# 使用Appuim进行ios-app功能自动化测试

## 1 appium的介绍

### 1.1 appium是怎样的框架？

Appium是一个移动端的自动化框架，可用于测试原生应用，移动网页应用和混合型应用，且是跨平台的。可用于IOS和Android以及firefox的操作系统。原生的应用是指用android或ios的sdk编写的应用，移动网页应用是指网页应用，类似于ios中safari应用或者Chrome应用或者类浏览器的应用。混合应用是指一种包裹webview的应用,原生应用于网页内容交互性的应用。重要的是Appium是跨平台的，何为跨平台，意思就是可以针对不同的平台用一套api来编写测试用例。

### 1.2 支持平台

- iOS
- Android
- FirefoxOS

### 1.3 为何要选择appium

- 不需要以任何方式重新编译或者修改app，就可以在所有的平台上使用标准的自动化APIs；
- 可以用你喜欢的开发工具使用任何 WebDriver 兼容的语言来编写测试用例.比如 Java, Objective-C, JavaScript with Node.js (in both callback and yield-based flavours), PHP, Python, Ruby, C#, Clojure, 或者 Perl 可以使用标准的Selenium WebDriver API和特定语言的客户端库.
- 可以使用任何测试框架.
- 依托 WebDriver 意味着你可以押宝在一个已经成为事实上标准的独立,自由和开放的协议.而不会被限制在任何的专利中
- 如果在没有使用Appium的情况,你使用了Apple的UIAutomation库就只能通过Javascript，并且只能通过Instruments application插桩应用来运行你的测试。同样的，在Google的UiAutomator体系下,你只能用Java写你的测试案例，而Appium最终开启了跨平台原生移动自动化的可能，这才是最重要的。

### 1.4 与其他测试工具对比

| | UiAutomator | Appium | Robotium | Espresso |
|---|---|---|---|---|
| 支持语言 | Java | Java、python、Ruby、PHP等 | Java | Java |
| 是否需要签名 | 不需要 | 不需要 | 不需要 | 不需要 |
| 是否支持跨应用测试 | 支持 | 不支持 | 不支持 | 不支持 |
| 是否跨平台 | 仅支持Android平台 | Android、IOS都支持 | 仅支持Android平台 | 仅支持Android平台 |
| 是否支持WebView | 不支持 | 支持Native App、Hybird App、Web App | 不支持 | 不支持 |

总结：Appium是最近比较热门的框架，社区也很活跃。这个框架应该是是功能最强大的。

1）开源；支持Native App、Hybird App、Web App；支持Android、iOS、Firefox OS；Server也是跨平台的，你可以使用Mac OS X、Windows或者Linux；

2）用Appium自动化测试不需要重新编译App；支持很多语言来编写测试脚本，Java、Javascript、PHP、Python、C#、Ruby等主流语言；不需要为了自动化测试来重造轮子，因为扩展了WebDriver。（WebDriver是测试WebApps的一种简单、快速的自动化测试框架，所以有Web自动化测试经验的测试人员可以直接上手）；

3）原理：使用的是Client/Server架构，运行的时候Server端会监听Client端发过来的命令，翻译这些命令发送给移动设备或模拟器，然后移动设备或模拟器做出响应的反应。正是因为这种架构，所以Client可以使用Appium client libraries多种语言的测试脚本，而且Server端完全可以部署在服务器上，甚至云服务器。

4）其他概念说明：

Session，表示每个Client连接到Server以后都会有一个Session ID，而且Client发送命令到Server端都需要这个Session ID，因为这个seesion id代表了你所打开的浏览器或者是移动设备的模拟器。所以你甚至可以打开N个Session，同时测试不同的设备或模拟器。

Desired Capabilities，其实就是一个键值对，设置一些测试的相关信息来告诉Server端，我们需要测试iOS、还是Android，或者是WebApp等信息。

Appium Server是Node.js写的，所以可以直接用NPM来进行安装。

Appium Clients，Mac OS和Win下提供GUI，不需要装Node.js，方便测试人员操作。

5）限制：

如果你在Windows使用Appium，你没法使用预编译专用于OS X的.app文件，因为Appium依赖OS X专用的库来支持iOS测试，所以在Windows平台你不能测试iOS Apps。这意味着你只能通过在Mac上来运行iOS测试。

## 2 appium环境搭建与用例编写、执行

在网上搜索appium都非常多资源和教程，但都不适用于我们目前的设备配置或者不适用于现在Appium的最新版本，再加上要根据ios版本，就少之有少了，所以我猜在这过程中肯定会遇到很多的坑，不出意料，这大半个月来，经历了各种各样大的、坑小的坑，深的、神的坑，也一度觉得要放弃不能做下去了 ，幸而咬紧牙关，终于爬了出来，所以说，不管如何，坑还是要入的，入了还是要爬的。

### 2.1 环境准备

系统： mac OS Sierra 版本：10.12.5；
开发工具：Xcode 版本 8.3.2
测试工具：appium 版本 1.6.5

截止2017-09-15，这些都是最新版本；

mac OS: Mac OS是一套运行于苹果Macintosh系列电脑上的操作系统。Mac OS是首个在商用领域成功的图形用户界面操作系统。现行的最新的系统版本是OS X 10.12 ，且网上也有在PC上运行的Mac系统，简称 Mac PC

Xcode： 是运行在操作系统Mac OS X上的集成开发工具（IDE），由苹果公司开发。Xcode是开发OS X 和 iOS 应用程序的最快捷的方式。Xcode 具有统一的用户界面设计，编码、测试、调试都在一个简单的窗口内完成。

因为之前我有一些linux的基础，再加上熟悉了几天mac系统，所以以下的环境搭建我都是通过打开终端工具，并用命令行来进行搭建的，首先我们先来搭建的是ios模拟器的环境，如果连ios模拟器的环境都成功不了，那么ios真机就不要进行了，原因不多说了，反正都是泪。

### 2.2 模拟器环境搭建 （mac OS10.12+Xcode8.3+ios模拟器10.3）

**1） 命令： ~hum$ java -version** （~hum是本机admin用户名字，可忽略，命令是$后面的内容）

mac一般都会自带java，所以java的安装和配置可以忽略，但我们要确认java的版本，最好是在1.7版本以后，不然后面肯定有深坑等着跳，所以输入此命令就是要确认java版本。如果低于1.7版本，就要卸载java并下载最新安装和配置，这步骤在网上有操作说明，在这里我就不重复了。

```
humdeMac-mini:~ hum$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
humdeMac-mini:~ hum$
```

**2） 命令： ~hum$ git --version**

同第1条，也是确认本机是否有安装git工具，Git是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任

何或小或大的项目，这个也是为了后面的安装和更新一些需要的项目，git没有版本要求，确认其已安装即可，如果没有安装，那可以通过homebrew来进行安装；

```
humdeMac-mini:~ hum$ git --version
git version 2.10.0
humdeMac-mini:~ hum$
```

