|  |
| --- |
|  |
| Android Lettuce |
| 用中文写用例的Behavior-Driven |
|  |
| **虞科敏** |
| **2015/12/25** |

|  |
| --- |
|  |

[背景 4](#_Toc454374102)

[环境准备 5](#_Toc454374103)

[0. 前提条件 5](#_Toc454374104)

[1. 准备一台Ubuntu虚拟机 5](#_Toc454374105)

[2. 先不要开机, 先保证你的Ubuntu虚拟机可以上网 6](#_Toc454374106)

[3. 保证你的Ubuntu可以识别你的Android设备 6](#_Toc454374107)

[4. 准备你的用例撰写环境 6](#_Toc454374108)

[5. 准备你的用例执行环境 7](#_Toc454374109)

[6. 环境就绪 8](#_Toc454374110)

[入门 9](#_Toc454374111)

[创建你的项目 9](#_Toc454374112)

[场景文件: features/\*.feature 11](#_Toc454374113)

[对应到TestLink用例的约定 13](#_Toc454374114)

[日志系统 15](#_Toc454374115)

[在android lettuce中抓取log 15](#_Toc454374116)

[集成到Jenkins执行 17](#_Toc454374117)

[进阶 20](#_Toc454374118)

[安卓莴苣系统:目录和常用命令 20](#_Toc454374119)

[配置文件 22](#_Toc454374120)

[配套脚本 24](#_Toc454374121)

[场景模板:scenario outlines & Examples 26](#_Toc454374122)

[使用 场景模板(Scenario Outline) 26](#_Toc454374123)

[执行时指定场景或标签 27](#_Toc454374124)

[特性或场景钩子 30](#_Toc454374125)

[常用正则表达式 31](#_Toc454374126)

[多设备支持 33](#_Toc454374127)

[页面元素属性 34](#_Toc454374128)

[物理键名字和键码 38](#_Toc454374129)

[导出Automatorviewer需要的数据到本地 40](#_Toc454374130)

[图片比较工具 43](#_Toc454374131)

[基于坐标的录制工具 50](#_Toc454374132)

[预定义步骤参考 55](#_Toc454374133)

[预定义步骤之click(点击) 55](#_Toc454374134)

[预定义步骤之longclick(长按) 60](#_Toc454374135)

[预定义步骤之image(截图) 62](#_Toc454374136)

[预定义步骤之doubleclick(双击) 63](#_Toc454374137)

[预定义步骤之fs(File System,文件系统操作) 64](#_Toc454374138)

[预定义步骤之press(物理键和键码) 66](#_Toc454374139)

[预定义步骤之drag(拖动) 67](#_Toc454374140)

[预定义步骤之fling(快速滑动) 69](#_Toc454374141)

[预定义步骤之scroll(滚动) 71](#_Toc454374142)

[预定义步骤之swipe(滑动) 75](#_Toc454374143)

[预定义步骤之assert(断言) 79](#_Toc454374144)

[预定义步骤之settings(设置) 82](#_Toc454374145)

[一些有意思的Q&A 85](#_Toc454374146)

[Jenkins使用 89](#_Toc454374147)

[ubuntu下Jenkins环境搭建 89](#_Toc454374148)

[Jenkins配置基于角色的项目权限管理 90](#_Toc454374149)

[Jenkins项目管理 96](#_Toc454374150)

[附录 99](#_Toc454374151)

[在Windows环境下安装Android Lettuce的支持环境 99](#_Toc454374152)

[Ubuntu虚拟机不能识别Android设备的解决方法 100](#_Toc454374153)

[Ubuntu adb不能识别Android设备的解决方法 102](#_Toc454374154)

# 背景

最近项目空闲期间一件比较大的事情就是整合uiautomator python wrapper和lettuce, 放在ubuntu环境中得到更好的中文支持.

我们的这个虚拟项目取名为Android Lettuce, 意为使用Lettuce来执行Android系统的UI测试. 项目中文名就是"安卓莴苣", 这是一个可以用中文来撰写用例, 整合进CI的用户界面测试框架.

希望本虚拟项目可以降低撰写Android应用测试脚本的入门门槛, 帮助更多测试同学将自己的重复工作自动化起来, 切实提高工作效率.

# 环境准备

## 0. 前提条件

你的Android设备应该是4.0以上的操作系统, 4.3以上更好.

你的宿主机, 即你的PC电脑, 配置至少应该4G内存, 8G更好. 安装Ubuntu虚拟机环境很费内存.

## 1. 准备一台Ubuntu虚拟机

有2种安装路径可供选择:

1. 使用我提供的已经环境就绪的虚拟机镜像

由于Ubuntu系统上还有较多的新程序需要安装，脚本程序需要改动，环境配套文件需要下载，为了方便, 我把这些内容都准备到了一台虚拟机中，请从指定位置获得Ubuntu虚拟机镜像导入你的VirtualBox虚拟机.

