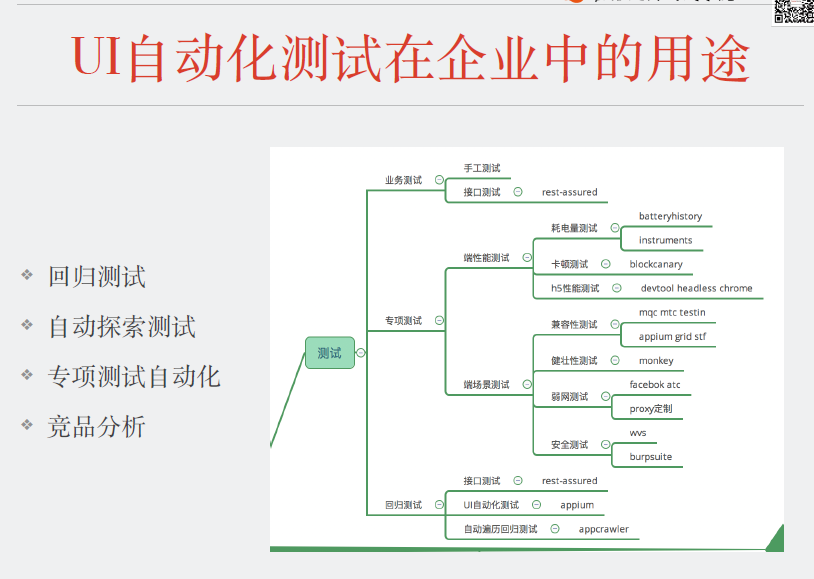
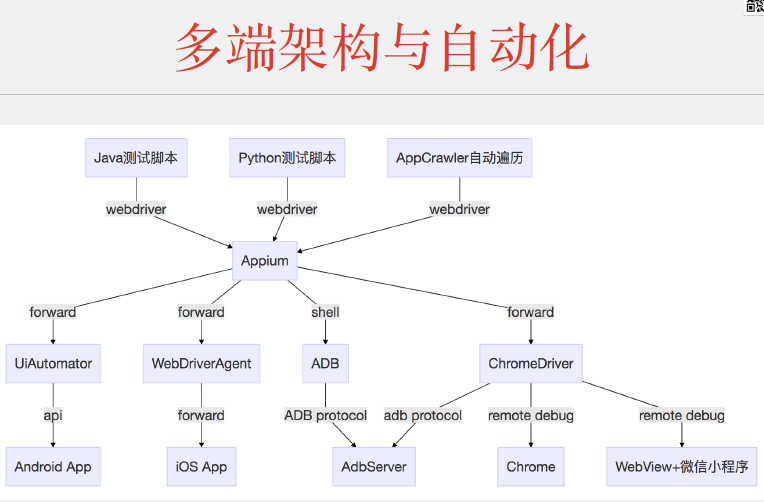
**课程表**