**3）命令： ~hum$ curl -LsSf http://github.com/mxcl/homebrew/tarball/master | sudo tar xvz -C/usr/local --strip 1**

此命令就是安装homebrew，是mac上的软件包管理工具，能在mac中方便的安装并且卸载软件，比如第2步检查到没有安装git，可以使用命令 brew install git，如果要卸载软件，那可以使用命令 brew unistall git，使用命令brew -v 可以检查版本或者检查是否有安装，没有版本要求；

```
humdeMac-mini:~ hum$ brew -v
Homebrew 1.3.2
Homebrew/homebrew-core (git revision fedb8; last commit 2017-09-11)
humdeMac-mini:~ hum$
```

**4）命令：~hum$ brew node**

node是什么就不具体说，反正是安装npm的必须，如果遇到权限问题，需要在命令前加sudo，也就是输入sudo brew node，然后再输入你的mac密码即可，检查node版本命令是，node -v；

```
humdeMac-mini:~ hum$ node -v
v7.1.0
humdeMac-mini:~ hum$
```

**5）命令：~hum$ brew npm**

npm是一个NodeJS包管理和分发工具，已经成为了非官方的发布Node模块（包）的标准，概念不懂没关系，先安装，安装到了总没错的；

**6）命令：~hum$ npm install -g cnpm --registry=https://registry.npm.taobao.org**

因为国内直接用npm下载安装会有好多网络问题，安装cnpm要比npm好用，所以安装多一个cnpm有备无患，安装完这个cnpm，如果使用npm安装软件出现什么问题，那么就使用cnpm吧，所有npm的命令都可以用cnpm代替。

说到这里我就遇到了一个深坑，当时后续使用npm安装appium或者其他依赖的时候，总有各种各样的问题，不得已，我只能用按照网上原因使用cnpm，使用它前提是需要安装，但输入命令后一直链接不上，总是报安装失败，找了很久找不到为什么安装不了的原因，这个问题卡了很久，

直到有一天我细细的看了下命令有taobao，又想起我们公司的网络是禁止上taobao的，taobao都上不去，能下载安装那就真有鬼了，这果然是个坑啊，后来解决原因就是让公司的网络管理员给我电脑给暂时开通了访问taobao的权限才解决这个问题。

**7）命令：~hum$ npm install -g appium**

轮到安装appium，如果npm安装有问题，就使用cnpm，如果遇到权限问题，就加上sudo，前面我提到了，后续就不提醒了，安装成功后，输入 appium -v 检查版本或检查是否安装成功；

```
humdeMac-mini:~ hum$ appium -v
1.6.5
humdeMac-mini:~ hum$
```

**8）命令：~hum$ npm install wd**

wd是可视化界面，具体干嘛的我也不清楚，百度了也没有找到答案，不知道是不是必须的，反正还是先安装吧。

**9）命令：~hum$ npm install -g appium-doctor**

安装好appium-doctor，输入appium-doctor查看其ios环境配置是否好（有关Android字眼的x可忽略，目前我们只做ios-appium测试），按照之前的命令，ios的环境配置肯定是没有配置好，因为还缺些依赖，所以我们还需要继续安装；

**10）命令：~hum$ brew install carthage**

carthage 使用于 Swift 语言编写，只支持动态框架，只支持 iOS8+的Cocoa依赖管理工具，不理解的话，还是直接安装，不用怀疑；

**11）命令：~hum$ brew install libimobiledevice --HEAD**

libimobiledevice 是一个跨平台的软件库，支持 iPhone®, iPod Touch®, iPad® and Apple TV® 等设备的通讯协议。不依赖任何已有的私有库，不需要越狱。应用软件可以通过这个开发包轻松访问设备的文件系统、获取设备信息，备份和恢复设备，管理 SpringBoard 图标，管理已安装应用，获取通讯录、日程、备注和书签等信息，使用 libgpod 同步音乐和视频。

不能理解没关系，直接安装就对了；

**12）命令：~hum$ cnpm install -g ios-deploy**

ios-deploy是一个使用命令行安装ios app到连接的设备的工具，原理是根据os x命令行工程调用系统底层函数，获取连接的设备、查询/安装/卸载app。

就是用来获取并连接设备，然后进行安装卸载查询等等，用处大大的有，装。

到这一步，再输入appium-doctor命令查看环境配置，完好的配置如下图：

```
humdeMac-mini:~ hum$ appium-doctor
info AppiumDoctor Appium Doctor v.1.4.3
info AppiumDoctor ### Diagnostic starting ###
info AppiumDoctor  ✓ The Node.js binary was found at: /usr/local/bin/node
info AppiumDoctor  ✓ Node version is 7.1.0
info AppiumDoctor  ✓ Xcode is installed at: /Applications/Xcode.app/Contents/Developer
info AppiumDoctor  ✓ Xcode Command Line Tools are installed.
info AppiumDoctor  ✓ DevToolsSecurity is enabled.
info AppiumDoctor  ✓ The Authorization DB is set up properly.
info AppiumDoctor  ✓ Carthage was found at: /usr/local/bin/carthage
info AppiumDoctor  ✓ HOME is set to: /Users/hum
WARN AppiumDoctor  ✖ ANDROID_HOME is NOT set!
info AppiumDoctor  ✓ JAVA_HOME is set to: /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home
WARN AppiumDoctor  ✖ adb could not be found because ANDROID_HOME is NOT set!
WARN AppiumDoctor  ✖ android could not be found because ANDROID_HOME is NOT set!
WARN AppiumDoctor  ✖ emulator could not be found because ANDROID_HOME is NOT set!
info AppiumDoctor  ✓ Bin directory of $JAVA_HOME is set
info AppiumDoctor ### Diagnostic completed, 4 fixes needed. ###
info AppiumDoctor
info AppiumDoctor ### Manual Fixes Needed ###
info AppiumDoctor The configuration cannot be automatically fixed, please do the following first:
WARN AppiumDoctor - Manually configure ANDROID_HOME.
WARN AppiumDoctor - Manually configure ANDROID_HOME and run appium-doctor again.
info AppiumDoctor ###
info AppiumDoctor
info AppiumDoctor Bye! Run appium-doctor again when all manual fixes have been applied!
info AppiumDoctor
```

如果缺少Xcode Command Line Tools，需要命令xcode-select --install，反正缺什么就装什么，上网百度，至此，appuim的环境已经搭配好了，以上的其实很简单，但是这只是一个开始，前面遇到什么坑都不算大风大浪，接下来就要安装WebDriverAgent；

**13）命令： npm i -g webpack**

这个非常重要，和WebDriverAgent相关，首先是要进行这个，没有这个会报错.遇到权限问题就sudo；

**14）安装WebDriverAgent依赖**

命令：~hum$ cd ~
命令：~hum$ git clone https://github.com/facebook/WebDriverAgent.git
命令：~hum$ cd WebDriverAgent
命令：~hum$ mkdir -p Resources/WebDriverAgent.bundle
命令：~hum$ ./Scripts/bootstrap.sh # 开始下载并编译

命令：~hum$ cd/Applications/Appium.app/Contents/Resources/app/node_modules/appium/node_modules/appium-xcuitest-driver/ （注意其路径，这路径不一定是对的，看其所在文件夹）
命令：~hum$ rm -rf WebDriverAgent # 把自带的删掉

