1103}|{1104}{1104}|{1105}{1105}|{1106}{1106}|{1107}{1107}|{1108}{1108}|{1109}{1109}|{1110}{1110}|{1111}{1111}|{1112}{1112}|{1113}{1113}|{1114}{1114}|{1115}{1115}|{1116}{1116}|{1117}{1117}|{1118}{1118}|{1119}{1119}|{1120}{1120}|{1121}{1121}|{1122}{1122}|{1123}{1123}|{1124}{1124}|{1125}{1125}|{1126}{1126}|{1127}{1127}|{1128}{1128}|{1129}{1129}|{1130}{1130}|{1131}{1131}|{1132}{1132}|{1133}{1133}|{1134}{1134}|{1135}{1135}|{1136}{1136}|{1137}{1137}|{1138}{1138}|{1139}{1139}|{1140}{1140}|{1141}{1141}|{1142}{1142}|{1143}{1143}|{1144}{1144}|{1145}{1145}|{1146}{1146}|{1147}{1147}|{1148}{1148}|{1149}{1149}|{1150}{1150}|{1151}{1151}|{1152}{1152}|{1153}{1153}|{1154}{1154}|{1155}{1155}|{1156}{1156}|{1157}{1157}|{1158}{1158}|{1159}{1159}|{1160}{1160}|{1161}{1161}|{1162}{1162}|{1163}{1163}|{1164}{1164}|{1165}{1165}|{1166}{1166}|{1167}{1167}|{1168}{1168}|{1169}{1169}|{1170}{1170}|{1171}{1171}|{1172}{1172}|{1173}{1173}|{1174}{1174}|{1175}{1175}|{1176}{1176}|{1177}{1177}|{1178}{1178}|{1179}{1179}|{1180}{1180}|{1181}{1181}|{1182}{1182}|{1183}{1183}|{1184}{1184}|{1185}{1185}|{1186}{1186}|{1187}{1187}|{1188}{1188}|{1189}{1189}|{1190}{1190}|{1191}{1191}|{1192}{1192}|{1193}{1193}|{1194}{1194}|{1195}{1195}|{1196}{1196}|{1197}{1197}|{1198}{1198}|{1199}{1199}|{1200}{1200}|{1201}{1201}|{1202}{1202}|{1203}{1203}|{1204}{1204}|{1205}{1205}|{1206}{1206}|{1207}{1207}|{1208}{1208}|{1209}{
```

```
return
```

调用：

```
# # for debug
# self.debugDrawScreenRect(curRect=bounds)
```

和：

```
Elements = self.get_elements_valuable(Elements)
# for debug
# self.debugDrawElementRect(Elements, isShowEach=True, isSave=False)
self.debugDrawElementRect(Elements, isShowEach=False, isSave=True)

Elements = [element for element in Elements if self.is_element(element)]
# for debug
# self.debugDrawElementRect(Elements, isShowEach=True, isSave=False)
self.debugDrawElementRect(Elements, isShowEach=False, isSave=True)
```

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 session
    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=scaledSize)
    return scaledFile
```

调用举例：

```

curScale = 3
if isResizeToOriginal:
    # and resize to original screen size
    savedImgFile = self.scaleToOrginSize(savedImgFile, cur

```

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, imageFormat)
        screenFilename = "%s.%s" % (curDatetimeStr, imageFormat)
        if filePrefix:
            screenFilename = "%s_%s" % (filePrefix, screenFilename)
            # 'com.netease.cloudmusic_20200221_170305.jpg'
        screenFilename = os.path.join(saveFolder, screenFilename)

        if imageFormat == "png":
            curPillowObj = self.wdaClient.screenshot(png_filename)
            savedScreenFile = screenFilename
        elif (imageFormat == "jpg") or (imageFormat == "jpeg"):
            curPillowObj = self.wdaClient.screenshot()
            # logging.debug("curPillowObj=%s", curPillowObj)
            # curPillowObj=<PIL.PngImagePlugin.PngImageFile image
            # convert to PIL.Image and then save as jpg
            curPillowObj.save(screenFilename)

            savedScreenFile = screenFilename
        else:
            logging.debug("Unsupported image format: %s", imageFormat)

        if savedScreenFile:
            logging.debug("saved screenshot: %s", savedScreenFile)
            return savedScreenFile

```

调用举例 + 相关函数：