|  |  |
| --- | --- |
| 移动测试自动化Appium  10月13日  10月20日  10月27日  11月3日  11月10日  11月17日 | 1、Appium环境安装与架构介绍 2、Appium用例录制与用例流程介绍  3、元素定位与常见自动化动作 4、高级定位与断言机制  5、Android自动化用例编写 6、Android WebView测试与Toast识别  7、Android自动化测试实战 8、iOS测试用例编写  9、iOS真机测试与WebView测试 10、iOS测试实战  11、Appium错误排查与日志分析 12、PageObject设计模式  13、参数化与数据驱动实战 14、微信公众号与小程序测试实战  15、App自动化测试实战 16、自动遍历测试方法与AppCrawler基本使用  17、自动遍历测试进阶与实战演练 18、Appium Grid模式  19、多设备管理平台STF 20、兼容性测试实战 |
| 接口测试实战  11月24日  12月1日  12月8日  12月15日 | 1、接口测试体系与常见协议解析 2、代理工具抓包和篡改结果  3、接口测试框架选型与入门 4、Restful接口测试入门  5、Restful接口测试进阶与断言 6、通用协议加解密与schema断言  7、参数化与数据驱动 8、接口测试与持续集成结合  9、Web hook机制与可视化报告 10、接口测试实战  11、dubbo测试入门 12、dubbo测试进阶与应用  13、接口管理工具swagger入门 14、接口管理工具swagger实战  15、接口测试答疑课 |
| 移动专项测试  12月22日  12月29日 | 1、App性能测试 2、App流量分析 3、弱网测试  4、内存泄漏检测 5、卡顿检测 6、崩溃检测  7、耗电量分析 8、App反编译分析 9、App行为篡改 |
| Web测试自动化Selenium  1月5日  1月12日  1月19日  1月26日 | 1、Selenium简介与环境搭建 2、Selenium IDE用例录制与常见API讲解  3、元素定位与常见操作命令 4、Selenium的断言机制与等待机制  5、表单切换与窗口处理（窗口切换、设置窗口大小、关闭窗口）  6、Selenium多浏览器处理（Chrome、Firfox、IE、Safari）  7、文件上传与弹框处理 8、Cookie机制  9、测试报告生成与定制 10、Selenium基本操作实战演练  11、参数化与数据驱动实战（读取csv、Excel）12、PageObject设计模式  13、Selenium grid浏览器集群管理 14、Selenium项目实战演练  15、Selenium整体答疑 |
| 持续集成Jenkins  2月16日  2月23日  3月2日 | 1、Jenkins安装与入门介绍 2、Jenkins Job机制  3、Web Hook机制与高级插件 4、Selenium自动化测试持续集成实战演练  5、Appium自动化测试持续集成实战演练 6、接口测试持续集成实战演练  7、Pipeline入门 8、Jenkins file语法  9、Blue Ocean使用 10、Jenkins实战演练 |
| 容器技术应用Docker | 1、docker介绍与入门 2、docker基础命令  3、使用docker搭建selenium、jenkins、sonar等  4、使用docker搭建android模拟器与appium  5、docker镜像制作入门 6、docker镜像制作应用 7、k8s简介 |
| |  | | --- | | 代码审计平台Sonar | | 1、Sonar平台安装与基础使用 2、Sonar在单元测试与覆盖率上的应用  3、Sonar api使用 4、Sonar自定义代码扫描规则 5、Sonar与jenkins集成 |
| 质量监控平台建设ELK | 1、ELK系统搭建与基础使用 2、LogStash数据导入 3、ElasticSearch基本使用 4、Kibana高级报表制作 5、质量数据收集与监控面板制作 |

**UI自动化测试在企业中的用途**



**多端架构与自动化**



**夜神模拟器**

1. SDK的adb需与夜神的nox\_adb版本一致(如不一致则将SDK中的adb复制到模拟器bin目录下，将该名称改为nox\_adb.exe).
2. 启动模拟器，cmd至夜神bin目录，使用命令“nox\_adb.exe connect 127.0.0.1:62001”. 成功连接了模拟器之后，用"adb devices -l"命令查看设备
3. appium-desktop🡪 "deviceName": "127.0.0.1:62001"
4. 开发者模式: 设置->关于平板电脑->点击5次”版本号”

**获取app的信息**

app信息

* 获取当前界面元素：adb shell dumpsys activity top
* 获取任务列表：adb shell dumpsys activity activities

app入口

* adb (–s 192.168.59.102:5555) logcat |grep -i displayed
* aapt dump badging mobike.apk | grep launchable-activity
* apkanalyzer 最新版本的sdk中才有

启动应用

adb shell am start -W -n com.xueqiu.android/.view.WelcomeActivityAlias –S

**capabilities设置**

* app apk地址
* appPackage 包名
* appActivity Activity名字
* automationName 默认使⽤uiautomator
* noReset fullReset 是否在测试前后重置相关环境
* unicodeKeyBoard resetKeyBoard 是否输⼊非英文,测试完成后重置输⼊法
* udid 手机唯一标识

**测试步骤三要素：定位、交互、断⾔**

定位

* id定位 对应id
* accessibility id定位 对应content-desc (盲人专用)
* xpath定位
* 绝对定位(全路径)  
  /hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGroup/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.RelativeLayout/android.widget.LinearLayout/android.widget.HorizontalScrollView/android.widget.LinearLayout/android.widget.RelativeLayout[2]/android.widget.RelativeLayout/android.widget.TextView
* 相对定位(半路径)

//android.widget.TextView[@instance=2] 用属性定位

//android.widget.TextView[@text='股票'][1]