命令：~hum$  ln -s ~/WebDriverAgent WebDriverAgent  # 用facebook的原版替换回去

这一步主要是安装WebDriverAgent依赖。

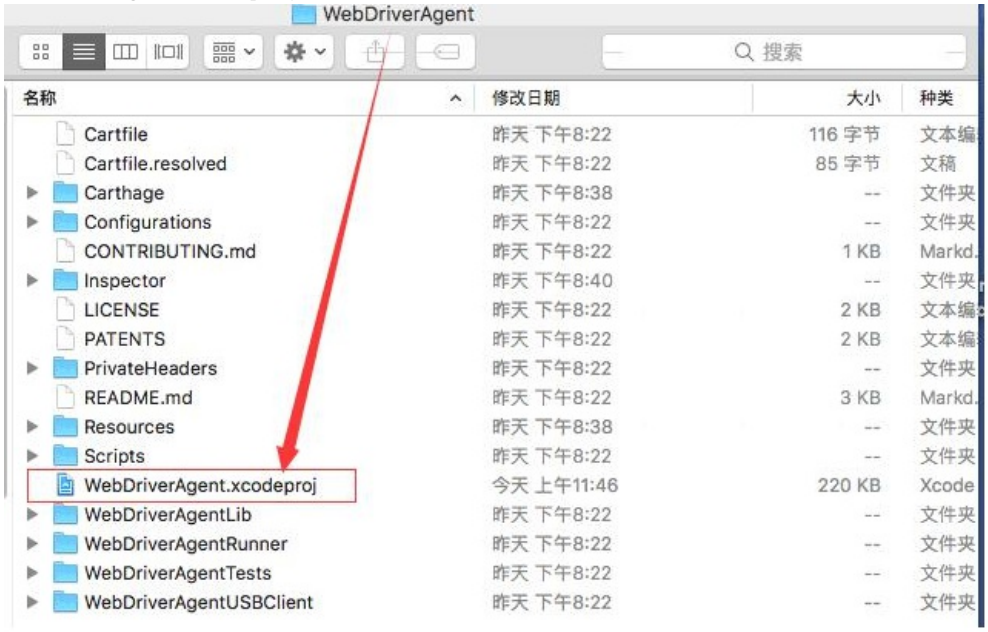
WebDriverAgent依赖就是ios自动化测试框架，表示在 iOS 端实现了一个 WebDriver server ，借助这个 server 我们可以远程控制 iOS 设备。你可以启动、杀死应用，点击、滚动视图，或者确定页面展示是否正确，它支持模拟器也支持真机。

而iOS 10+使用的是XCUITest，Appium使用的模块是appium-xcuitest-driver，所以我们引用了Facebook提供的WebDriverAgent来驱动iOS的测试。通俗来说就是通过WebDriverAgent来对手机进行Ui界面的操作，模拟用户的行为。

在之前使用命令行装Appium的时候，它里面其实是带了一个WebDriverAgent，但是这个自带的是有问题的，会造成不能使用Inspector，卡了很久，所以我们在这里使用命令，从Facebook那里自己clone一份，并覆盖掉原来的。

15）**在mac电脑用Xcode打开WebDriverAgen**t：

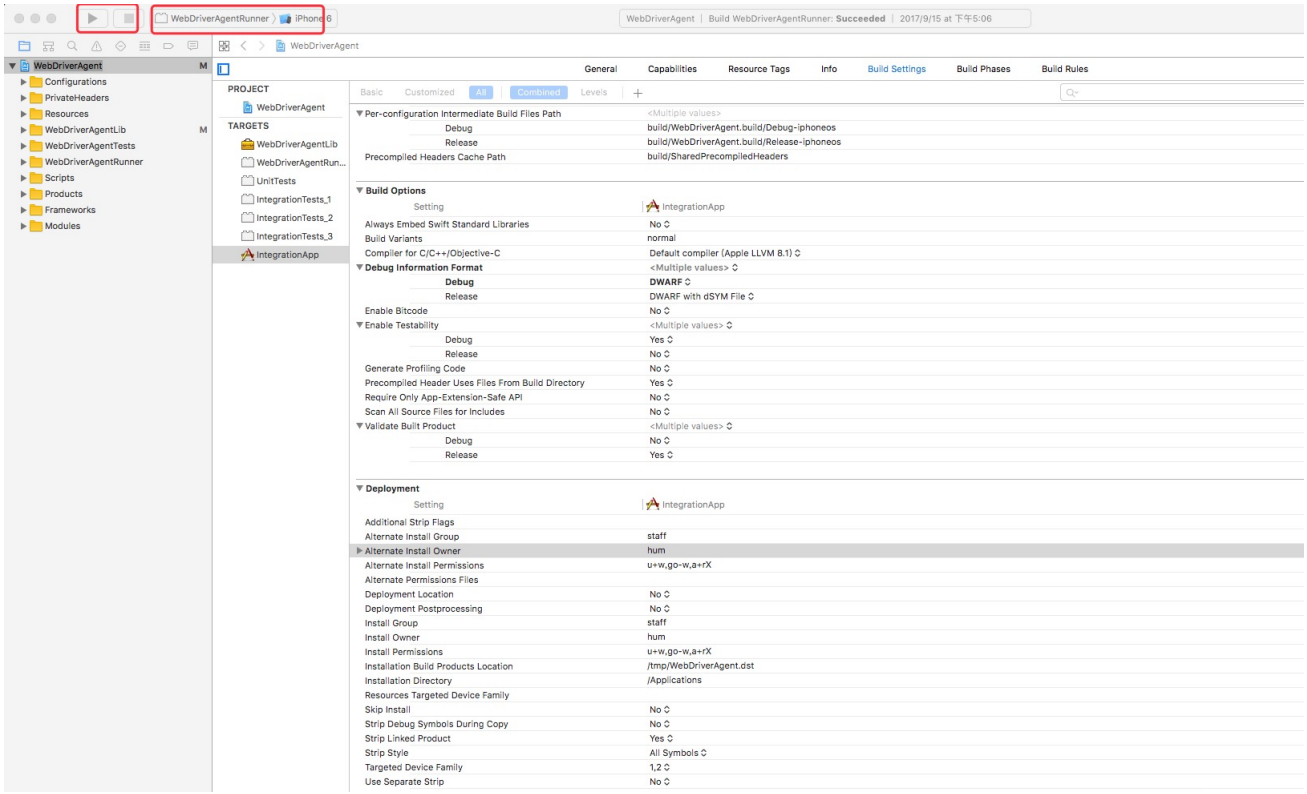
使用finder的前往文件夹，输入/user/hum/webDriverAgent/，进入此文件夹后，找到webDriverAgent.xcodeproject文件，双击打开即可，因为这是用模拟器测试，所以文件什么的暂时不用改；