\\10.120.10.100\ckt\_cd\_share\VAS\4.Test\android\_lettuce\vm

VirtualBox-4.2.6-82870-Win

=> 我使用的VirtualBox虚拟机安装程序. 我的机器为win7 64位版本, 如果此安装程序你不能使用, 可能需要自己去下载适合自己的版本.

myUbuntu.ova

=> 我为大家准备的AndroidLettuce执行准环境

2. 自助安装

Ubuntu虚拟机上我替大家准备了很多开发环境需要的软件, 比如安装了uiautomator python wrapper, lettuce, adb, vsftpd, sshd, snv等, 如自助安装需要你自己动手一遍, 可能会有些繁琐:

- 安装Linux虚拟机 (推荐Ubuntu,手册中是以Ubuntu为环境提供的命令)

- lettuce (<http://pythonhosted.org/lettuce/> , BBT框架)

- uiautomator python wrapper (<https://github.com/xiaocong/uiautomator> , uiautomator jsonrpc wrapper)

- adb (安卓环境)

- vsftpd, sshd (android lettuce环境所需的服务)

- svn/git (内网svn, 外网github, 如何使用请自己学习, 不在本文档范围之内)

说明: 由于本环境最开始以虚拟机镜像发布为主, 安装和更新脚本没有考虑自助安装的情况, 软件安装请尽量保留缺省安装地址.

## 2. 先不要开机, 先保证你的Ubuntu虚拟机可以上网

机器如果想要能更新有些软件, 就应该有个合法的局域网ip地址, 并且可以访问外网.

首先, 保证你的虚拟机MAC地址是唯一的不冲突的

你的虚拟机处于关闭状态, 在你的虚拟机上选择:

网络, 网卡1

连接方式, 桥接网卡

高级, MAC地址, 乱改一个吧

然后, 把你这个MAC地址告诉IT, 告诉他你的虚拟机也需要上网请求开通权限. 等一切搞好后, 你的Ubuntu就是一个可以正常上网工作的Ubuntu啦.

请注意!

你如果不想更新软件, 只是使用安卓莴苣, 以上请开通权限注册MAC地址的过程可以暂时略去, 以后需要再申请开通也可以.

开机, 登录, (账号, 密码请想我索要), 进去以后sudo –i切换到root

## 3. 保证你的Ubuntu可以识别你的Android设备

首先, 你的ubuntu能否识别你的手机usb设备吗?

root@ckt:/home/ckt# lsusb

Bus 001 Device 006: ID 2a47:0c02 ==> 我的手机设备. 插上手机后能多出你的手机设备行, 就标明设备能认识啦.

Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet

Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

如果本处碰到困难, 请参考附录相关方法

然后, 你的ubuntu中adb能识别你的手机吗?

sudo -i ==> 请记得adb是需要root权限才能执行的!!!!!!

adb devices ==> 显示你的手机设备.

如果本处碰到困难, 请参考附录相关方法

## 4. 准备你的用例撰写环境

当然, 你可以在Ubuntu系统下面直接编写你的测试用例.

我更推荐的是, 利用Ubuntu 系统下的ftp服务, 你可以在你的Windows机器(相信很多同学的工作用机还是windows平台)上直接编写Ubuntu机器上的用例. 我已经在Ubuntu虚拟机中配置了ftp服务, 开机后就会自动启动ftp服务.

如何设置, 可以参考我的环境

*我使用的是Notepad++.*

*首先,打开ftp插件.*

*以此选择: 插件, NppFtp, show NppFtp window*

*插件上的齿轮图标, Profile Settings*

*你可以在你面添加你的连接.*

*连接配置好后,点击插件上第一个连接按钮, 选择你的profile就可以连接到远端ftp服务器.*

如果你使用的其他编辑器, 你就要看一下你的编辑器是否支持通过ftp编辑远程机器上的文件. 现在, 你可以开始用这个编写你的脚本, 一定记住: 编码格式 utf-8无BOM

## 5. 准备你的用例执行环境

当然, 像编辑一样, 你也可在Ubuntu 系统GUI界面窗口中直接执行Lettuce测试用例.

我更推荐的是, ssh登录ubuntu机器, 这样在windows下执行会更方便.

因为安卓莴苣项目是为了中文编写用例, ssh登录终端需要支持中文, 才能正常执行测试.

[设置ssh登录工具中文支持, 可以参考我的设置](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_shell_tool_for_chinese)

*本处只针对常用的ssh工具: SecureCRT说明, 其他shell工具可以找到类似配置进行中文显示的设置*

*注意, 我提供的Ubuntu虚拟机已经配置, 可以跳过以下Linux的配置.*

*首先, sshd端要支持中文. 修改远程linux机器的配置*

*[root@rhel ~]#vi /etc/sysconfig/i18n*

*把LANG改成支持UTF-8的字符集*

*如： LANG=”zh\_CN.UTF-8″ 或者是 LANG=”en\_US.UTF-8″ 本文修改为后者*

*其次, SecureCRT需要设置中文显示支持, 避免中文显示乱码*

*问题：在SecureCRT终端中文件内容中文显示乱码，但是直接使用linux系统terminal打开此文件时中文显示正常，确诊问题出现在客户端即SecureCRT的显示问题*

*解决方法：*

*修改Secure CRT的Session Options*

*Options->Session Options->Appearance->Font->新宋体；*

*字符集：中文GB2312;*

*Character encoding 为UTF-8*

## 6. 环境就绪

到此, 你已经为我们的安卓莴苣准备好了环境, 可以开始你的新旅程啦...