用属性,位置定位(xpath索引从1开始,属性值为单引号)(text是股票的第1个元素)

(//android.widget.TextView[@text='股票'])[1] 全局,第1个是text是股票的TestView   
//\*[@text='股票' and @resource-id='com.xueqiu.android:id/text']

未知节点,并用多个属性定位  
//\*[contains(@resource-id, 'text') and @text='股票'] 用包含,and定位

//\*[contains(@text, ‘登录’) or contains(@label, ‘登录’)]] 用包含,or定位

//android.widget.ListView[@resource-id='com.xueqiu.android:id/listview']/

android.widget.LinearLayout[last()] 取最后一个LinearLayout元素

//\*[contains(@text, '看点')]/ancestor::\*//\*[contains(name(), ‘EditText’)] 取先辈节点

//\*[@clickable="true"]/android.widget.TextView[string-length(@text)>0 and

string-length(@text)<20] 用属性长度定位

* Android UIAutomator Selector

new UiSelector().text("自选") #查找text为””自选”的元素

new UiSelector().className("android.widget.TextView").instance(1)

#查找第一个TextView

new UiScrollable(new UiSelector().scrollable(true).instance(0)).getChildByText(new UiSelector().className("android.widget.TextView"), "食品饮料") #查找滚动元素

交互:

动作: click, sendKeys, sendKeyEvent

self.driver.find\_element\_by\_id("agree").click()

手势: press, release, moveTo, tap wait, longPress, cancel, perform

滑屏: TouchAction(driver).press(x=300, y=2400).move\_to(x=300, y=300).release(). perform()

模拟点击: TouchAction(driver).press(500,700).perform()

其他常用API:

截图: driver.get\_screenshot\_as\_file("login.png")

隐式等待: driver.implicitly\_wait(5)

获得xml布局: print(driver.page\_source)

系统操作: lock, background, hideKeyBoard, openNotifications, shake, startActivity, currentActivity, getCurrentPackage

APP操作: installApp, removeApp, isInstalled, closeApp, launchApp, reset, getAppStrings, getContextHandles getContextContext

https://github.com/appium/appium/blob/master/docs/en/writing-running-appium/other/

appium-bindings.md

**安装卸载Appium**

安装: npm install -g appium

或 淘宝cnpm:

npm install -g cnpm --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/)

cnpm install -g appium (--verbose)

启动: appium –p 4723

卸载: npm remove –g appium

查询已发布版本: npm view appium versions --json

指定安装版本: npm install [appium@1.8.1](mailto:appium@1.8.1)

**调试分析方法**

* Appium Log分析

appium –g /tmp/appium.log

grep “Running” /tmp/appium.log | less

grep “Running” /tmp/appium.log | grep –o “192.168.59.101\\\:5555.\*” | awk ‘{print “adb” $0}’ | sed –e “s#’\$##g”

[BaseDriver] Session created with session id: 743b31e0-e8ff-421a-b38c-174a94fab34 #创建session，用于跟客户端通信

[AndroidDriver] Java version is: 1.8.0\_65 #获取java版本

[ADB] Using adb.exe from C:\Users\zhang\AppData\Local\Android\Sdk\platform-tools\adb.exe

#通过环境变量ANDROID\_HOME获取adb位置

[AndroidDriver] Using device: db23bebd #通过”adb devices”命令,查找当前连接的设备及udid

[ADB] Running 'adb.exe -P 5037 -s db23bebd shell getprop ro.build.version.sdk' #获得设备的安卓系统版本

adb.exe shell pm list packages com.xueqiu.android #列出所有安装包com.xueqiu.android

adb.exe wait-for-device

adb.exe shell echo ping #检查设备是否能够正常响应adb命令

adb.exe shell pm list packages io.appium.settings #列出所有io.appium.settings安装包

adb.exe shell dumpsys package io.appium.settings #获取安装包信息

[ADB] Using aapt.exe from C:\Users\zhang\AppData\Local\Android\Sdk\build-tools\28.0.3\ aapt.exe #检查aapt是否可用

adb.exe shell ps #查看手机里正在运行的进程

adb.exe shell appops set io.appium.settings android\:mock\_location allow #允许io.appium.settings创建模拟位置

