目录

移动端自动化测试概览

- 最新版本： `v1.0`
- 更新时间： `202010701`

简介

总结安卓和iOS等移动端自动化测试开发心得，包括常见框架Appium、uiautomator2、facebook-wda、AirTest等。以及一些常见问题的总结。

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/mobile_automation_overview: 移动端自动化测试概览

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

- 移动端自动化测试概览 book.crifan.com
- 移动端自动化测试概览 crifan.github.io

离线下载阅读

- 移动端自动化测试概览 PDF
- 移动端自动化测试概览 ePub
- 移动端自动化测试概览 Mobi

版权说明

鸣谢

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

更多其他电子书

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

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

移动端自动化概述

此处针对移动端自动化测试进行简单概要的介绍：

- 移动端自动化测试概览
  - **移动端**：主要指的是**安卓**和**iOS**设备
  - **自动化测试**
    - 根据用途和场景分
      - **自动化测试**
        - 典型用途：测试移动端的app的功能是否满足预期
      - **自动化操作**
        - 典型用途：模拟人的手指去操作屏幕，点击元素等操作，以便于自动化一套操作流程
  - **概览**
    - 介绍总体概况
      - 有哪些主流的库

自动化测试 vs 自动化操作

移动端的自动化领域，根据用途和场景可以分为2类：

- 自动化测试
  - 又称
    - **移动端测试**
  - 侧重于：测试（移动端，主要指手机中）app的功能是否有问题
    - 比如
      - app是否会崩溃
      - 功能是否符合预期
        - 往往涉及到断言assertion，期望特定的输出
          - 举例
            - 输入非法手机号，点击注册
              - 希望：弹框提示 非法手机号
- 自动化操作
  - 又称：
    - **自动化抓包**
  - 侧重于：模拟人的手去操作手机
    - 更多关注的是：
      - 页面上有哪些元素
      - 以及如何处理到这些元素
        - 比如
          - 模拟人手去点击
            - 解放双手，写自动化脚本，实现自己的功能

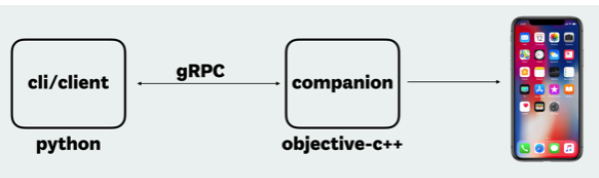
        - 举例
            - 自动化操作：每天定时收取支付宝蚂蚁深林中的能量
    - 提取元素中的内容
        - 保存出来
            - 就属于 抓包，保存特定数据 的方面了
                - 所以也可以叫做：自动化抓包
                    - 举例
                        - 自动化抓包：天猫app中的 热卖商品信息的爬取和保存

说明：

- 此文主要侧重于介绍：**自动化操作 = 自动化抓包**
- 不论是 自动化操作 还是 自动化测试 其使用的底层框架都是一样的
    - 比如facebook-wda用于iOS的自动化操作和测试

常见框架

移动端自动化测试常见框架：

- 多平台支持
  - Appium
    - 概述：一个非常流行的、支持多种终端类型（Windows、Mac、Linux、Android、iOS）的自动化测试框架
    - 详见：后续章节 Appium
  - Airtest
    - 主页
      - GitHub
        - AirtestProject/Airtest: UI Automation Framework for Games and Apps
    - 支持平台
      - Android
      - Emulator
      - iOS
      - Windows
      - Unity
      - Cocos2dx
      - Egret
      - WeChat
- 单个平台
  - `Android`
    - `uiautomator2` = `u2`
  - `iOS`
    - `facebook-wda`
    - `idb` = `iOS Development Bridge`
      - 主页
        - GitHub
          - facebook/idb: idb is a flexible command line interface for automating iOS simulators and devices
        - 官网
          - idb · iOS Development Bridge
      - Facebook新出的
      - 架构
        - 

        - 缺点：
            - 需要改动被测app的代码才能自动化测试？
                - 想要测试（iOS模拟器或真机）设备，要在被测设备中安装xctest测试用例才可以

u2和facebook-wda都是ATX拆分出来的

最早是：

NetEaseGame/ATX: Smart phone automation tool. Support iOS, Android, WebApp and game

后来拆分成：

- `Android` 的 `uiautomator2`
- `iOS` 的 `facebook-wda`

iOS自动化测试框架发展历史

- iOS底层测试框架
    - iOS `8.0 ~ 9.3` ：UIAutomation
        - 缺点：只能调试单台设备
            - 原因： `instruments` 限制单台Mac只能对应单台iOS设备
    - iOS `9.3+` ：XCUITest
        - 目的：用以替代旧的 `UIAutomation`
- 第三方
    - `WebDriverAgent`
        - 作者： `Facebook`
        - 核心原理：实现了 `WebDriver` 的server
            - 通过 server 可以远程控制 iOS 设备
                - 支持各种操作：启动应用、关闭应用、点击、滚动等
                - 通过连接 XCTest.framework 调用苹果的 API 执行动作
        - 优点
            - 能够支持单台 `Mac` 对应多个iOS设备
                - 支持多个设备同时进行自动化
            - `Appium` 、 `Macaca` 已经集成

