Appium+RobotFramework 实现Android应用自动化测试

1. 环境搭建

=======所需环境===================

1. Python：

资源地址：<https://www.python.org/>

RF框架是基于Python的，所以一定要有Python环境

1. Robot Framework

资源地址：<https://pypi.python.org/pypi/robotframework/2.8.5>

1. wxPython

资源地址：<http://www.wxpython.org/download.php>

wxPython是Python非常有名的一个GUI库，因为RIDE 是基于这个库开发的，所以这个必须安装。

1. Robot Framework-ride

资源地址：<https://pypi.python.org/pypi/robotframework-ride>

RIDE就是一个图形界面的用于创建、组织、运行测试的软件。

1. Appium（参考Appium相关安装）

方法1：

使用pip指令，具体为：pip install robotframework-appiumlibrary，当然前提条件是安装好了Python2.7，并且也安装了pip工具。

方法2：

使用setup.py，可以官网页面右侧的下载链接下载后再进行解压缩，在命令行模式下进行到解压缩后所在的目录，最后执行指令python setup.py install，或者先把git安装好，然后依次执行三条指令：

git clone <https://github.com/jollychang/robotframework-appiumlibrary.git>  
cd robotframework-appiumlibrary  
python setup.py install

没有提示错误就是安装成功了。

1. Robot framework-selenium2library

资源地址：

<https://pypi.python.org/pypi/robotframework-selenium2library/1.5.0>

RF-seleniumlibrary 可以看做RF版的selenium 库，selenium （webdriver）可以认为是一套基于web的规范（API），所以，RF 、appium 等测试工具都可以基于这套API进行页面的定位与操作。

可以通过python 的pip工具包进行安装：

>pip install robotframework-selenium2library

1. RobotFrameWork
2. RF表格

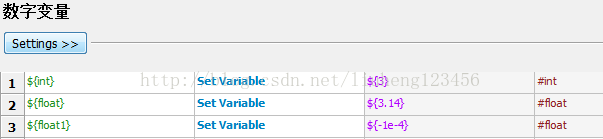
表格里的蓝色字体是库中的关键字，绿色字体是变量，黑色字体是系统自带关键字。

知识点：

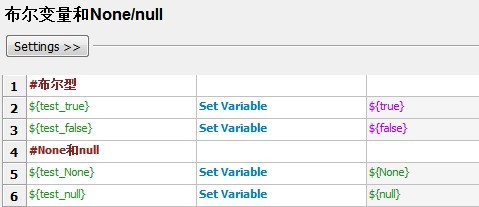
1. Robotframework 的基本使用，掌握buildin的方法，Log，Run keyword if以及数据类型等等
2. Uiautomatorview获取app元素
3. Appium 元素的定位和click,swipe点击操作。
4. RIDE快捷键

|  |  |  |  |
| --- | --- | --- | --- |
| **快捷键** | **用法** | **快捷键** | **用法** |
| F2 | 重命名 | Ctrl+鼠标悬浮 | 显示关键字信息 |
| F5 | 搜索关键字 | Ctrl+shift+空格 | 自动补全关键字 |
| F8 | 执行用例 | Ctrl+d | 删除行 |
| Ctrl+n | 创建新工程 | Ctrl+shift+d | 删除单元格 |
| Chrl+shift+f | 创建新测试套件 | Ctrl+shift+i | 插入单元格 |
| Chrl+shift+t | 创建新用例 | Ctrl+i | 插入行 |
| Chrl+shift+k | 创建新关键字 | Ctrl+# | 屏蔽代码 |
| Ctrl+ ↑ | 向上移动用例 | Ctrl+$ | 取消屏蔽 |
| Ctrl+↓ | 向下移动用例 | Ctrl+shift+s | 保存整个工程 |
| Ctrl+s | 局部保存 | Ctrl+L | 查看log |
| Ctrl+r | 查看report |  |  |

