

macOS 下的Appium安装与配置 Appium Installation & Setup With macOS

iOS

系统要求及说明：

- macOS 10.11 或更高
- `brew` 需要最新版本的 `Xcode` 或 `Xcode Command Line`

介绍

iOS9.3及以下使用 Automation 作为底层技术，在iOS9.3出现了XCUITest，并且在iOS10时抛弃了Automation，所以Appium在iOS9.3及以下和iOS10的配置方法不同。

安装Appium

如果已安装过部分依赖，可直接跳过

安装Homebrew

强烈推荐所有macOS上的软件使用Homebrew进行安装，使用brew安装过的软件，不再需要sudo命令。

打开终端，运行：

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

brew的使用非常简单，安装一个软件使用

```
brew install #<package name>
```

如果brew提示Xcode版本过低，需要安装新版本的Xcode（安装时保存两个版本即可）

下载地址：<https://developer.apple.com/download/more/>

安装Nodejs

```
brew install node
```

检查是否安装成功

```
node -v
# v6.9.1
```

```
npm -v  
# 3.10.8
```

安装cnpm

由于npm源在国外，我们使用npm安装库时会比较慢，可以使用taobao的镜像源

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

检查安装是否成功

```
cnpm -v  
# 4.4.0
```

以后均使用 `cnpm` 命令代替 `npm`

安装appium

使用Node.js

appium本质上就是一个Nodejs应用，我们可以使用npm对其进行安装，安装完毕后就可以使用命令行启动

```
cnpm install -g appium
```

可以使用appium-doctor来确认安装环境是否完成

```
cnpm install -g appium-doctor
```

```
appium-doctor
```

下载App

使用软件比较简单，拥有图形化界面，并且有Inspector工具，帮助检测界面元素

下载地址：<https://github.com/appium/appium-desktop/releases>

设置Appium

安装Carthage

Carthage 是一个管理 iOS 开发库依赖的包

```
brew install carthage
```

安装语言依赖

python

安装Python3（使用Python2跳过）

```
brew install python3
```

检测安装成功

```
python3 -V  
# Python 3.6.0
```

```
pip3 -V  
# pip 9.0.1 from /usr/local/lib/python3.6/site-packages (python 3.6)
```

出现相应代码即成功

macOS自带有Python2版本，默认情况下输入 `python` 调用的就是系统的python2和 `pip`

此外，如果系统版本之前安装过python3导致安装后输入python3没有找到命令，可以尝试 `brew link --overwrite python3` 重新链接

使用 `pip` 安装

```
pip3 install Appium-Python-Client  
# python3 -> pip3  
# python -> pip
```

详细信息：<https://github.com/appium/python-client>

Java

安装JDK

```
brew update  
brew cask install java
```

使用Maven进行依赖

```
<dependency>  
  <groupId>io.appium</groupId>  
  <artifactId>java-client</artifactId>  
  <version>5.0.0-BETA7</version>  
  <scope>test</scope>  
</dependency>
```

详细信息：<https://github.com/appium/java-client>

安装相关依赖

包含了记录日志的 `idevicesyslog` 和端口转发的 `iProxy`、`usbmuxd`

```
brew install libimobiledevice --HEAD #和ios手机通讯使用
```

```
brew install ideviceinstaller # 安装app使用
```

在iOS10需要使用 `ios-deploy`

```
cnpm install -g ios-deploy
```

对真机需要 `xcpretty`

```
gem install xcpretty
```

`gem` 安装速度慢可以更换镜像源

```
gem sources --add https://gems.ruby-china.org/ --remove  
https://rubygems.org/
```

第一次运行配置

不配置会出现 `xcodebuild exited with code '65' and signal 'null'` 错误

开发者账号：

AppiumDesktop中加入 `xcodeOrgId` 和 `xcodeSigningId` 字段

Desired Capabilities		Saved Capability Sets (2)	
deviceName	text	platformVersion	text
noReset	boolean <input checked="" type="checkbox"/>		
app	text <input type="text" value="/Users/.../GitHu"/>		
udid	text <input type="text" value="...011bd1a001a"/>		
xcodeOrgId	text <input type="text" value="...2Y2"/>		
xcodeSigningId	text <input type="text" value="iPhone Developer"/>		