[AndroidDriver] Got language: 'undefined' and country: 'undefined' #获取设备的语言，国家

adb.exe install 'C:\\Program Files (x86)\\Appium\\resources\\app\\node\_modules\\appium \\node\_modules \\appium-android-ime\\bin\\UnicodeIME-debug.apk'' #安装支持中文输入的包

adb.exe shell settings get secure default\_input\_method #获取系统默认输入法

adb.exe shell pm list packages io.appium.unlock #列出io.appium.unlock，辅助设备解锁

adb.exe shell dumpsys package io.appium.unlock #查看io.appium.unlock信息

adb.exe shell getprop ro.build.version.release #获取设备系统版本

adb.exe shell wm size #获取设备分辨率

'adb.exe shell getprop ro.product.model #获取设备型号

adb.exe shell getprop ro.product.manufacturer #获取设备厂商

adb.exe shell am force-stop com.xueqiu.android #强制停止应用

adb.exe shell pm clear com.xueqiu.android #清除数据和缓存

adb.exe forward tcp\:4724 tcp\:4724 #建立Appium Server到设备上的4724端口转发

adb.exe push 'C:\\Program Files (x86)\\Appium\\resources\\app\\node\_modules\\appium\\ node\_modules\\appium-android-driver\\bootstrap\\bin\\AppiumBootstrap.jar' /data/local/tmp/' #开启设备上appiumbootstrap服务

[ADB] Creating ADB subprocess with args: ["-P",5037,"-s","db23bebd","shell","uiautomator", "runtest", "AppiumBootstrap.jar","-c","io.appium.android.bootstrap.Bootstrap","-e","pkg", "com.xueqiu.android","-e","disableAndroidWatchers",false,"-e","acceptSslCerts",false] #不监控ANR和Crash，只针对基于UIAutomator的测试有效，这将较少CPU消耗

adb.exe shell dumpsys window #获取当前活动窗口信息

adb.exe shell am start -W -n com.xueqiu.android/com.xueqiu.android.common.SplashActivity -S 启动雪球应用

[Appium] New AndroidDriver session created successfully, session 73fd45cb-761f-447b-8365-b05b62aaef88 added to master session list #创建新的AndroidDriver Session

[HTTP] --> POST /wd/hub/session/73fd45cb-761f-447b-8365-b05b62aaef88/context

[MJSONWP] Calling AppiumDriver.setContext() with args: ["NATIVE\_APP","73fd45cb-761f-447b-8365-b05b62aaef88"] #设置Appium Driver运行时的上下文

adb.exe shell cat /proc/net/unix #获取设备上Unix域套接字信息

获取当前的WebView

获取当前页面元素（appium服务端接收Get请求，解析参数并将参数通过4724端口发送给AndroidBootstrap，AndroidBootstrap发送命令到bootstrap，bootstrap完成操作后，结果再逐层返回）

其他：

adb shell input keyevent 3 #发送按键“3”给设备  
adb shell wm density #获取屏幕密度  
adb shell rm -rf /data/local/tmp/strings.json # 删除json文件  
adb shell pm path com.xueqiu.android #列出包名的所在位置  
adb shell am instrument -w io.appium.uiautomator2.server.test/android.support.test.runner.

AndroidJUnitRunner #运行所有用例  
https://testerhome.com/topics/12499

* 得到PageSource

界⾯的完整dom结构. xml⽂件: print(driver.page\_source)



* 调试脚本

driver.findElementsByXPath(“//\*") #取出所有元素,方便下一步过滤/获取信息

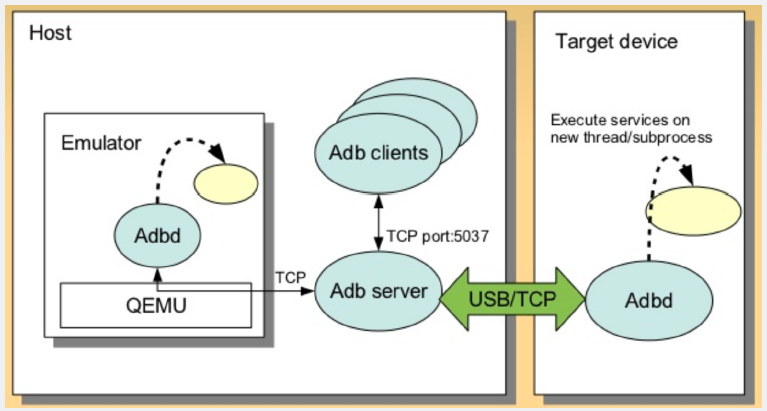
for x in self.driver.findElementsByXPath(“//\*")：

print(“text=”, x.text)

print(“tag\_name=”, x.tag\_name)

print(“location=”, x.location)