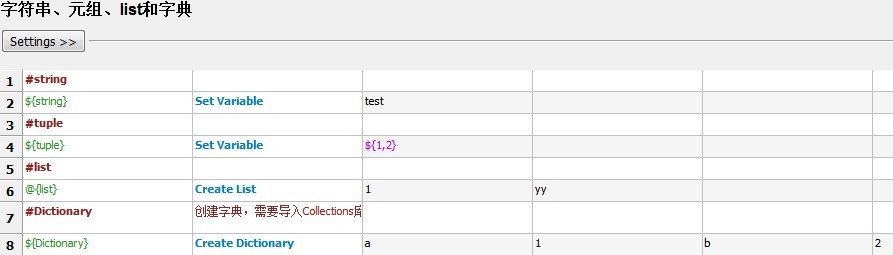
1. 数据类型
2. 数字变量



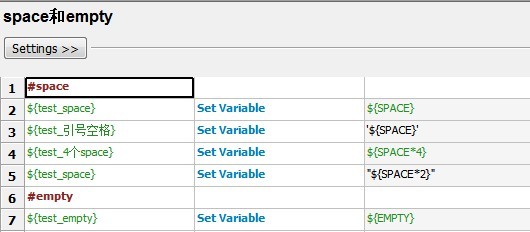
1. 布尔变量和None/null



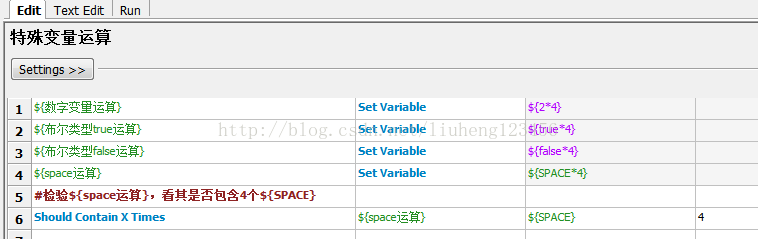
1. 字符串、元组、list和字典



1. space和empty

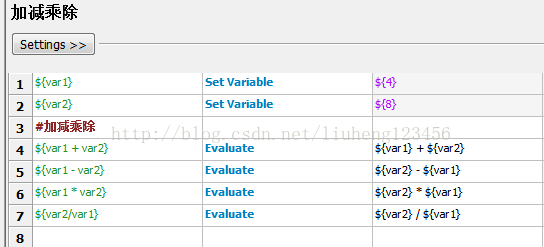


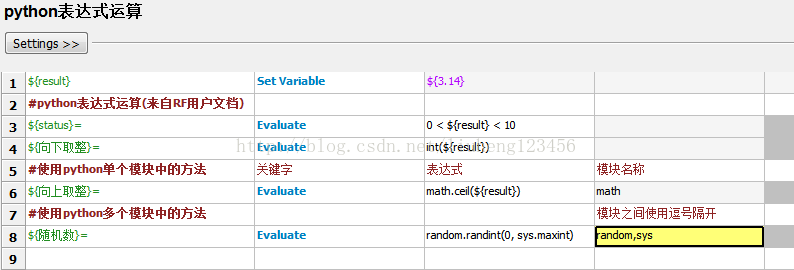
1. 变量运算与Evaluate
2. 特殊变量运算



1. Evaluate使用

函数释义：Evaluate是执行Python表达式，并返回执行结果





1. 控制流之if语句---Run Keyword If

在RF中if语句是使用关键字Run Keyword If来代替

函数释义：如果给出额判断条件满足，就执行给出的关键字

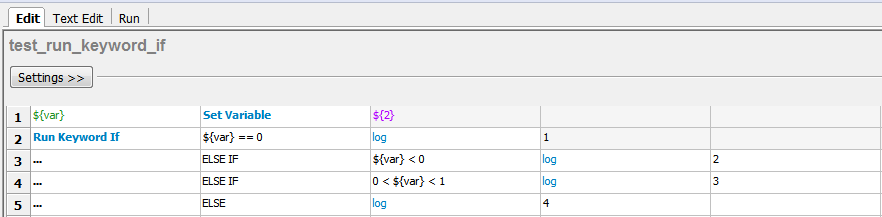
结构：

Run Keyword If 判断条件 其他关键字

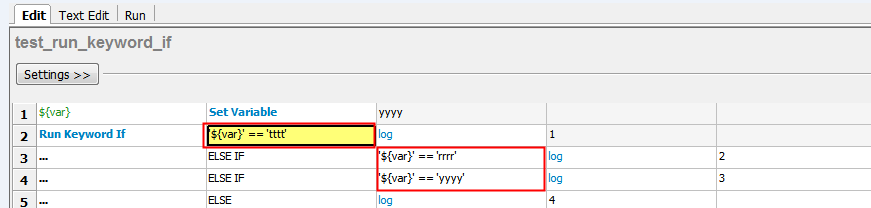
… ELSE IF 判断条件 其他关键字

… ELSE 判断条件 其他关键字

示例1：数字对比



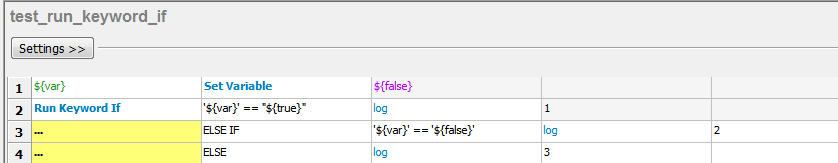
示例2：字符对比



注：

1. 变量${var}值为字符时，在判断条件中其需要加单引号或双引号，为‘${var}’或“${var}”
2. 在判断条件中，与变量${var}对比的字符常量也需要加单引号或双引号

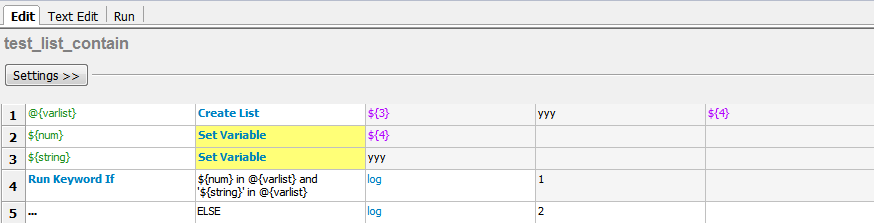
示例3：布尔类型对比



注意：

1. 变量${var}值为true或false时，在判断条件中其需要加单引号或双引号，为‘${var}’或者“${var}”。
2. 在判断条件中，与变量${var}对比的布尔值也需要加单引号或双引号，为‘${true}’或者“${true}”。