不过开始前, 有几点提请大家注意, 以免造成阅读上的误会:

后面提到的一些脚本, 特别是内容更新是需要内网svn仓库支持的. 对于在外网中的使用, 我准备提供另一套基于github仓库的管理脚本, 但是缺乏充分的验证, 可能存在较多问题;

后面提到的一些命令, 比如al\_new等是在虚拟机镜像中我手工添加的link, 并没有通过更新脚本强制添加到系统中. 你如果觉得好用, 可以自己运行allink.sh添加.

# 入门

## 创建你的项目

为大家准备了命令来快速创建你的项目

/home/ckt/lettuce/tools# ./svnlettuce.py -n myproj

A steps/fs.py

A steps/regex.py

A steps/press.py

A steps/drag.py

A steps/scroll.py

A steps/fling.py

...

Checked out revision 604.

此命令的提供命令链接al\_new, 可以在任意目录下执行. 项目都创建在下文所提及的位置.

### 目录结构

在/home/ckt/lettuce下已经为你创建了myproj项目 (你可以通过cdlettuce快速切换到lettuce目录)

/home/ckt/lettuce/myproj# ls /home/ckt/lettuce/myproj/\* -l

/home/ckt/lettuce/myproj/features: =>　这是你测试features的主目录, 这个目录必须名为"features", 这是lettuce框架的约定俗成, 不要变动!

total 0

/home/ckt/lettuce/myproj/steps: => 这是我们为你准备的预定义步骤

total 80

-rw-r--r-- 1 root root 5394 8月 22 11:43 allog2.py

-rw-r--r-- 1 root root 5394 8月 22 11:43 allog3.py

-rw-r--r-- 1 root root 5394 8月 22 11:43 allog.py

-rw-r--r-- 1 root root 4589 8月 22 11:43 assert2.py

-rw-r--r-- 1 root root 4589 8月 22 11:43 assert3.py

-rw-r--r-- 1 root root 4420 8月 22 11:43 assert.py

-rw-r--r-- 1 root root 9413 8月 22 11:43 click2.py

-rw-r--r-- 1 root root 9413 8月 22 11:43 click3.py

-rw-r--r-- 1 root root 9172 8月 22 11:43 click.py

...

在features目录下你可以撰写这些测试文件. 这是你存放你的测试套件/测试用例/自定义步骤的地方.

root@ckt-VirtualBox:/home/ckt/lettuce/myproj/features# ls -l

total 0

-rw-r--r-- 1 root root 0 7月 29 20:19 demo.feature

=>

安卓莴苣里面我们叫它为场景文件. 这类文件必须以.feature为后缀.

在xxx.feature文件里实现你的具体测试.

你大部分的测试, 可以使用我们提供给你的预定义步骤在这个文件里编写.

预定义步骤可以使用工具showsteps.py查看.

-rw-r--r-- 1 root root 0 7月 29 20:20 mystep.py

=> 安卓莴苣里面我们叫它为步骤文件.

本目录下所有xxx.py都被视为自定义步骤, 会被安卓莴苣加载, 可以供你设计一些专属于你自己的步骤.

如何自定义步骤属于比较高级的教程, 会在比较靠后的wiki中介绍.

-rw-r--r-- 1 root root 848 8月 22 11:43 callback.py

=>

安卓莴苣里面我们叫它为回调文件.

本文件中定义了在执行过程中特定的时间点你希望框架替你执行的函数.

你可以定义你想在feature/scenario前后想要执行的操作, 请按说明实现本文件中的函数.

## [场景文件: features/\*.feature](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Introduction2)

一个最简单的场景文件

language:zh-CN

特性: 演示莴苣交互

为了和莴苣交互

作为初学者

我们实现此函数

背景:

打开计算机应用

场景: 点击

# 在计算器里

点击页面坐标【540,960】

安卓莴苣优先考虑中文写作用例, 第一行请为"language:zh-CN", 意为用简体中文书写用例.

先解释一些概念:

关键字是不能自由发挥的, 不可变动.

1. 特性: xxxxxx (关键字＂特性:＂)　－　可以理解为测试套件，即你这一组用例测试的功能特性是什么．

后面一段文字＂为了和．．．实现此函数＂　－　特性的描述，提供比名字更丰富的描述性内容

2. 背景:(关键字＂背景:＂)　－　每个场景执行的背景，背景定义的步骤会在每个场景之前被执行．　可以理解为前提条件．

3. 场景: xxxxxx　(关键字＂场景:＂或"场景模板:") - 可以理解为测试用例, 每一个场景就是一个个具体的测试用例, 里面包含步骤和预期.