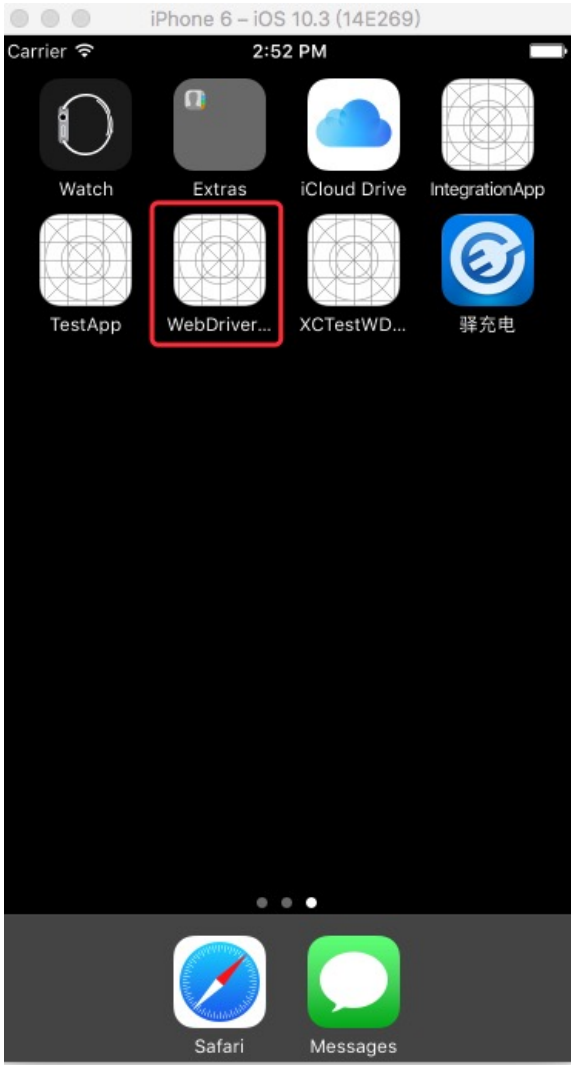



16）**选择WebDriverAgentRunner，选择iPhone6 plus，点击run按钮，**

一般没有问题，都会build成功，然后启动iPhone6 plus模拟器成功，并在此模拟器可以看到它装了个webDriverAgen应用，点击它会闪一下，黑屏再回到首页，这是正常的；

至此，mac端的相关依赖安装结束；



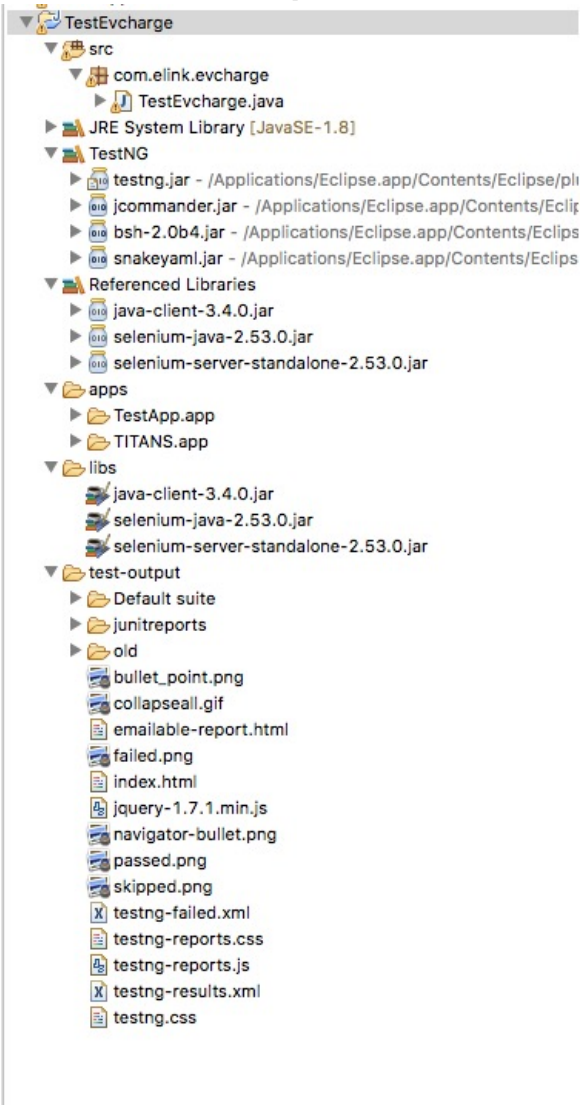

**17）打开ecplise建立测试项目；**

（1）打开ecpilise，新建一个javaproject，名字为testEvcharge

（2）在项目buildPath--》AddLibraries-->TestNG，添加TestNG；

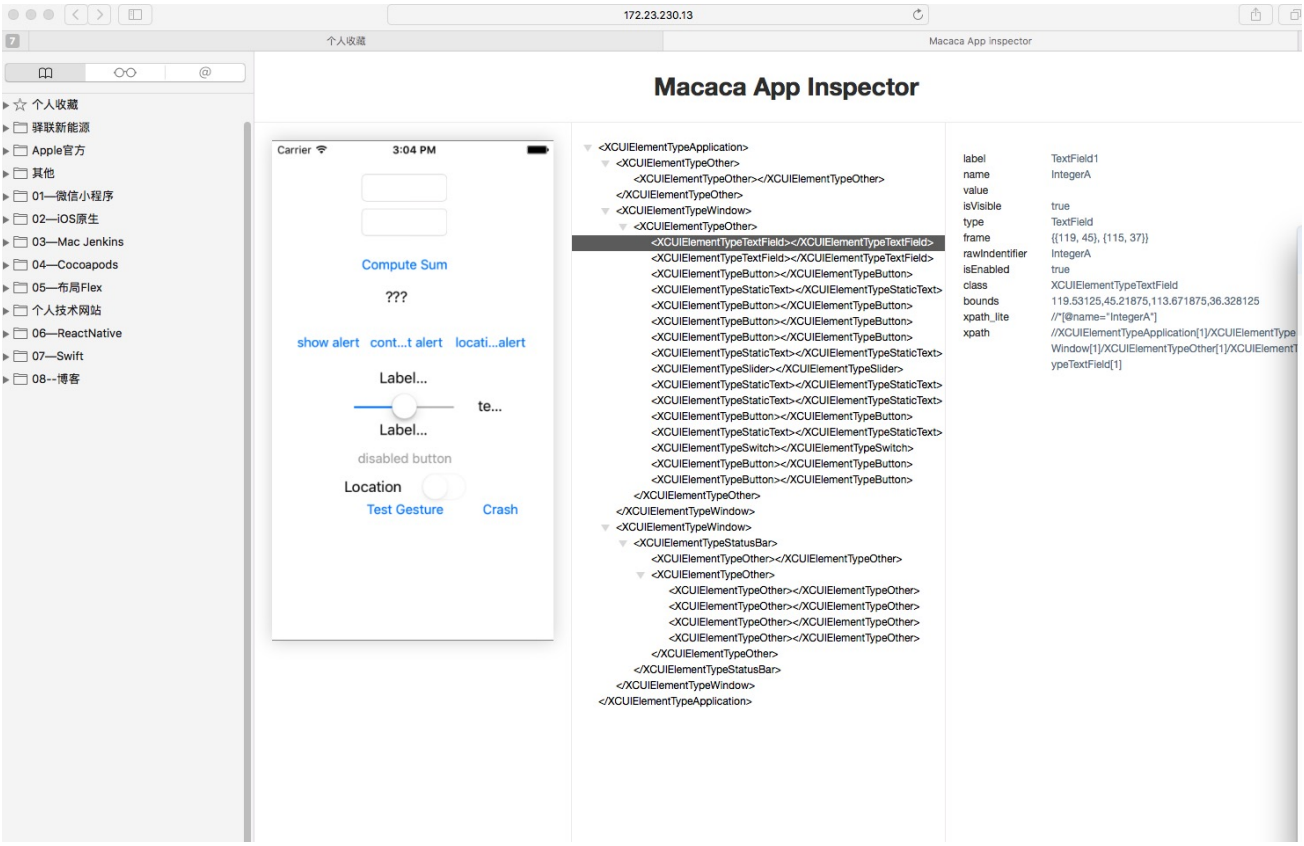
（3）在项目新建一个libs，放入Selenum-server-standalone.jar、java-client.jar和Selenum-java-standalone.jar包，并通过buildaPath-->addtoBuildPath；