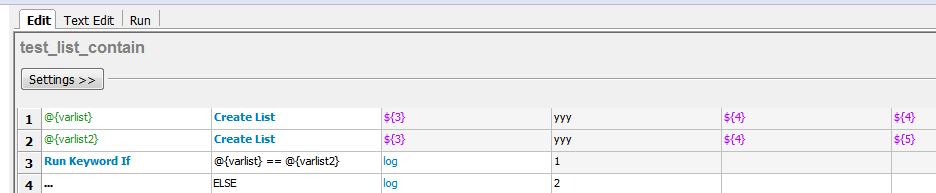
示例4：列表是否包含元素



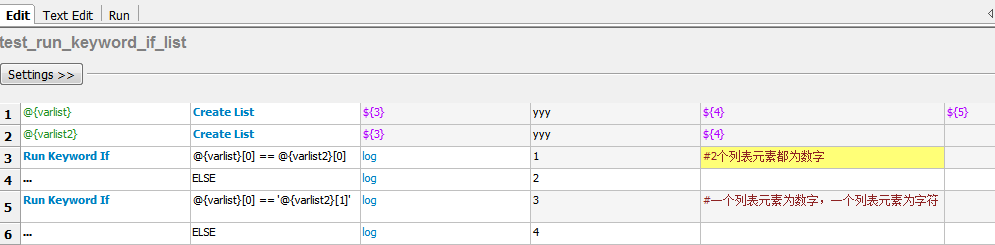
注意：

1. 变量${string}值为字符时，在判断条件中其需要加单引号或双引号，为‘${string}’或者“${string}”。

示例5：列表与列表是否相等对比



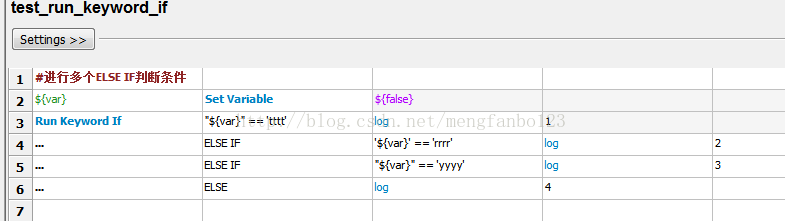
示例6：列表元素对比



注意：

1. 列表元素值(如上图中@{varlist2}[1])为字符时，在判断条件中其需要加单引号或双引号，如下图中@{varlist2}[1]，变为‘@{varlist2}[1]’或者“@{varlist2}[1]”。