场景模板为比较高级的功能, 在以后的wiki中介绍.

4. 步骤(非关键字) - 场景后面接着的就是用例中的执行语句, 里面包含执行步骤和预期.

步骤来自预定义或者自定义步骤, 如果语句不能匹配, 将不会执行, 会出错退出.

5. 场景模板 & 场景集|例如 (关键字＂场景模板＂ 和 ＂场景集:＂或＂例如:＂) - 可以理解为数据集, 多组数据应用于同一个场景, 相似的数据驱动同一个用例.

场景集为比较高级的功能, 在以后的wiki中介绍.

6. 注释 - 以#开头的行为注释, 不会被莴苣解释执行.

步骤的写法要求：

在步骤中请将步骤的顺序通过数字标注。

例如：

场景:点击

1、 点击页面坐标【540,960】

2、 点击页面坐标【54,96】

3、 点击页面坐标【22,33】

标注了序号的好处是，在生成报告后会自动排列每个场景中的步骤，从而与脚本的执行顺序保持一致。

## [对应到TestLink用例的约定](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Introduction3)

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Android_lettuce_cases.jpg)

约定1. 一个直接包含用例的目录对应一个feature文件, 请用序号表达此目录.

比如:　1-1-1.feature, 1-1-2.feature

约定2. 特性名字请使用格式: [目录1名-目录2名-目录3名], 尽量使用目录全名. 如果名字太长, 可以序号代替.

比如: "1取景模式-1普通录像-1", "1取景模式-1普通录像-3显示已经录制的时间"

约定3. 场景名字, 请直接使用用例全名.

比如: 1-1-1-1用例对应的场景名"1点击屏幕开始录像"

约定4. 用例的关键字, 请填写到场景对应的tag.

比如:　@F1

约定5. 场景中的每一步请标注清楚序号

错误:

设置路径【/sdcard/DCIM/Camera】下【3gp】类型文件数检查点

点击页面坐标【960,570】

应该看不到文本【"480P-30"】

正确:

1) 设置路径【/sdcard/DCIM/Camera】下【3gp】类型文件数检查点

2) 点击页面坐标【960,570】

3) 应该看不到文本【"480P-30"】

约定6. 场景中的步骤是匹配到行尾, 前半句可以体现出TestLink中的步骤描述.

即只要保证语句结束部分能正确匹配定义步骤即可, 可以适当在语句前半段加一些说明和解释, 提高用例的可读性.

比如: 2) 按拍照键拍照: 点击页面坐标【960,570】

其中后半句"点击页面坐标【960,570】"可以正确匹配定义步骤, 前半句"按拍照键拍照: "就是为了提高可读性添加的说明.

## [日志系统](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-08-21%E3%80%91lettuce%E6%8A%A5%E5%91%8A%E4%BD%BF%E7%94%A8%E6%96%B9%E6%B3%95)

在android lettuce中抓取log

如果使用的是我统一打包的虚拟机，可在/home/ckt/lettuce/android\_lettuce.conf 路径找到相应的配置文件。

* log可配置文件 android\_lettuce.conf

其中包含了很多可配置的内容，如[phone]、[image]、[allog]以及[logtype]或更多。 log主要用到的两个部分(section)为[allog]、[logtype]

* log基本逻辑

1. 启动抓取log进程，此框架大部分log(以logcat、top为例)采用进程的方式在后台不断的更新log内容。
2. 执行一条抓取log的命令将log的信息保存在用户定义的路径下，记为“公共log”。
3. 每当一条用例执行前，自动清空“公共log”中的内容。
4. 用例执行······ing
5. 用例执行结束，自动将“公共log”中的内容拷贝至项目指定的文件夹(用户可配置)下。
6. 执行完所有用例后，杀掉“公共log”抓取进程。

* 解析conf文件

android\_lettuce.conf主要包含以下内容，蓝色部分为log涉及到的内容。

**[phone]**

is\_only\_one\_device = yes

device1 = CKT Z

device2 = HIKe 868

device3 = HIKe 838

**[image]**

is\_compared = yes

**[allog]**

is\_logged = yes #log抓取开关，yes为开启log抓取，no为关闭log抓取

dir=allog #存放log文件的目录名称，此处为allog

logpath\_record\_file=logpath.rec #中间产出文件，存放所有log文件路径

**[logtype]**

logcat = adb shell logcat \*:W #抓取logcat log的命令，格式logtype(logcat)= command(adb shell logcat \*:W)

logcat\_file = /tmp/\_\_logcat.log #logcat类型log的公共文件存放位置，格式logtype\_file(logcat\_file)=filepath(/tmp/\_\_logcat.log)

logcat\_kill = adb shell kill -9 #杀掉logcat抓取的进程,格式logtype\_kill(logcat\_kill)=command(adb shell kill -9)