**Adb知识与排错**



**脚本优化**

* 将”定位”与”行为”隔离: findElementByXXX 更新为findElement(By.xxx())

self.driver.find\_element\_by\_id(“agree”).click() 更新为

self.button\_agree=(MobileBy.ID,”agree”)

self.driver.find\_element(self.button\_agree).click()

* MobileBy.ANDROID\_UIAUTOMATOR不生效，使用By.ANDROID\_UIAUTOMATOR

**Toast**

模拟器地址：https://github.com/appium/sample-code/raw/master/sample-code/

apps/ ApiDemos/bin/ApiDemos-debug.apk

* caps["automationName"]="uiautomator2"
* 必须使用xpath查找

driver.find\_element\_by\_xpath("//\*[@class='android.widget.Toast']")

driver.find\_element\_by\_xpath("//\*[contains(@text, "xxxxx")]")

**移动浏览器**

依赖chromedriver, 计算机需要安装与设备浏览器对应的chromedriver

查找设备的浏览器包: adb shell pm list packages |less

caps['browserName']=’Safari’/’Chrome’/’Chromium’/’Browser’

caps['chromedriverExecutable']='/Users/xxx/projects/chromedriver/2.20/chromedriver'

driver.get(“https://www.baidu.com”) #用自带浏览器访问百度

判断控件是否为android.webkit.WebView或子孙:用UiautomatorViewer查看classname

print(self.driver.contexts) #获得上下文  
print(self.driver.current\_context) #

self.driver.switch\_to.context("WEBVIEW\_com.xueqiu.android") #切进APP里的浏览器  
print(self.driver.page\_source) #获取APP里浏览器的显示内容  
self.driver.get("https://www.baidu.com") #在APP中访问百度

**adb模拟操作**

* [adb shell模拟点击事件(input tap)](https://www.cnblogs.com/yoyoketang/p/8983193.html)

命令行：adb shell input tap 200 500

代码：adb1=’adb shell input tap 200 500’

os.system(adb1)

* [adb shell按键操作(input keyevent)](https://www.cnblogs.com/yoyoketang/p/8988203.html)

adb shell input keyevent KEYCODE\_BACK 或 adb shell input keyevent 4

* [adb shell输入英文(input text)](https://www.cnblogs.com/yoyoketang/p/8988203.html)

adb shell input text AAA

* [adb shell输入中文(ADBKeyBoard)](https://www.cnblogs.com/yoyoketang/p/8998789.html) (手机先安装ADBKeyBoard.apk)

adb shell am broadcast -a ADB\_INPUT\_TEXT --es msg '上海'

**IOS**

* 概念

Certificates：苹果签发  
开发证书：开发者使用  
发布证书：对外分化到app store或者使用ad-hoc模式分发  
Provisioning Profiles：  
App ID：bundleID相关设置  
Devices List：可内测的真机列表  
分为开发和发布两种大类型  
发布又分为ad-hoc、app store、enterprise级别  
.app文件：编译后生成的项目打包代码  
.ipa文件：经过签名后的打包文件，本质是zip文件

* 常见命令

ls ~/Library/MobileDevice/Provisioning\ Profiles  
security find-identity -p codesigning

instruments -s devices 查看模拟器列表

idevice\_id –l 查看真机列表

ideviceinstaller -i demo.app 安装app

xcodebuild -scheme UICatalog -target iOS clean build  
xcodebuild -scheme UICatalog -target iOS archive

* WebDriverAgent

git clone https://github.com/facebook/WebDriverAgent.git  
cd WebDriverAgent/

bash -x ./Scripts/bootstrap.sh 可以查看输出细节

启动: open WebDriverAgent.xcodeproj and start WebDriverAgentRunner test.

或xcodebuild -project WebDriverAgent.xcodeproj \

-scheme WebDriverAgentRunner \

-destination 'platform=iOS Simulator,name=iPhone 6' test