示例7：执行多个判断条件



1. For循环

函数结构：

：For 变量 IN 序列（or列表）

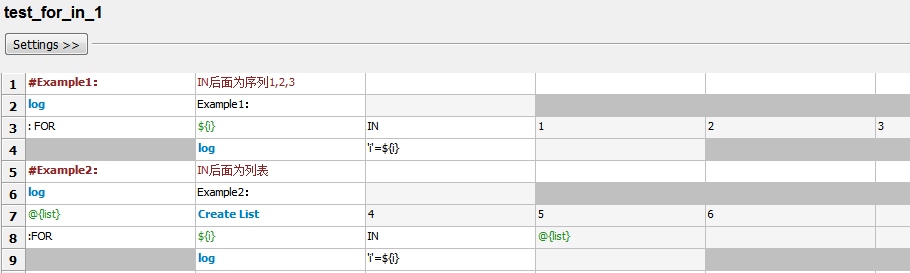
关键字 参数值

：For 变量 IN RANGE 循环限量

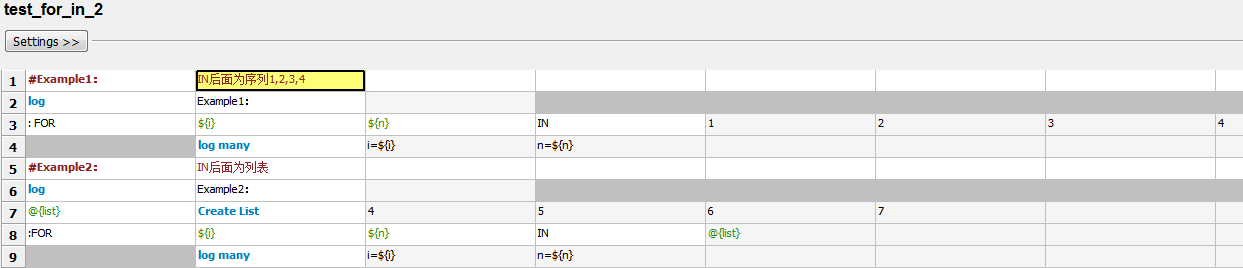
关键字 参数值

示例1：For IN 循环

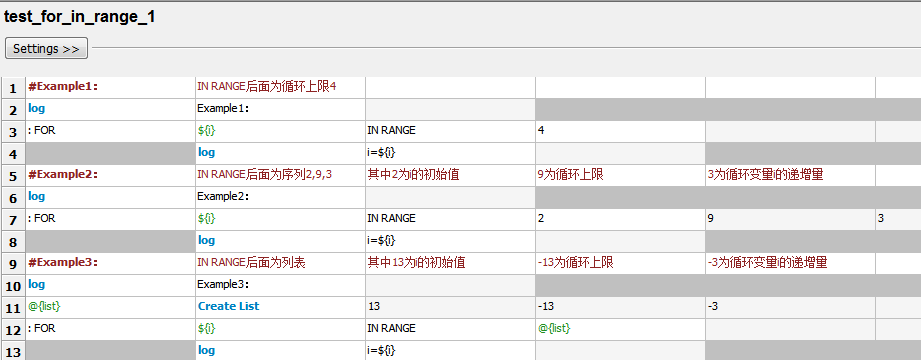
单个循环变量

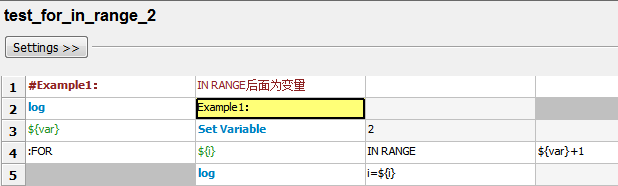


多个循环变量

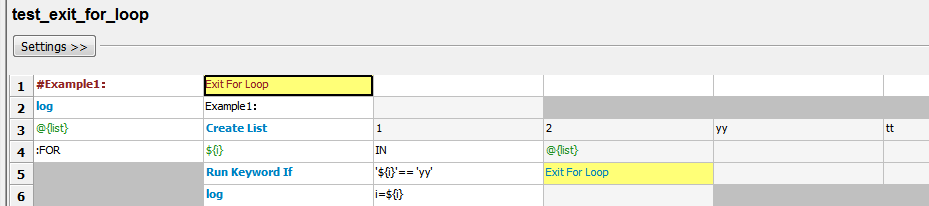


示例2：For IN RANGE循环





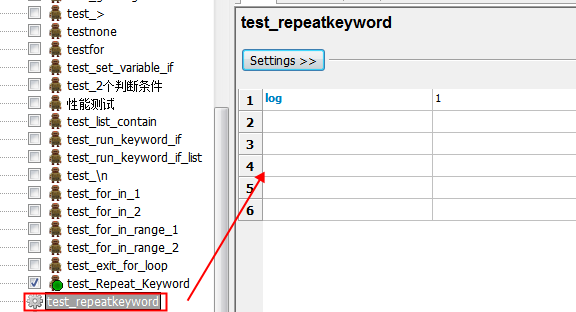
示例3：退出循环Exit For Loop



示例4：重复执行关键字

重复执行单个关键字

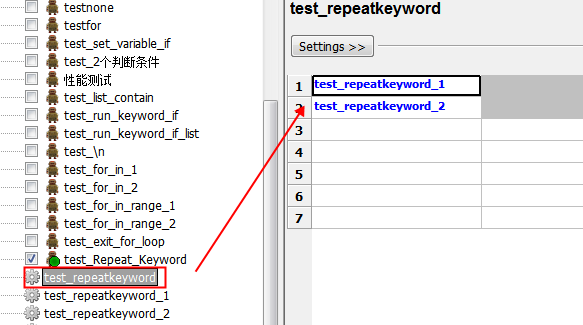
1. 设置需要重复的关键字





重复执行多个关键字

若需要重复执行多个关键字，可把多个关键字封装到一个关键字里，然后用例再重复执行



1. RF执行顺序

case：按照定义的上下位置顺序执行，可通过Ctrl+上下方向键进行调整

suite：包含suite的目录是按字母顺序执行，可通过添加如：01\_xxx.txt这样的前缀来控制顺序

1. RF截屏

RF提供了一个“Screenshot”库，在使用这个库之前，需要确认window环境已安装wxPython。当wxPython安装后，先引入该测试库：

Library Screenshot

Screenshot库有3个关键字：

Set Screenshot Directory ${CURDIR}：将图片保存在测试用力的目录下（载引入Screenshot库时设置路径同样可以：Library Screenshot &{CURDIR}）

Take Screenshot：截屏后将图片保存在报告中

Take Screenshot Without Embedding：截屏后，报告中保存截屏图片的链接

1. RF报告

Output  是和xunit兼容的一种xml格式的输出文件。可以被各种持续集成工具读取。所有被记录下来的信息其实都在里边。

Log是RF从output文件中提取并格式化的html文件，里边有所有keyword执行的细节情况。

Report也是从output里提取出来的html文件，里边以报表的形式给出了执行情况的统计信息。

1. Appium
2. 查看Android应用包名package和入口activity名称

方法1：使用aapt

aapt是sdk自带的一个工具，在sdk\builds-tools\目录下

命令行切换到aapt.exe目录执行：aapt dump badging { apk路径} ，例：aapt dump badging E:\apk\yixun.apk

运行结果：

package: name='com.idealsee.yixun'

launchable-activity: name='com.idealsee.ar.activity.ActivityTMain'

方法2：查看AndroidManifest.xml

1. 使用apktool反编译app：apktool.bat d E:\apk\yixun
2. 打开AndroidManifest.xml

manifest节点的package属性值是应用包名：

查找android.intent.action.MAIN和android.intetn.category.LAUNCHER对应的activity，该activity对应的android：name属性即是入口activity名称

1. 输入中文解决办法

在Open Application这个方法中，加入关键字**unicodeKeyboard**=${True}和**resetKeyboard**=${True}

1. Appium服务关键字

## Appium 服务关键字

<expand\_table>

|关键字|描述|实例|

|  |  |  |
| --- | --- | --- |
| automationName | 你想使用的自动化测试引擎 | ‘ppium’ (默认) 或 ‘Selendroid’ |
| platformName | 你要测试的手机操作系统 | ‘OS’, ‘Android’, 或 ‘FirefoxOS |
| platformVersion | 手机操作系统版本 | 例如： ‘7.1’, ‘4.4’ |
| deviceName | 使用的手机类型或模拟器类型 | iOS：‘Phone Simulator’, ‘iPad Simulator’, ‘iPhone Retina 4-inch’, （iOS上，此值必须是用‘instruments -s devices’得到的可使用的设备名之一）  Android：‘Android Emulator’, ‘Galaxy S4’, |
| app | ‘.ipa’ or ‘.apk’文件的本地绝对路径或者远程路径,或其‘.zip’。 Android上，若已指定‘app-package’和‘app-activity’，可不指定‘app’。 \*\*会与 ‘browserName’ 冲突\*\* | 比如‘/abs/path/to/my.apk’或‘http://myapp.com/app.ipa’ |
| browserName | 需测试的手机 web 浏览器名称。若是对应用进行自动化测试，此值应为空。 | iOS的 'Safari' ，Android 的'Chrome', 'Chromium', 或 'Browser'。 |
| newCommandTimeout | 设置命令超时时间，单位：秒。达到超时时间仍未接收到新的命令时 Appium 会假设客户端退出然后自动结束会话。 | 比如 ‘60’ |
| autoLaunch | Appium是否需要自动安装和启动应用。默认值‘true’ | ’true’, ‘false’ |
| language | (Sim/Emu-only) 设定模拟器 ( simulator / emulator ) 的语言。 | 如： ‘fr’ |
| locale | (Sim/Emu-only) 设定模拟器 ( simulator / emulator ) 的区域设置。 | 如： ‘fr\_CA’ |
| udid | 连接的物理设备的唯一设备标识 | 如： ‘1ae203187fc012g’ |
| orientation | (Sim/Emu-only) 在一个设定的方向模式中开始测试 | ‘LANDSCAPE’ (横向) 或 ‘PORTRAIT’ (纵向) |
| autoWebview | 直接转换到 WebView 上下文。 默认值 ‘false’ | ’true’, ‘false’ |
| noReset | 不要在会话前重置应用状态。默认值‘false’ | ’true’, ‘false’ |
| fullReset | (iOS) 删除整个模拟器目录。(Android) 通过卸载——而不是清空数据——来重置应用状态。在 Android 上，这也会在会话结束后自动清除被测应用。默认值 ‘false’ | ’true’, ‘false’ |