top = adb shell top -m 10 #类似以上几点，所有命令为adb shell开头的即为手机操作，不含adb shell即为pc端操作

top\_file = /tmp/\_\_top.log

top\_kill = adb shell kill -9

top2 = top -d 3 -b

top2\_file = /tmp/\_\_top2.log

top2\_kill = kill -9

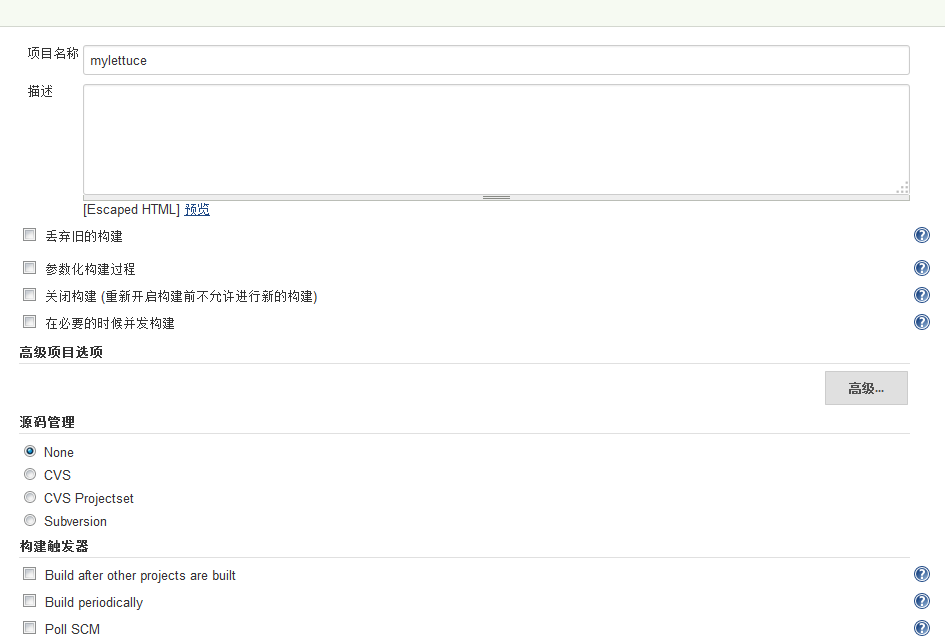
## [集成到Jenkins执行](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-08-22%E3%80%91lettuce%E5%9C%A8Jenkins%E4%B8%AD%E7%9A%84%E4%BD%BF%E7%94%A8)

### 构建自己的lettuce项目

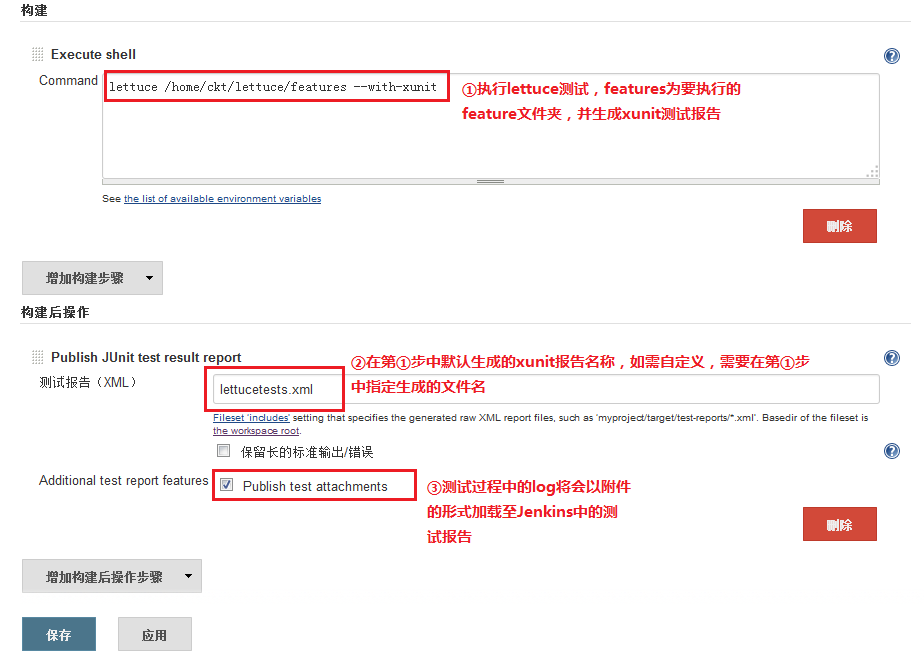
1. 需要一台配置好的提供Jenkins服务虚拟机。
2. 在虚拟机中启动Jenkins服务，使用命令service jenkins start
3. 在任意浏览器中输入Jenkins的地址, [http://ip:port](http://ip:port/) 如<http://10.120.2.242:8888/>

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Create_project.jpg)

* 在项目配置中填入相应的信息

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Lettuce-config1.jpg)

* 在构建中填入执行命令。以下为执行features文件夹中的测试，并生成xunit测试报告。

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Lettuce-config2.jpg)

* 进入project后点击左侧立即构建，即可开始测试（注意：将手机切换至虚拟机连接）
* 在Build History中可查看,红色失败，蓝色通过

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Lettuce-build.jpg)