（4）在项目新建一个apps，专门放置被测app，这里我放了一个格式为app的驿充电Test.App；
（5）在src新建一个包，名字为com.elink.evcharge
（6）在包上新建一个TestEvchage.java的class
（7）编写测试用例，在编写测试用例前需要打开inspector来定位元素来进行用例编写，操作请查看第18步；




**18）打开inspector；**

（1）.获取serverid：~hum$ xcrun simctl list
（2） 命令：~hum$app-inspector -u your serverid --verbose
（3）打开insprestor就可以定位元素来进行用例编写，完成第17步的第7步；
（4）完成用例编写好关闭浏览器的inspector




以下是编写好的测试代码：运行TestApp，并输入数字计算它的总和；

```
import java.io .File;
import java.net .URL;

import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import io.appium.java_client.ios.IOSDriver;
import io.appium.java_client.ios.IOSElement;

public class TestEvcharge {


 static IOSDriver driver= null;
 @BeforeClass
 public void setUp() throws Exception {
  // set up appium

  File classpathRoot = new File(System.getProperty("user.dir"));
  File appDir = new File(classpathRoot, "apps");
  File app = new File(appDir, "TestApp.app");
  DesiredCapabilities capabilities = new DesiredCapabilities();
  capabilities.setCapability("platformName", "iOS");// 设备平台
  capabilities.setCapability("app", app.getAbsolutePath());
  capabilities.setCapability("deviceName", "iPhone 6");// 设备名称
  capabilities.setCapability("platformVersion", "10.3");// 版本
  capabilities.setCapability("noReset", true);
  capabilities.setCapability("unicodeKeyboard", "True");
  capabilities.setCapability("resetKeyboard", "True");
  capabilities.setCapability("noSign", true);// 禁止重签名
  capabilities.setCapability("autoAcceptAlerts", true);// 自动接受提示信息
  capabilities.setCapability("sessionOverride", true);// 每次启动时覆盖sesssion，否则第二次运行报错后不能新建session
```

```
    driver = new IOSDriver(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);

  }

  @AfterClass

  public void tearDown(){

    driver.quit();

  }
  @Test
  public void testCaseTestApp01() throws Exception {
    Thread.sleep(5000);
    driver.findElementByName("IntegerA").sendKeys("23");
    driver.findElementByName("IntegerB").sendKeys("34");

    IOSElement buttonEL = (IOSElement) driver.findElementsByClassName("UIAButton").get(0);
    buttonEL.click();
    System.out.println(driver.findElementByName("Answer").getText());
    Thread.sleep(5000);
  }

}
```