### **Android特有**

<expand\_table>

|关键字|描述|实例|

|  |  |  |
| --- | --- | --- |
| appActivity | 要从应用中启动的Android Activity 名，通常需在前添加‘.’。如：‘.MainActivity’ | ‘MainActivity’, ‘.Settings’ |
| appPackage | Android应用的包名 | 如’com.example.android.myApp’, ‘com.android.settings’ |
| appWaitActivity | 等待启动的 Android Activity 名称 | ’SplashActivity |
| deviceReadyTimeout | 设置等待一个模拟器或真机准备就绪的超时时间 | 如：5 |
| androidCoverage | 用于执行测试的 instrumentation 类。作为命令 ‘adb shell am instrument -e coverage true -w’ 的 ‘-w’ 参数。 | ‘com.my.Pkg/com.my.Pkg.instrumentation.MyInstrumentation’ |
| enablePerformanceLogging | (仅适用于 Chrome 和 webview) 开启 Chromedriver 的性能日志。 (默认 ‘false’) | ‘true’, ‘false’ |
| androidDeviceReadyTimeout | 等待设备在启动应用后准备就绪的超时时间。以秒为单位。 | 如：30 |
| androidDeviceSocket | 开发工具的 socket 名称。仅被测应用是使用 Chromium 内核的浏览器时需要。 socket 会被浏览器打开，然后 Chromedriver 把它作为开发者工具来进行连接。 | 如 ‘chrome\_devtools\_remote’ |
| avd | 需要启动的 AVD (安卓虚拟设备) 名称 | 如 ‘api19’ |
| avdLaunchTimeout | 以毫秒为单位，等待 AVD 启动并连接到 ADB 的超时时间。(默认值 ‘120000’) | 300000 |
| avdReadyTimeout | 以毫秒为单位，等待 AVD 完成启动动画的超时时间。(默认值 ‘120000’) | 300000 |
| avdArgs | 启动 AVD 时需要加入的额外的参数 | 如 ‘-netfast’ |
| useKeystore | 使用一个自定义的 keystore 来对 apk 进行重签名。默认值 ‘false’ | ’true’ or ‘false’ |
| keystorePath | 自定义 keystore 的路径。默认： ~/.android/debug.keystore | 如 ‘/path/to.keystore’ |
| keystorePassword | 自定义 keystore 的密码 | 如：foo |
| keyAlias | key 的别名 | 如 ‘androiddebugkey’ |
| keyPassword | key 的密码 | 如：foo |
| chromedriverExecutable | webdriver 可执行文件的绝对路径  (如果 Chromium 核心提供了对应的 webdriver， 应该用它代替 Appium 自带的 webdriver) | /abs/path/to/webdriver |
| ’autoWebviewTimeout | 以毫秒为单位，等待 Webview 上下文激活的时间。默认值 ‘2000’ | 如：4 |
| intentAction | 用于启动 activity 的 intent action。 (默认值 ‘android.intent.action.MAIN’) | 如 ‘android.intent.action.MAIN’, ‘android.intent.action.VIEW’ |
| intentCategory | 用于启动 activity 的 intent category。 (默认值 ‘android.intent.category.LAUNCHER’) | 如 ‘android.intent.category.LAUNCHER’, ‘android.intent.category.APP\_CONTACTS’ |
| intentFlags | 用于启动 activity 的标识 ( flags ) (默认值 ‘0x10200000’) | 如 ‘0x10200000’ |
| optionalIntentArguments | 用于启动 activity 的额外 intent 参数。 | 如 ‘--esn <EXTRA\_KEY>‘, ‘--ez <EXTRA\_KEY> <EXTRA\_BOOLEAN\_VALUE>‘ |
| stopAppOnReset | 在使用 adb 启动应用前停止被测应用的进程 ( process ) 。如果被测应用是被另一个应用创建的，当这个参数被设定为 false 时，允许另一个应用的进程在使用 adb 启动被测应用时继续存活。默认值 ‘true’ | ‘true’ 或 ‘false’ |
| unicodeKeyboard | 使用 Unicode 输入法。默认值 ‘false’ | ‘true’ 或 ‘false’ |
| resetKeyboard | 在设定了 ‘unicodeKeyboard’ 关键字的 Unicode 测试结束后，重置输入法到原有状态。如果单独使用，将会被忽略。默认值 ‘false’| | ‘true’ 或 ‘false’ |
| noSign | 跳过检查和对应用进行 debug 签名的步骤。只能在使用 UiAutomator 时使用，使用 selendroid 是不行。默认值 ‘false’ | ‘true’ 或 ‘false’ |
| ignoreUnimportantViews | 调用 uiautomator 的函数 ‘setCompressedLayoutHierarchy()’， 默认值 ‘false’ | ‘true’ 或 ‘false’ |