```
def debugiOSSaveScreenshot(self, saveFolder, isResizeToOriginal=False):
    """for debug, save iOS screenshot"""
    # # for if enable debug screenshot image content to original screen size
    # # so disable debug screenshot
    # needEnableLater = False
    # if wda.DEBUG:
    #     wda.DEBUG = False
    #     needEnableLater = True
    # savedImgFile = self.saveiOSScreenshot(filePrefix="ios")
    # if needEnableLater:
    #     wda.DEBUG = True
    def _saveScreenshot():
        return self.saveiOSScreenshot(filePrefix="ios", saveFolder=saveFolder)
    savedImgFile = self.runButNotOutputWdaDebug(_saveScreenshot)

    logging.debug("Saved iOS screenshot %s", savedImgFile)
    if isResizeToOriginal:
        # and resize to original screen size
        savedImgFile = self.scaleToOrginSize(savedImgFile)
    return savedImgFile
```

调用：

```
elif self.isiOS:
    fullImgFilePath = self.debugiOSSaveScreenshot('ios')
```

以及：

```
def runButNotOutputWdaDebug(self, funcToRun):
    # disable then re-enable debug for
    # avoid debug print too many binary image data of
    # 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
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2020-06-21 11:07:45

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:@selector(handleURL:)],
        [[FBRoute POST:@"/session"].withoutSession respondWithTarget:self action:@selector(handleSession:)],
        [[FBRoute POST:@"/wda/apps/launch"] respondWithTarget:self action:@selector(handleLaunch:)],
        [[FBRoute POST:@"/wda/apps/activate"] respondWithTarget:self action:@selector(handleActivate:)],
        [[FBRoute POST:@"/wda/apps/terminate"] respondWithTarget:self action:@selector(handleTerminate:)],
        [[FBRoute POST:@"/wda/apps/state"] respondWithTarget:self action:@selector(handleState:)],
        [[FBRoute GET:@"/wda/apps/list"] respondWithTarget:self action:@selector(handleList:)],
        [[FBRoute GET:@""].respondWithTarget:self action:@selector(handleRoot:)],
        [[FBRoute DELETE:@""].respondWithTarget:self action:@selector(handleDelete:)],
        [[FBRoute GET:@"/status"].withoutSession respondWithTarget:self action:@selector(handleStatus:)]
    };

    // Health check might modify simulator state so it should be handled by session
    [[FBRoute GET:@"/wda/healthcheck"].withoutSession respondWithTarget:self action:@selector(handleHealthcheck:)]

    // Settings endpoints
    [[FBRoute GET:@"/appium/settings"] respondWithTarget:self action:@selector(handleSettingsGet:)],
    [[FBRoute POST:@"/appium/settings"] respondWithTarget:self action:@selector(handleSettingsPost:)]
};
}
```

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 arguments converted to
        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/:name

```

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/FBFIndE
lementCommands.m

```

+ (NSArray *)routes
{
    return @{
        [[FBRoute POST:@"/element"] respondWithTarget:self act:^{
            [[FBRoute POST:@"/elements"] respondWithTarget:self act:^{
                [[FBRoute POST:@"/element/:uuid/element"] respondWithTarget:self act:^{
                    [[FBRoute POST:@"/element/:uuid/elements"] respondWithTarget:self act:^{
                        ...
                }
            }
        }
    }
}

```

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": "通讯录"}'
Return (984ms): {
    "value": [
        {
            "ELEMENT": "15000000-0000-0000-0502-000000000000",
            "element-6066-11e4-a52e-4f735466cecf": "15000000-0000-0000-0502-000000000000"
        }
    ],
    "sessionId": "C69F63F6-80EA-47DD-BA04-0A912945CE39"
}
...
Shell: curl -X GET -d '' 'http://192.168.31.43:8100/session/C69F63F6-80EA-47DD-BA04-0A912945CE39'
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。

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2020-06-21 22:23:11

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-06-01 18:17:10

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.mobletimer
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.appleMaps
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

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook 最后更新: 2020-06-21 10:37:38

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](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 22:26:27

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NSXPConnection的
`_XCT_fetchSnapshotForElement:attributes:parameters:reply`错误
- 【规避解决】WebDriverAgent获取页面源码报错：`xpc.exceptions NSXPConnection 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`
-
- [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](#)
-

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-06-21 22:45:56