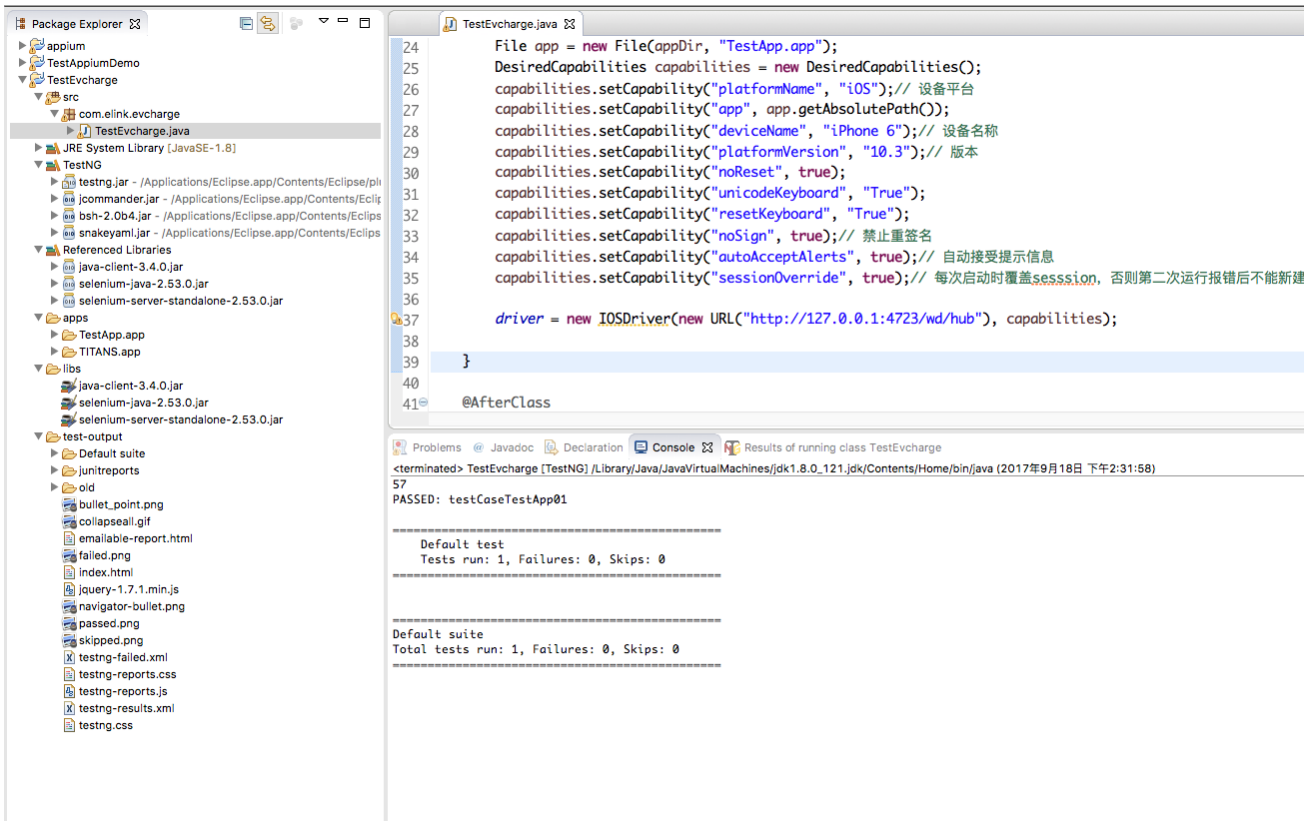
19）打开appium服务；

命令为：~hum$ appuim

```
humdeMac-mini:~ hum$ appium
[Appium] Welcome to Appium v1.6.5
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```

20）右键点击TestEvcharge.class，再点击run as，选择TestNG test运行测试；

从测试代码所知，运行的测试用例是testCaseTestApp01，第一个输入框输入23，第二个输入框输入34，计算总和是57，从下图运行结果来看，测试为57，通过；

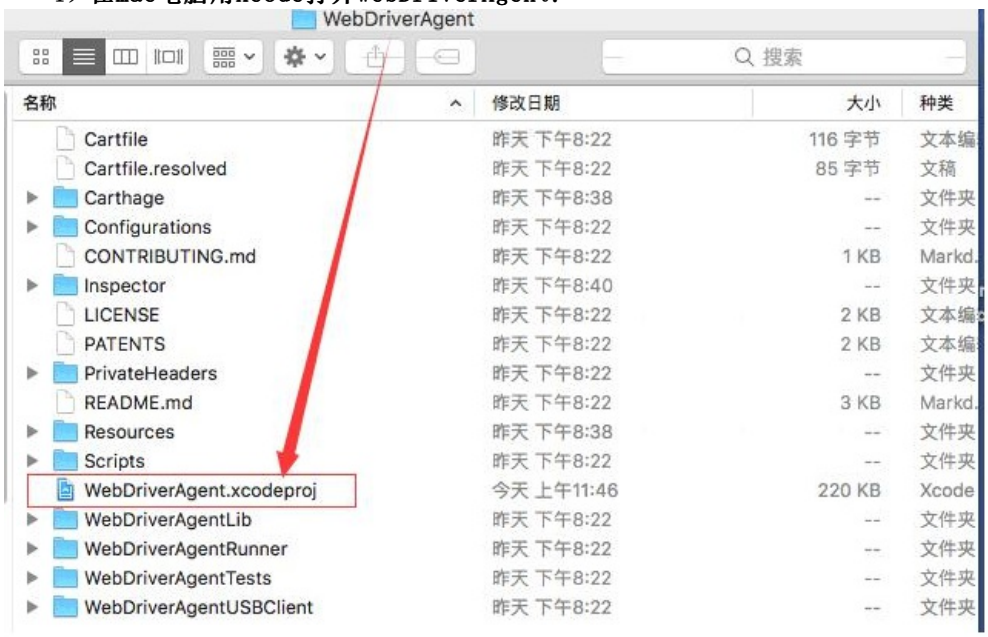


**至此，模拟器的ios-appium自动化测试完成；**

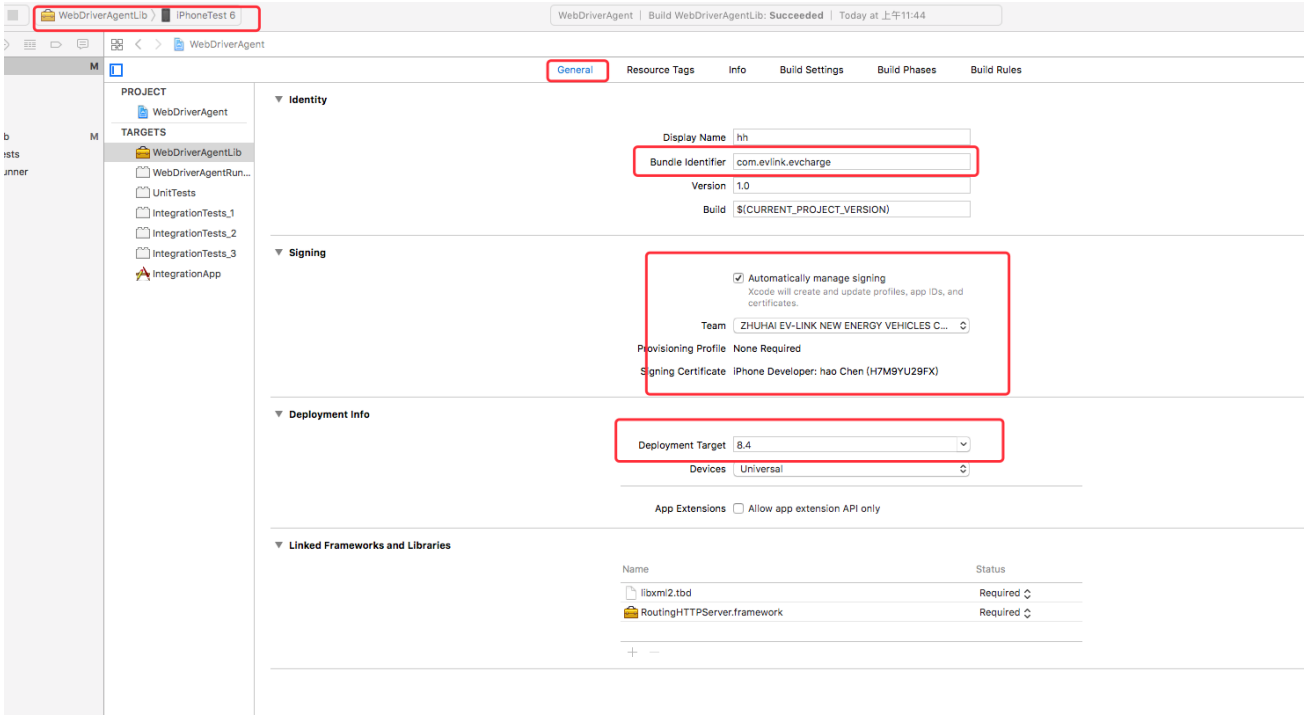
**2.3真机环境搭建 （mac OS10.12+Xcode8.3+iPhone6真机+10.3）**

如果模拟器的环境已经搭建好并成功的运行了测试，那么真机的步骤就只需要从第15步开始；

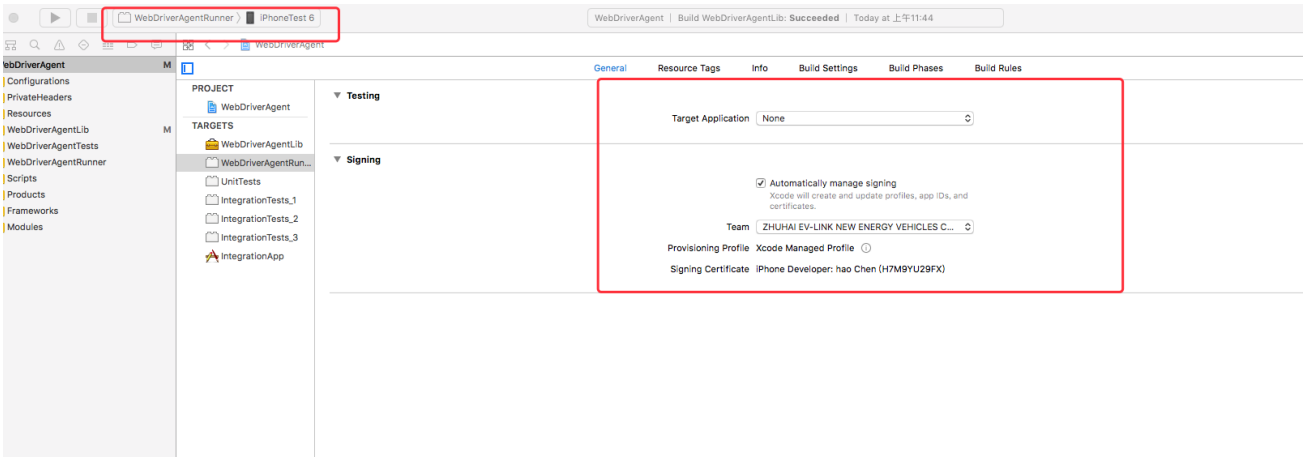
1）**在mac电脑用Xcode打开WebDriverAgent**：



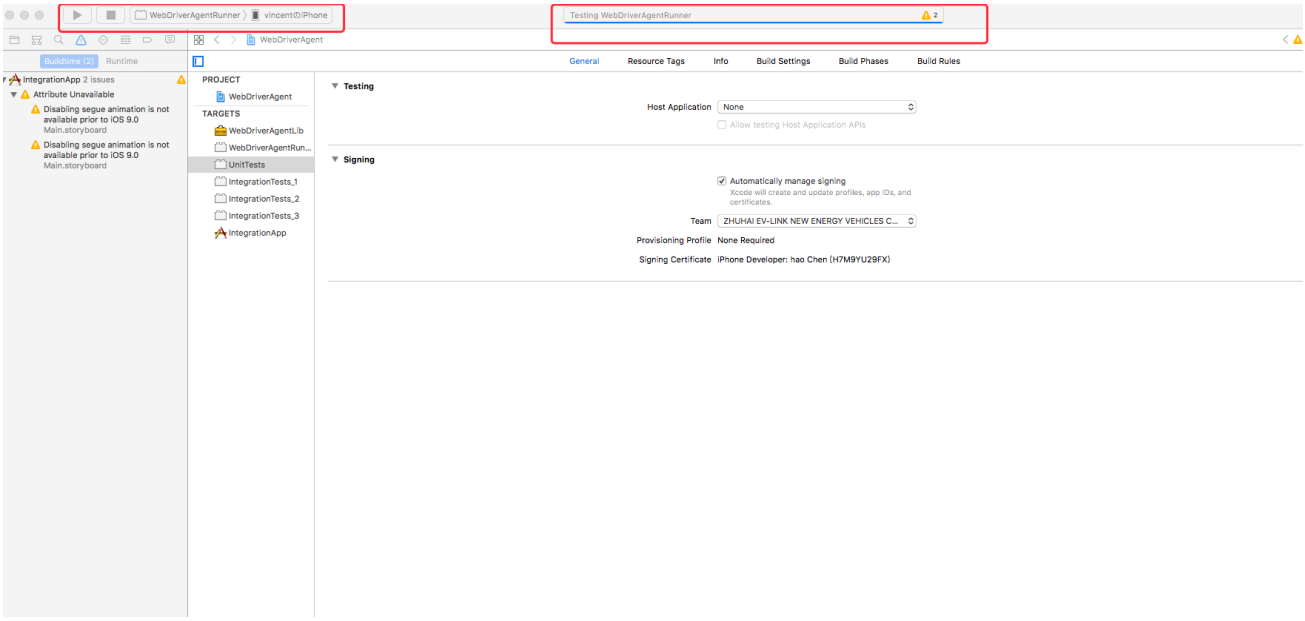