### **iOS特有**

<expand\_table>

|关键字|描述|实例|

|  |  |  |
| --- | --- | --- |
| calendarFormat | (Sim-only) 为iOS的模拟器设置日历格式 | 如 ‘gregorian’ (公历) |
| bundleId | 被测应用的 bundle ID。用于在真实设备中启动测试，也用于使用其他需要 bundle ID 的关键字启动测试。在使用 bundle ID 在真实设备上执行测试时，你可以不提供 ‘app’ 关键字，但你必须提供 ‘udid’ 。 | 如 ‘io.appium.TestApp’ |
| udid | 连接的真实设备的唯一设备编号 ( Unique device identifier ) | 如 ‘1ae203187fc012g’ |
| launchTimeout | 以毫秒为单位，在 Appium 运行失败之前设置一个等待 instruments 的时间 | 比如： ‘20000’ |
| locationServicesEnabled | (Sim-only) 强制打开或关闭定位服务。默认值是保持当前模拟器的设定 | ’true’ 或 ‘false’ |
| locationServicesAuthorized | (Sim-only) 通过修改 plist 文件设定是否允许应用使用定位服务，从而避免定位服务的警告出现。默认值是保持当前模拟器的设定。请注意在使用这个关键字时，你同时需要使用 ‘bundleId’ 关键字来发送你的应用的 bundle ID。 | ’true’ 或 ‘false’ |
| autoAcceptAlerts | 当 iOS 的个人信息访问警告 (如 位置、联系人、图片) 出现时，自动选择接受( Accept )。默认值 ‘false’。 | ’true’ 或 ‘false’ |
| autoDismissAlerts | 当 iOS 的个人信息访问警告 (如 位置、联系人、图片) 出现时，自动选择不接受( Dismiss )。默认值 ‘false’ |  |
| nativeInstrumentsLib | 使用原生 intruments 库 (即关闭 instruments-without-delay ) | ’true’ 或 ‘false’ |
| nativeWebTap | (Sim-only) 在Safari中允许"真实的"，非基于 javascript 的 web 点击 (tap) 。 默认值： ‘false’。注意：取决于 viewport 大小/比例， 点击操作不一定能精确地点中对应的元素 | ’true’ 或 ‘false’ |
| safariInitialUrl | (Sim-only) (>= 8.1) 初始化 safari 的时使用的地址。默认是一个本地的欢迎页面 | 如 ‘https://www.github.com’ |
| safariAllowPopups | (Sim-only) 允许 javascript 在 Safari 中创建新窗口。默认保持模拟器当前设置 | ’true’ 或者 ‘false’ |
| safariIgnoreFraudWarning | (Sim-only) 阻止 Safari 显示此网站可能存在风险的警告。默认保持浏览器当前设置。 | ’true’ 或者 ‘false’ |
| safariOpenLinksInBackground | (Sim-only) Safari 是否允许链接在新窗口打开。默认保持浏览器当前设置。 | ’true’ 或者 ‘false’ |
| keepKeyChains | (Sim-only) 当 Appium 会话开始/结束时是否保留存放密码存放记录 (keychains) (库(Library)/钥匙串(Keychains)) | ’true’ 或者 ‘false’ |
| localizableStringsDir | 从哪里查找本地化字符串。默认值 ‘en. lproj’ | en.lproj |
| processArguments | 通过 instruments 传递到 AUT 的参数 | ’ 如 ‘-myflag’ |
| interKeyDelay | 以毫秒为单位，按下每一个按键之间的延迟时间 | 如 ‘100’ |
| showIOSLog | 是否在 Appium 的日志中显示设备的日志。默认值 ‘false’ | ’true’ 或者 ‘false’ |
| sendKeyStrategy | 输入文字到文字框的策略。模拟器默认值：’oneByOne’ (一个接着一个) 。真实设备默认值：’grouped’ (分组输入) | ’oneByOne’, ‘grouped’ 或 ‘setValue’ |
| screenshotWaitTimeout | 以秒为单位，生成屏幕截图的最长等待时间。默认值： 10。 | 如：5 |
| waitForAppScript | 用于判断 "应用是否被启动” 的 iOS 自动化脚本代码。默认情况下系统等待直到页面内容非空。结果必须是布尔类型。 | 例如 ‘true;’, ‘target.elements().length > 0;’, ‘$.delay(5000); true;’ |

1. Appium客户端API

仅限于Python语言

锁定屏幕：driver.lock(5)

将app置于后台：driver.background\_app(5)

收起键盘：driver.hide\_keyboard()

启动Activity：在当前应用中打开一个 activity 或者启动一个新应用并打开一个 activity 。 \*只能在 Android 上使用\*

driver.start\_activity(‘com.example.android.apis’,’.Foo’)

打开通知栏：打开下拉通知栏\*只能在Android上使用\*

driver.open\_notifications()

检查应用是否已安装：driver.is\_app\_installed(‘com.example.android.apis’)

安装应用：driver.install\_app(‘path/to/my.apk’)

删除应用：driver.remover\_app(‘com.example.android.apis’)

模拟设备摇晃（Shake）：driver.shake()

关闭应用：driver.close\_app()；

启动（Launch）：根据服务关键字 (desired capabilities) 启动会话 (session) 。请注意这必须在设定 `autoLaunch=false` 关键字时才能生效。这不是用于启动指定的 app/activities ——你可以使用 `start\_activity` 做到这个效果——这是用来继续进行使用了 `autoLaunch=false` 关键字时的初始化 (Launch) 流程的。

driver.launch\_app()

重置（Reset）：driver.reset() \*（相当于卸载重装）\*

可用上下文：driver.contexts

当前上下文：driver.current\_context

切换到默认的上下文：driver.switch\_to.context(None)

获取应用的字符串：driver.app\_strings

给设备发送一个按键事件：driver.keyevent(176)

获取当前activity：driver.current\_activity \*（仅限于Android）\*

触摸动作（TouchAction）/多点触摸（MultiTouchAction）：生成触摸动作的接口

action=TouchAction（driver）

action.press(element=el,x=10，y=10).release().perform()