* 测试结果可一层层深入查看，如下为一个测试用例的内容。

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Lettuce-result.jpg)

# 进阶

## [安卓莴苣系统:目录和常用命令](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Directories)

### 项目目录

1. allog - 存放本次执行各用例的各种日志文件, 以文件名标识:　feature\_scenario\_....log
2. captures - 存放本次执行各用例的截图, 以目录结构组织: feature/scenario/xxx.png, xxx为feature文件中截图名字
3. features - 存放feature定义文件/自定义步骤/自定义的回调函数
4. images - 存放各用例的预期图片, 以目录结构组织: feature/scenario/xxx.png, xxx为feature文件中截图名字
5. steps - 存放预定义步骤

### 常用命令

#### 进入目录

1. cdlettuce 进入lettuce工作主目录
2. cdlab1 进入lab1项目, 这是我保留给大家的演示项目
3. cdtools 进入常用工具目录
4. cduiauto 进入uiautomator wrapper的python代码目录
5. cdlettucelib 进入lettuce的python代码目录
6. cdjenkinslettuce 进入jenkins的My\_Android\_Lettuce的工作空间目录

#### 安卓莴苣系列命令

al\_xxx系列命令可在任意目录执行. al, android lettuce的缩写.

如果你的环境中还没有某命令, 请执行添加命令:

/home/ckt/lettuce/tools/scripts/allink.sh

### al\_image

操作某project下的图片相关目录captures/images.

常用选项:

-u 把captures下的内容更新到images

--cc 清空captures目录

--ci 清空images目录

Usage: myimages.py [options] project

myimages.py -h|--help for more help

### al\_imgtool

启动image辅助工具. 请在图形界面环境下执行, 比如X窗口环境.

### al\_new

创建新项目project.

在新创建的项目下, 将准备好预定义步骤; 在features下准备好空白features/callback/mysteps模板.

Usage: update\_project.sh <project>

### al\_steps

显示项目project下的预定义步骤.

常用选项:

-s STEPPATH 如果project没有指定, 那么显示STEPPATH下的预定义步骤.

-d 如果1.project没有指定, 2. -s没有指定, 那么使用android\_lettuce.conf/project/current的值作为项目名, 显示预定义步骤

Usage: showsteps.py [options] project

showsteps.py -h|--help for more help

### al\_update

更新项目project的svn内容.

Usage: update\_project.sh <project>

### al\_config

打印或使用vi编辑的安卓莴苣的配置文件.

Usage: myconfig.sh cat|vi

- cat: show config using cat command

- vi: edit config using vi command

### al\_recorder

启动安卓莴苣基于坐标步骤录制工具. 请在图形界面环境下执行, 比如X窗口环境.

## [配置文件](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Configuration)

### 安卓莴苣的配置文件

安卓莴苣系统缺省的配置文件是: /home/ckt/lettuce/android\_lettuce.conf.

你也可以在/usr/local/lib/python2.7/dist-packages/uiautomator-0.1.30-py2.7.egg/uiautomator/myConfig.py中修改文件的路径. 请不要轻易修改此配置文件定义, 我故意把它藏在代码中, 而不是用配置形式暴露出来.

在tools/config.demo目录下保留有配置文件样例.

### 配置项

[phone]

is\_only\_one\_device = no # yes, 只有一个设备. 设备1/2/3名字指定无用.

device1 = CKT Z # 设备1名字.

device2 = HIKe 828 # 设备2名字.

device3 = HIKe 838 # 设备3名字.

[image]

is\_compared = yes # 是否执行图片比对(captures vs. images下对应内容). 收集预期图片时候设置为no, 这样比较语句将只执行截图, 不进行对比.

[project]

basepath = /home/ckt/lettuce/

current = lab2

[allog]

is\_logged = yes

dir=allog

logpath\_record\_file=logpath.rec

[logtype]

logcat = adb shell logcat \*:W

logcat\_file = /tmp/\_\_logcat.log

logcat\_kill = adb shell kill -9

top = adb shell top -m 10

top\_file = /tmp/\_\_top.log

top\_kill = adb shell kill -9

## [配套脚本](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Utility_for_Ubuntu_Environment)

### showsteps.py

列出全部的预定义步骤

运行:

cd /home/ckt/lettuce/lab1/tools

./showsteps.py

或者

/home/ckt/lettuce/lab1/tools/showsteps.py

/home/ckt/lettuce/tools# ./showsteps.py -h

Usage: showsteps.py [options]

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-s STEPPATH show pre-defined steps under directory path

### svnlettuce.py

同步svn中的预定义步骤/uiautomator包装器的扩展

使用:

执行前请更新到最新版本 svn update 或者 svn update --username yourname --password yourpassword

更新你的项目lab1下的steps

./svnlettuce.py -u lab1

更新你的tools内容

./svnlettuce.py -t

更新你的lettuce/uiautomator代码

更新前请先更新你的tools内容

./svnlettuce.py -m

创建新的项目lab2

./svnlettuce.py -n lab2

为了更新你的测试环境到最新版本, 请依次执行以下命令

./svnlettuce.py -t

./svnlettuce.py -m

./svnlettuce.py -u [your project]