Appium

- 概述
  - Appium是最主流的自动化测试框架，支持自动化操作 `iOS` 手机、 `Android` 手机和 `Windows` 系统中的 `原生` 、 `移动Web` 和 `混合` 的应用。
    - `原生应用` ： 用 `iOS` 、 `Android` 或 `Windows SDKs` 编写的应用
    - `移动Web应用` ： 用移动端浏览器访问的应用
      - `iOS` 上的 `Safari` 、 `Chrome`
      - `Android` 上的内置浏览器
    - `混合应用` ： 带有一个 `Webview` 的包装器，用来和Web内容交互的原生控件
- 详解
  - 独立教程：主流跨平台自动化框架：Appium

uiautomator2

详见另一完整教程：

安卓自动化测试利器：uiautomator2

facebook-wda

详见另一完整教程：

iOS自动化测试利器：facebook-wda

AirTest

另外还有一个，自动化测试工具：网易的 `AirTest`

- `AirTest`
  - 一句话描述：
    - 网易游戏推出的一款跨平台的UI自动化测试框架，适用于游戏和App
  - 官网
    - 主页
      - Airtest Project
    - 文档
      - 欢迎使用 - Airtest Project Docs
      - 欢迎来到Airtest官方文档！ — airtest 文档

常见问题

移动端自动化测试会遇到的一些问题，现整理如下。

比如 Same test cases for Android and iOS automation — pros and cons | by Satyajit Malugu | Medium，其中就提到了：

- 返回按钮的处理等问题
- 界面的源码结构不同
  - iOS：会返回当前页面中所有的元素，包括不可见的（需要滚动后才可见的那些）元素
  - Android：只会当前页面中可见的元素

导致实现移动端的多平台统一测试用例，很不容易

OCR识别复杂游戏界面中文字，偶尔会误判出错

比如之前折腾：

【已解决】安卓游戏暗黑觉醒自动化：稳定的检测出是首充豪礼首充6元首充98元的首充弹框

期间，对于 首充豪礼 弹框页面：

偶尔可以返回：

```
{'chars': [{'char': '首', 'location': {'width': 36, 'top':
```

完美解析的效果，即：`首充6元首充98元`，是图片中准确的文字和充值金额。

由于游戏界面中文字比较复杂，尤其是：

- 特殊的字体
  - 首充豪礼 字体很特别
    - 估计是游戏常用字体
- 额外加了闪光等效果
  - 比如 首充98元
    - 中的 首充 或 98元 外圈闪光

等特殊情况，导致文字检测出来，常常会误判：

比如：

（1）首元首充8元

```
{'chars': [{'char': '首', 'location': {'width': 36, 'top':
```

（2）首元首8元

```
{'chars': [{'char': '首', 'location': {'width': 36, 'top':
```

（3）首充元+首充98

```
{'chars': [{'char': '首', 'location': {'width': 37, 'top':
```

（4）首充5元充98元

```
{'chars': [{'char': '首', 'location': {'width': 58, 'top':
```

（5）首兄元首9元

```
{'chars': [{'char': '首', 'location': {'width': 36, 'top':
```

（6）首元98元

```
{'chars': [{'char': '首', 'location': {'width': 36, 'top':
```

（7）首元首元

```
{'chars': [{'char': '首', 'location': {'width': 34, 'top':
```

（8）首元道充98元

```
{'chars': [{'char': '首', 'location': {'width': 35, 'top':
```

（9）首充元首充98元

```
{'chars': [{'char': '首', 'location': {'width': 61, 'top':
```

（10）首6元首充8元

```
{'chars': [{'char': '首', 'location': {'width': 41, 'top':
```

等等情况。

所以如果用如下逻辑（正则）

```
"首充((\d+元)|(元)|(\d+))", # 首充元 / 首充xxx元 / 首充xxx
```

的代码

```
    def isGotoPayPopupPage(self, isRespLocation=False):
        """Check is goto payment popup page or not"""
        gotoPayStrList = [
            "^前往充值$", # 剑玲珑
            "^立即充值$", # 至尊屠龙
            "^充值$", # 剑玲珑，首充之后，手动点击 每日充值 后
            "^充点小钱$", # 御剑仙缘
            # "^首充豪礼?", # 暗黑觉醒：首充豪礼 但有时候无法识别出
            # 暗黑觉醒：首充6元 和 首充98元 ->
            # "首充\d+元", #（1）识别成 首元首充8元
            # "首充?\d+元", #  （2）首元首8元
            "首充((\d+元)|(元)|(\d+))", # 首充元 / 首充xxx元 /


            # 剑玲珑，首充 之后，但是点击 领取 并不能进入下一页
            # "^领取$",
            # "^领取奖励$", # 偶尔会由于屏幕弹框 领域奖励 而误判进
        ]
        respBoolOrTuple = self.isExistAnyStr(gotoPayStrList
        logging.info("GotoPay: respBoolOrTuple=%s", respBoo
        return respBoolOrTuple
```

注：`isExistAnyStr`的具体实现，可参考：图像 · Python常用代码段

去单次调用，往往会误判，而无法识别出我们希望的值

所以，为了稳定的检测出是否是首充豪礼的弹框充值野蛮，后来改用逻辑：

多次尝试调用，直到检测成功为止

具体代码是：

```
    def isGotoPayPopupPage_multipleRetry(self, isRespLocat:
        """鉴于 暗黑觉醒 的 首充弹框 的检测很不稳定，所以 增加此函数
        respBoolOrTuple = CommonUtils.multipleRetry(
            functionInfoDict = {
                "functionCallback": self.isGotoPayPopupPage
                "functionParaDict": {
                    "isRespLocation": isRespLocation,
                },
            },
            isRespFullRetValue = isRespLocation,
        )
        return respBoolOrTuple
```

注：其中`multipleRetry`详见：通用逻辑 · Python常用代码段

附录

下面列出相关参考资料。

参考资料

- iOS自动化测试利器：facebook-wda
- 安卓自动化测试利器：uiautomator2
-
- XCUITest
- UIAutomation
- UiAutomator / UiAutomator2
- WinAppDriver
-