模拟滑动（Swipe）：

driver.swipe(startx=75,starty=500,endx=75,endy=0,duration=800)

模拟捏合（pinch）：driver.pinch(element=le)

放大屏幕（zoom）：driver.zomm(element=el)

滑动到某个元素（scroll to）：todo：python

从设备中拉出文件（pull file）：

driver.pull\_file ('Library/AddressBook/AddressBook.sqlitedb')

推送文件到设备中去（push file）：

data=“some data for the file”

path=“/data/local/tmp/file.txt”

drver.push\_file(path, data.encode(‘base64’))

设置：从这里你可以获取/设置 appium 的服务器设置

current\_settings = driver.get\_settings()

driver.update\_settings({"someSetting": true})

1. Android自动化测试常用命令
   1. **android**

android sdk : 打开SDK管理器。

android avd : 打开虚拟设备管理器。

android --help : 查看帮助信息。

* 1. **adb**

adb devices -l : 查看可用的android设备，真机和模拟器。

adb install -r xxx.apk : 向设备中安装并替换应用。

adb --help : 查看帮助信息。

* 1. **appium**

appium-doctor: 诊断appium安装环境是否正常。

appium -U, --udid : 连接指定UDID的真实设备。

appium -h : 查看帮助信息。

appium -v : 版本信息。

1. adb服务端口2037被占用，导致adb和appium无法工作

**症状1：**

命令行运行 adb 相关命令，提示如下：

adb server is out of date. killing...

ADB server didn't ACK

\* failed to start daemon \*

**症状2：**

启动appium服务失败，也报类似上面的错误。

**解决办法：**

由于adb的服务端口5037被占，导致adb启动失败。

查看是谁占用此端口，执行：**netstat -ano | findstr "5037"**

C:\Users\xxx>netstat -ano | findstr "5037"

TCP 127.0.0.1:5037 0.0.0.0:0   LISTENING    38516

TCP 127.0.0.1:5037 127.0.0.1: 45049 ESTABLISHED 38516

TCP 127.0.0.1:45049 127.0.0.1:5037 ESTABLISHED  35884

找到id为38516的程序，将其结束掉，执行：**tskill 38516**

也可以在任务管理器按进程ID进行查找、杀掉，一般为手机助手之类的应用。

1. 元素定位
2. Selenium API
3. 浏览器驱动

通过不同的浏览器执行脚本

|  |  |  |
| --- | --- | --- |
| Open Browser | Htpp://www.xxx.com | chrome |

浏览器对应的关键字：

|  |  |
| --- | --- |
| firefox | FireFox |
| ff |
| internetexplorer | Internet Explorer |
| ie |
| googlechrome | Google Chrome |
| gc |
| chrome |
| opera | Opera |
| phantomjs | PhantomJS |
| htmlunit | HTMLUnit |
| htmlunitwithjs | HTMLUnit with Javascipt support |
| android | Android |
| iphone | Iphone |
| safari | Safari |

1. 关闭浏览器

关闭浏览器：关闭当前的浏览器

|  |  |  |
| --- | --- | --- |
| Close Browser |  |  |

关闭所有浏览器：关闭所有打开的浏览器和浏览器缓存重置

|  |  |  |
| --- | --- | --- |
| Close All Browsers |  |  |

1. 浏览器最大化

使当前打开的浏览器全屏

|  |  |  |
| --- | --- | --- |
| Maximize Browser Window |  |  |

1. 设置浏览器宽、高

以像素为单位，第一个参数800表示宽度，第二个参数600表示高度

|  |  |  |
| --- | --- | --- |
| Get Window Size | 800 | 600 |

1. 文本输入

Xpath=//\* [@] ：表示元素定位，定位文本输入框。

|  |  |  |
| --- | --- | --- |
| Input Text | Xpath=//\* [@] | 输入信息 |

1. 点击元素

Xpath=//\* [@] ：表示元素定位，定位点击的元素。

|  |  |  |
| --- | --- | --- |
| Click Element | Xpath=//\* [@] |  |

1. 点击按钮

Xpath=//\* [@] ：表示元素定位，定位点击的按钮。

|  |  |  |
| --- | --- | --- |
| Click Button | Xpath=//\* [@] |  |

1. 注释

注释1：

|  |  |  |
| --- | --- | --- |
| Comment | 注释说明 |  |

注释2: 除了使用Comment 关键字进行注释外，Robot framework框架是基于python语言开发的，所以提供了python语言的注释“#”方式。

|  |  |  |  |
| --- | --- | --- | --- |
| # 注释说明 |  |  |  |

1. 固定时间休眠

Sleep表示执行到当前行固定休眠多长时间，以“秒”为单位。

42表示42秒；

1.5 表示1.5秒；

2 minutes 10 seconds 表示2分10秒。

|  |  |  |
| --- | --- | --- |
| Sleep | 42 |  |
| Sleep | 1.5 |  |
| Sleep | 2 minutes 10 seconds |  |

1. 等待元素出现在当前页面

Xpath=//\* [@] ：表示元素定位，这里定位出现的元素

42 ： 表示最长等待时间。

Error ： 表示错误提示，自定义错误提示，如：“元素不能正常显示”

|  |  |  |  |
| --- | --- | --- | --- |
| Wait Until Page Contains Element | Xpath=//\* [@] | 42 | error |