/home/ckt/lettuce/tools# ./svnlettuce.py -h

Usage: svnlettuce.py [options] arg1

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-t update tools

-u UPDATE, --update=UPDATE

update the project xxxxx

-a ACCOUNT, --account=ACCOUNT

svn account

-p PASSWORD, --password=PASSWORD

svn password

-n NEW, --new=NEW create new lettuce project

-m, --modify modify lettuce/uiautomator

## [场景模板:scenario outlines & Examples](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-08-21%E3%80%91lettuce%E8%BF%9B%E9%98%B6%E4%BD%BF%E7%94%A8)

使用 场景模板(Scenario Outline)

当我们一个特性(feature)文件里有多个场景(scenario)的时候，而且每个场景(scenario)的步骤都差不多，只是测试的数据不同。在这种情况下我们就可以使用场景模板(Scenario Outline),定义一个场景(scenario)，然后把测试数据参数化就可以了。 xxx.feature文件编写如下：

language: zh-CN

特性: 加法

使用【场景模板】的方式执行多条用例

场景模板: 将两个数相加

1、点击文本【"<数值\_1>"】

2、点击文本【"<按钮\_1>"】

3、点击文本【"<数值\_2>"】

4、点击文本【"<按钮\_2>"】

4、应该可看到文本【"<结果>"】

例如:

| 数值\_1 | 按钮\_1 | 数值\_2 | 按钮\_2 | 结果 |

| 1 | + | 2 | = | 3 |

| 5 | + | 6 | = | 11 |

| 7 | + | 8 | = | 15 |

解析：

1. 语句必须已经定义,否则会提示"This step must be implemented",步骤需要定义。
2. 将想要参数化的地方写成例如"<数值\_1>", "<按钮\_1>", "<数值\_2>", "<按钮\_2>"以及"<结果>"
3. 当执行每一个场景(Scenario)时, 对应名称的参数会调用【例如】表中对应的数值。
4. 【例如】table中的每一行就代表一条用例。

## [执行时指定场景或标签](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Scenario_Tag)

### 指定场景Scenario

lettuce执行时候可以使用 -s 或 --scenarios=SCENARIOS来指定要执行的场景.

#### 场景定义

场景在安卓莴苣中相当于TestLink中的用例, 为具体的功能点测试组织单位.

安卓莴苣中以场景的自然序列号代表某一场景. 1代表feature中的第1个场景, 2代表第2个, 以此类推.

场景之间使用英文","分隔.

#### 指定场景

例如, 特性文件:

language:zh-CN

特性: 计算器

为了和莴苣交互

作为初学者

我们实现此函数

场景: 加法

点击文本【"7"】

点击文本【"+"】

点击文本【"8"】

点击文本【"="】

场景: 减法

点击文本【"8"】

点击文本【"-"】

点击文本【"7"】

点击文本【"="】

1. lettuce -s 1:　只执行场景"加法" (执行第1个场景)
2. lettuce -s 2:　只执行场景"减法" (执行第2个场景)
3. lettuce -s 1,2:　执行场景"加法"和"减法" (执行第1个和第2个场景)
4. lettuce --scenarios=1,2: 等同于lettuce -s 1,2

### 指定标签Tag

#### 标签定义

标签在安卓莴苣中是用"@TAG"定义.

可以对特性或场景设置标签, 应该在特性或场景定义的上一行, 使用"@TAG"语法定义标签.

如果要指定多个标签, 使用类似的语法, "@TAG"之间以空格分隔.

#### 指定标签

例如, 特性文件:

language:zh-CN

特性: 计算器

为了和莴苣交互

作为初学者

我们实现此函数

@F1

场景: 加法

点击文本【"7"】

点击文本【"+"】

点击文本【"8"】

点击文本【"="】

@F1 @F2

场景: 减法

点击文本【"8"】

#点击文本【"-"】

点击文本【"7"】

点击文本【"="】

场景: 乘法

点击文本【"7"】

点击文本【"\*"】

点击文本【"8"】

点击文本【"="】

1. lettuce -t F1:　执行场景"加法"和"减法" (执行标签F1的用例)
2. lettuce -t F2:　只执行场景"减法" (执行标签F2的用例)
3. lettuce -t F1 -t F2:　执行场景"加法"和"减法" (执行标签F1和F2的用例)
4. lettuce -t -F1:　执行场景"减法"和"乘法" (执行无标签F1的用例. "乘法"没有打任何标签, 也会被匹配上)
5. lettuce -t ~F2:　只执行场景"减法" (执行和标签F2相似的用例)
6. lettuce --tag=F1: 等同于lettuce -t F1

### 附: Lettuce命令帮助

Usage: lettuce or type lettuce -h (--help) for help

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-v VERBOSITY, --verbosity=VERBOSITY