2）打开并修改webDriverAgent.xcodeproject文件，首选在TARGETS里面选中WebDriverAgentlib进行证书设置，然后build；



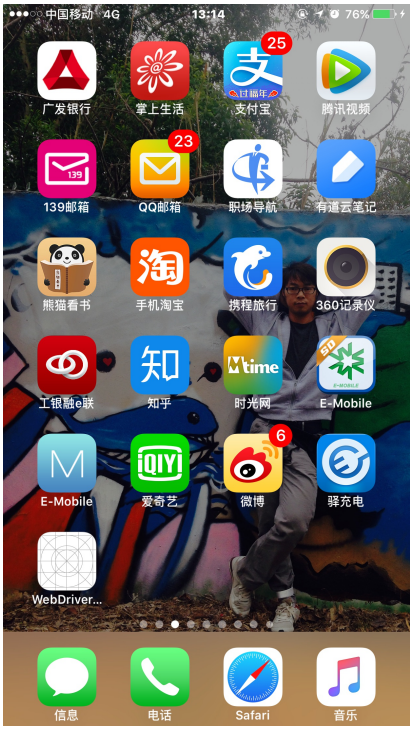

3）接着在TARGETS里面选中WebDriverAgentRunner修改，设置也如上，然后build；




4）将手机连接真机，菜单栏选择目标设备，Scheme选择WebDriverAgentRunner，最后运行 Product -> Test，最后将WebDriverAgentRunner安装到真机上；

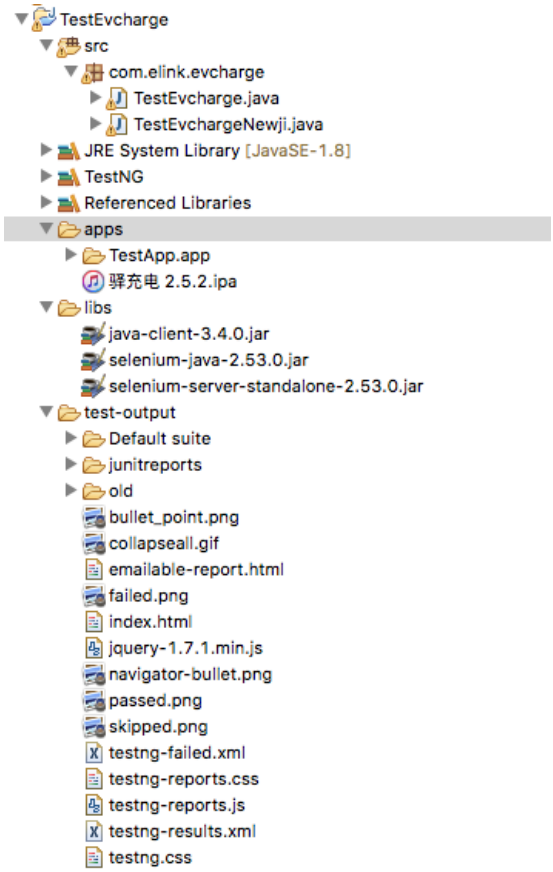



5）一般没有问题，都会build成功，然后启动真机成功，并在真机可以看到它装了个webDriverAgen应用，点击它会闪一下，黑屏再回到首页，这是正常的，至此，mac端的在真机相关依赖安装结束；




6）用**ecplise**打开之前建立好的测试项目Testevcharge，在com.elink.evcharge再建一个测试类

TestEvchageNew.java，并编写代码




以下是编写好的测试代码：运行驿充电app，让它进行登录与退出测试；

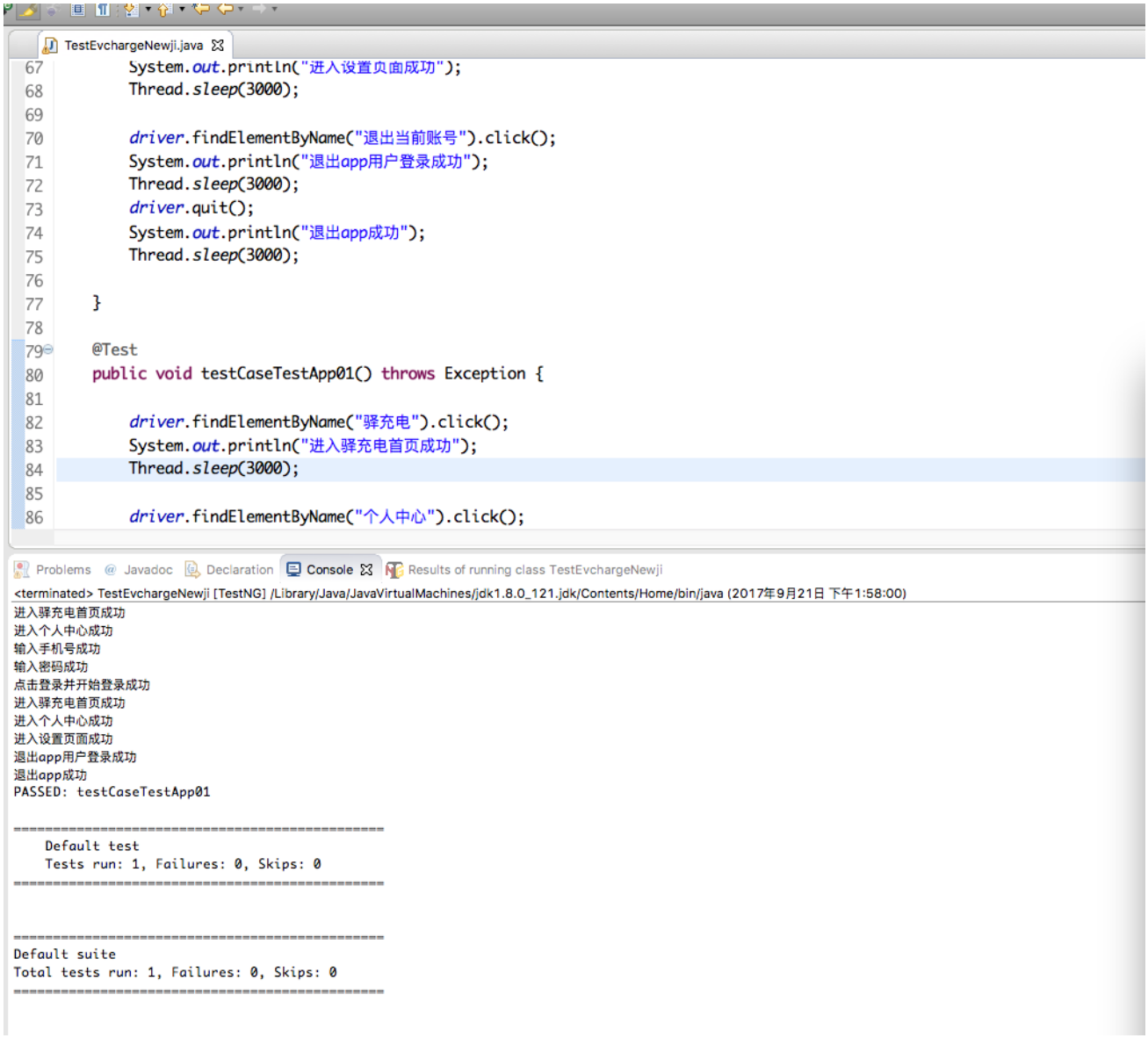
```
package com.elink.evcharge;