1. 获取title

获得当前浏览器窗口的title 信息。这里只获取title 是没有意义的，我们通常会将获取的title 传递给一个变量，然后与预期结果进行比较。从而判断当前脚本执行成功。

|  |  |  |
| --- | --- | --- |
| Get Title |  |  |

1. 获取文本信息

Xpath=//\* [@] ： 定位文本信息的元素。

|  |  |  |
| --- | --- | --- |
| Get Text | Xpath=//\* [@] |  |

1. 获取元素属性值

id=kw@name ： id=kw 表示定位的元素。@nam 获取这个元素的name属性值。

|  |  |  |
| --- | --- | --- |
| Get Element Attribute | id=kw@name |  |

1. cookie处理

获取cookie：获得当前浏览器的所有cookie

|  |  |  |
| --- | --- | --- |
| get cookies |  |  |

获得cookie值，Key\_name ： key\_name 表示一对cookie中key的name 。

|  |  |  |
| --- | --- | --- |
| get cookie value | Key\_name |  |

删除cookie：删除key为name 的cookie信息。

|  |  |  |
| --- | --- | --- |
| delete cookie | Key\_name |  |

删除所有cookies：删除当前浏览器的所有cookie。

|  |  |  |
| --- | --- | --- |
| delete all cookies |  |  |

添加cookie：添加一对cooke （key：value）

|  |  |  |
| --- | --- | --- |
| add cookie | Key\_name | Value\_name |

1. 声明变量

定义变量a为hello。

|  |  |  |
| --- | --- | --- |
| ${a} | Set Variable | hello |

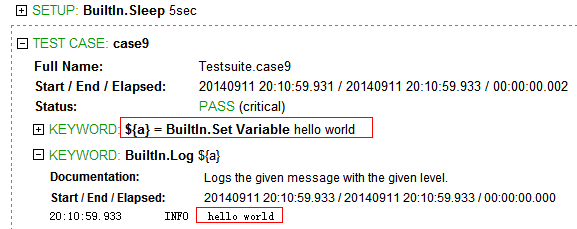
 定义变量a为hello ，b为world 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ${a} | ${b}= | Set Variable | hello | world |

1. 日志（输出）

在测试报告中输出a变量的值hello word。

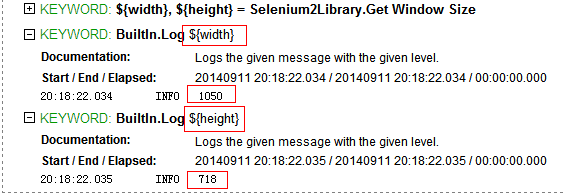
|  |  |  |
| --- | --- | --- |
| ${a} | Set Variable | Hello World |
| log | ${a} |  |



1. 获得浏览器窗口的宽、高

获得浏览浏览器窗口宽、高，通过log 将宽高，打印到报告中。

|  |  |  |
| --- | --- | --- |
| ${width} | ${height} | get window size |
| log | ${width} |  |
| log | ${height} |  |



1. 验证

Open Browser 通过chrome打开百度首页。

Get Title 获得浏览器窗口的titile ，并赋值给变量${title}

Should Contain 比较${title}是否等于“百度一下，你就知道”。

|  |  |  |
| --- | --- | --- |
| open browser | http://www.baidu.com | chrome |
| ${title} | Get Title |  |
| should contain | ${title} | 百度一下，你就知道 |



如果item1 不包含 item2 一次或多次，那么失败。

1. 表单嵌套

Select Frame 进入表单，Xpath=//\* [@] 表示定位要进入的表单。

Unselect Frame 退出表单。

|  |  |  |
| --- | --- | --- |
| Select Frame | Xpath=//\* [@] |  |
| Unselect Frame |  |  |

1. 下拉框选择

Xpath=//\* [@] 定位下拉框；

Vlaue 选择下拉框里的属性值。

|  |  |  |
| --- | --- | --- |
| Unselect From List By Value | Xpath=//\* [@] | vlaue |

1. If分支语句

 首先定义两个变量a ，b 分别为 2 和5 。

if 判断 a 大于等于1 ，满足条件log 输出 “a大于1 ”；

不满足上面的条件，接着else if 判断b小于等于5 ，满足条件log 输出 “b小于等于5”；

上面两个条件都不满足，else  log输出“上面两个条件都不满足”。

 备注：注意sele if 和else前面的三个点点点（...）

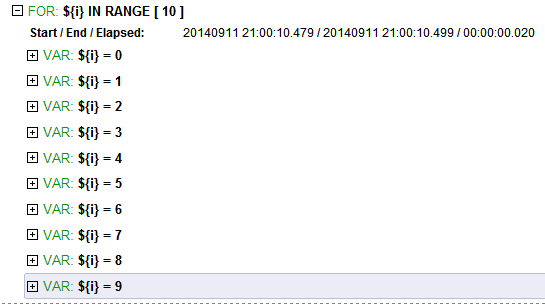
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ${a} | Set variable | 2 |  |  |
| ${b} | Set variable | 5 |  |  |
| run keyword if | ${a}>=1 | log | a大于1 |  |
| ... | ELSE IF | ${b}<=5 | log | b小于等于5 |
| ... | ELSE | log | 上面两个条件都不满足 |  |

1. for 循环语句

循环1

|  |  |  |  |
| --- | --- | --- | --- |
| :FOR | ${i} | in range | 10 |
|  | log | ${i} |  |

查看结果：



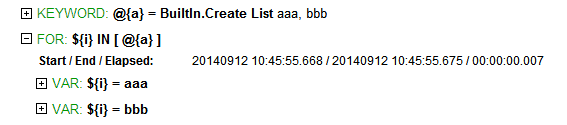
循环变量i 从0 到9 循环10次。

循环2

|  |  |  |  |
| --- | --- | --- | --- |
| @{a} | create list | aaa | bbb |
| :FOR | ${i} | in | @{a} |
|  | log | ${i} |  |

@{a} 定义为一个字符串列表。

通过in 可遍历非整型（in range）



说明：

Log 、if 分支，for 循环并非selenium关键字库的提供的方法，是由BuiltIn包提供。

1. 收到