TeamID 在 <https://developer.apple.com/account/> 里左侧的 Member ship 中找到

需要 `xcodeOrgId` 和 `xcodeSigningId` 写在desired capabilities 中

```
{
  "xcodeOrgId": "<Team ID>",
  "xcodeSigningId": "iPhone Developer"
}
```

Team ID 在 <https://github.com/developer.apple.com/account> 中的Membership

免费AppleID帐号

1. 在appium的出错日志中找到WebDriverAgent的位置，打开进行签名

```
[XCUITest] Using WDA path:
'/usr/local/lib/node_modules/appium/node_modules/.2.23.1@appium-xcuitest-
driver/WebDriverAgent'
```

2. 更改 `WebDriverAgent` 的 `BundleID`，原来是 `com.facebook.WebDriverAgentLib`，我建议把 `facebook` 改为你的名字或公司的名字，这里还有个坑，改完这个之后发现还是提示有错误，最后在 `Build Settings` 里的 `Product Bundle Identifier` 里再修改一下。

Info.plist file	WebDriverAgentLib/Info.plist
▶ Product Bundle Identifier	com.wangxiaoying.WebDriverAgen...
Product Name	WebDriverAgentLib

3. 点击上面菜单栏的Product -> Test 运行
第一次运行会提示没有权限，打开真机上的设置App，通用 -> 描述文件与设备管理，找到你的邮箱，选择信任，再继续执行test，成功启动后就说明可以了，执行完毕后点击停止。
4. 没有了，打开appium测试吧

查看界面元素

- 使用WebDriverAgent

```
从 https://github.com/facebook/WebDriverAgent 克隆，执行
./Scripts/bootstrap.sh 命令，然后打开项目进行签名，
执行方式
1. 接着点击上面菜单栏的Product --> test
2. xcodebuild -project WebDriverAgent.xcodeproj -scheme
   WebDriverAgentRunner -destination id='udid' test
# udid 可以使用命令 idevice_id -l
```

- 使用AppiumDesktop
- app-inspector

```
# macaca-cli
npm install macaca-cli -g
# macaca doctor
macaca doctor
# app-inspector
npm install app-inspector -g
# 使用
app-inspector -u 'udid'
```

253错误

如果你运行官方的TestApp示例，可能会爆出253错误，这时需要对测试App重新build（需要源代码）

xcodebuild 命令

project

```
xcodebuild [-project projectname] [-target targetname ...] [-configuration
configurationname]
            [-sdk [sdkfullpath | sdkname]] [buildaction ...] [setting=value
...]
```

workspace

```
xcodebuild -workspace workspaceName -scheme schemeName [-destination
destinationSpecifier]
            [-destination-timeout value] [-configuration configurationName]
            [-sdk [sdkfullpath | sdkname]] [buildaction ...] [setting=value
...]
```

示例

```
# xcodebuild -project RobotCalibration.xcodeproj -target RobotCalibration -
sdk iphoneos -configuration development
```

DesiredCapabilities 配置信息

- 确定 `platformVersion`
- 确定 `bundleId`
- 一台设备 `udid` 可以设置为 `auto`
- 不提供 `app` 属性也可以调起 App，但需要设置 `bundleId`
- 测试 iOS 9.3以下时，需要设置 `automationName` 为 `Automation`，其它为 `XCUITest`

常用命令

切换Xcode版本

```
sudo xcode-select -s #path /Applications/Xcode.app/
```

查看设备udid

```
idevice_id -l  
# 或者使用 ideviceinfo 可以在返回的数据中找到 udid
```

Automation (iOS 10以下)

推荐使用:

Xcode7.3.1

注意:

iOS8 中需要确定设置中的开发者选项里的UIAutomation 为打开状态

有开发者帐号 —> 打开Xcode —> 对设备注册

免费帐号 —> 打开项目 —> 修改 bundleID —> 登录帐号进行签名

测试前查看设定的App目录是否正确

接着就可以打开Appium, 和测试用例了

如果不牵扯自动安装和卸载App, 可以先手动安装App到目标设备, 再进行测试

前提需要 `ideviceinstaller`

```
ideviceinstaller -u <UDID of device> -i <path of .app/.ipa>
```

Instruments exited with code: 253

[Instruments] Error launching instruments: Instruments crashed on startup

可能是App路径不正确

授权iOS模拟器

需要测试模拟器进行此项, 使用npm安装

```
cnpm install -g authorize-ios
```

安装完毕后运行

```
sudo authorize-ios
```

注意: 需要在安装完新的 Xcode 后再次执行

通过 Jenkins 运行 iOS 测试（未测试）

确认之前 `authorize-ios` 运行成功

下载 jenkins

```
wget https://jenkins.ci.cloudbees.com/jnlpJars/jenkins-cli.jar
```

接下来定义一个 Jenkins 的 LaunchAgent来自动登录，请确保其不包含 `SessionCreate` 或 `User` key 防止测试运行

```
java -jar jenkins-cli.jar \  
-s https://team-appium.ci.cloudbees.com \  
-i ~/.ssh/id_rsa \  
on-premise-executor \  
-fsroot ~/jenkins \  
-labels osx \  
-name mac_appium
```

最后设置并启动


```

sudo nano /Library/LaunchAgents/com.jenkins.ci.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.jenkins.ci</string>
  <key>ProgramArguments</key>
  <array>
    <string>java</string>
    <string>-Djava.awt.headless=true</string>
    <string>-jar</string>
    <string>/Users/appium/jenkins/jenkins-cli.jar</string>
    <string>-s</string>
    <string>https://instructure.ci.cloudbees.com</string>
    <string>on-premise-executor</string>
    <string>-fsroot</string>
    <string>/Users/appium/jenkins</string>
    <string>-executors</string>
    <string>1</string>
    <string>-labels</string>
    <string>mac</string>
    <string>-name</string>
    <string>mac_appium</string>
    <string>-persistent</string>
  </array>
  <key>KeepAlive</key>
  <true/>
  <key>StandardOutPath</key>
  <string>/Users/appium/jenkins/stdout.log</string>
  <key>StandardErrorPath</key>
  <string>/Users/appium/jenkins/error.log</string>
</dict>
</plist>

```

Finally set the owner, permissions, and then start the agent.

```

sudo chown root:wheel /Library/LaunchAgents/com.jenkins.ci.plist
sudo chmod 644 /Library/LaunchAgents/com.jenkins.ci.plist

launchctl load /Library/LaunchAgents/com.jenkins.ci.plist
launchctl start com.jenkins.ci

```

清理文件（可选）

```
$HOME/Library/Logs/CoreSimulator/
```

```
/Library/Caches/com.apple.dt.instruments/
```

```
$HOME/Library/Developer/Xcode/DerivedData/
```

以上目录下的所有文件

Android

安装maven

```
brew install maven
```

额外的设置

1. 下载JDK
2. 通过 Android Studio 下载 Android SDK，并设置目录

macOS 设置 JavaHome和 Android SDK 目录，将下列代码放置到用户目录下的 `.bashrc` 或 `.zshrc` 中

```
# 使用Android Studio安装后的路径
# android sdk
export ANDROID_HOME=~/.Library/Android/sdk
export PATH="$HOME/.yarn/bin:$PATH"
# java_home
export JAVA_HOME=$(/usr/libexec/java_home)
export PATH=$JAVA_HOME/bin:$PATH
export CLASS_PATH=$JAVA_HOME/lib

# adb等
export PATH=${PATH}:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools
export PATH=${PATH}:$ANDROID_HOME/tools/bin
```

3. 最后，可以使用 appium-doctor 来检查配置环境

```
appium-doctor
```

4. 在 `capabilities` 写清楚

- o 确定 `platformVersion`
- o 确定 `appPackage`
- o 确定 `appActivity`