import java.io

.File;
import java.net

.URL;

import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import io.appium.java_client.ios.IOSDriver;
import io.appium.java_client.ios.IOSElement;

public class TestEvchargeNewji {


 static IOSDriver driver= null;
 @BeforeClass
 public void setUp() throws Exception {
  // set up appium

  File classpathRoot = new File(System.getProperty("user.dir"));
  File appDir = new File(classpathRoot, "apps");
  File app = new File(appDir, "驿充电 2.5.2.ipa");
  DesiredCapabilities capabilities = new DesiredCapabilities();
  capabilities.setCapability("platformName", "iOS");// 设备平台
  capabilities.setCapability("app", app.getAbsolutePath());
```

```
  capabilities.setCapability("deviceName", "iPhone 6s plus");// 设备名称
  capabilities.setCapability("platformVersion", "10.0");// 版本
  capabilities.setCapability("udid", "24a445091e192208e23e3fd2f81d28bc0e8625e7");// 手机ID
  capabilities.setCapability("noReset", true);
  capabilities.setCapability("unicodeKeyboard", "True");
  capabilities.setCapability("resetKeyboard", "True");
  capabilities.setCapability("noSign", true);// 禁止重签名
  capabilities.setCapability("sessionOverride", true);// 每次启动时覆盖sesssion，否则第二次运行报错后不能新建session

  driver = new IOSDriver(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);

  driver.findElementByName("驿充电").click();
  System.out.println("进入驿充电首页成功");
  Thread.sleep(3000);
  Thread.sleep(3000);
  driver.findElementByName("个人中心").click();
  Thread.sleep(3000);
  System.out.println("进入个人中心成功");


  driver.findElementByName("手机号/用户名").sendKeys("15992667873");
  Thread.sleep(3000);
  System.out.println("输入手机号成功");
  driver.findElementByName("请输入6位数密码").sendKeys("123456");
  Thread.sleep(3000);
  System.out.println("输入密码成功");
  driver.findElementByName("登  录").click();
  System.out.println("点击登录并开始登录成功");
  Thread.sleep(8000);


 }

 @AfterClass

 public void tearDown() throws InterruptedException{

  driver.findElementByName("个人中心").click();
  System.out.println("进入个人中心成功");
  Thread.sleep(3000);

  driver.findElementByName("topnav cog ico wir").click();
  System.out.println("进入设置页面成功");
  Thread.sleep(3000);

  driver.findElementByName("退出当前账号").click();
  System.out.println("退出app用户登录成功");
  Thread.sleep(3000);
  driver.quit();
  System.out.println("退出app成功");
  Thread.sleep(3000);

 }

 @Test
 public void testCaseTestApp01() throws Exception {

  driver.findElementByName("驿充电").click();
  System.out.println("进入驿充电首页成功");
  Thread.sleep(3000);

  driver.findElementByName("个人中心").click();
  System.out.println("进入个人中心成功");
  Thread.sleep(3000);

  driver.findElementByName("我的记录").click();
  System.out.println("进入我的记录成功");
  Thread.sleep(3000);
```

```
    }

}
```

20）右键点击TestEvchargeNewJI.class，再点击run as，选择TestNG test运行测试；

从测试代码所知，运行的测试用例是testCaseTestApp01，测试用例是执行用户登录与退出，测试通过；




**至此，真机的ios-appium自动化测试完成；**

## 3 最后的总结：

1）过程很简单，但实施很难，很多人别说是进入到建测试项目、写测试用例，就是死在搭建环境上也是常事。

2）遇到很多的坑，上面我只提及了一个，其中有几个坑都卡了好几天甚至一两个星期没办法解决，这时候我们需要的就是静心，多翻资料，多看运行日志，遇到英文看不懂的，可以使用翻译，另外要多思考，这点很重要；

3）另外，网上也特别缺乏java语言，特别是使用ecplise来进行建立测试项目的，资料少之又少，对于所谓的TestNG测试框架了解也不足，更增加了进一步研究的困难。

4）后续工作目标是熟悉TestNG测试框架，并进行appium的case开发以及case分层结构设计，以便更好的管理测试用例；

## 4. 参考资料

- testerhome论坛
- appium源码地址
- appium官网地址
- app-inspector
- macaca
- appium中文说明文档
- http://www.jianshu.com/p/05943804c25e