The verbosity level

-s SCENARIOS, --scenarios=SCENARIOS

Comma separated list of scenarios to run

-t TAGS, --tag=TAGS Tells lettuce to run the specified tags only; can be

used multiple times to define more tags(prefixing tags

with "-" will exclude them and prefixing with "~" will

match approximate words)

-r, --random Run scenarios in a more random order to avoid

interference

--with-xunit Output JUnit XML test results to a file

--xunit-file=XUNIT\_FILE

Write JUnit XML to this file. Defaults to

lettucetests.xml

--failfast Stop running in the first failure

--pdb Launches an interactive debugger upon error

## [特性或场景钩子](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_New_Hook)

我已经修改了lettuce执行框架, 新加了2类(4个)非常有用的钩子机制:

在**某特性前**执行指定函数

在**某特性后**执行指定函数

在**某特性的每一个场景前**执行特定函数

在**某特性的每一个场景后**执行特定函数

你需要的具体操作如下:

在你使用al\_new创建的项目features下, 有一个本钩子机制的模板文件"callback.py"

按照你想要的特定位置, 去掉相应的注释, 添加你想执行的代码

### 举例

比如我想, 在特性"计算器"的每一个场景前, 打印每个场景的名字

找到"某某特性中每一个场景前..."部分

# 某某特性中每一个场景前执行的函数

#@before\_scenario\_in\_feature\_of(u"计算器") # provide your feature name here

#def before\_scenario(scenario):

# # adding the operation before <each scenario> of <this feature>

# # e.g.

# # print "before @ ", scenario.name.encode("utf-8")

# pass

1. 去掉注释符号

2. @before\_scenario\_in\_feature\_of(u"xxx") 中指定你想要执行操作的特性名字

3. 添加想要执行的功能, 比如本例中我打印场景名: print "before @ ", scenario.name.encode("utf-8")

# 特性"计算器"中每一个场景前执行本函数

@before\_scenario\_in\_feature\_of(u"计算器")

def before\_scenario(scenario):

print "before @ ", scenario.name.encode("utf-8")

## [常用正则表达式](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Fequent_Re)

### 步骤撰写涉及的格式

* mode\_pvs - pv表达式模式, 常用于元素属性定义

# 属性值定义的表达式样例: 如【text=9】或【text=9,index=2】

# pv, 即property-value简写.

# 属性值以等式"property=value"表达;

# 等式左右用英文等号"="分隔, 等号前后不能有空格

# 有1个或多个等式;

# 多个等式之间用英文逗号","分隔, 逗号前后不能有空格

# 形如:　【xxx=yyy,aaa=bbb,...】

# 其中, x/y/a/b由定义的validch组成(请参考mode\_validch介绍)

mode\_pvs = ur'((?:%s+=%s+[,]?)+)' % (mode\_validch, mode\_validch) # property=value,...的正则表达式

* mode\_text - text模式, 文本

# 文本格式的表达式样例: 如【"xyz123"】, xyz123为text值

# 其中, ""中间的字符串被提取为text值, 中间的字符不能为"

mode\_text = ur'"([^"]+)"' #ur'([^\s]+)'

* mode\_int - int模式, 整数

# 无符号整数的表达式样例: 如【123】或【7】

mode\_int = ur'(\d+)'

* mode\_position - position模式, 坐标

# 无符号整数形成的坐标的表达式样例: 如【8,10】

# 其中, 8为x坐标, 10为y坐标

mode\_position = ur'(\d+),(\d+)'

* mode\_word - word模式, 单词

# a-z/A-z/\_形成的单词的表达式样例: 如【supermarket】

mode\_word = ur'(\w+)'

* mode\_keycode - keycode模式, 键码

# 格式0x的十六进制表达式样例: 如【0x3E】或【0xFF】

# 以后可能会使用整数, 不再使用键码.

mode\_keycode = ur'(0x[\da-fA-F]+)'

* mode\_file - file模式, 文件路径

# 文件路径的表达式样例: 如【123.txt】或【/sdcard/123.txt】

# 非@字符组成的字符串,请满足日常文件路径/目录路径/文件名/目录名要求

mode\_file= ur'([^@]+)'

* mode\_activity - activity模式, 安卓activity名

mode\_activity, 同mode\_file的定义

### 步骤扩展涉及的格式(feature文件中步骤撰写很少涉及)

* pv表达式的合法字符, 普通feature文件撰写不会直接用到此模式.

# mode\_validch, pv表达式中的合法字符:除了1. 空白字符（如空格／回车／制表等不会被打印显示的字符）2. 英文逗号, 3. 英文等号=, 其他的都认为是pv表达式的合法字符

mode\_validch = ur'(?:[^\s,=])'

* [代码扩展用到]提取pv表达式为各个等式的正则表达式, 普通feature文件撰写不会直接用到此模式.

mode\_pv\_sentence = ur'(%s+=%s+)[,]?' % (mode\_validch, mode\_validch)

* [代码扩展用到]从pv表达式分解为p和v的正则表达式, 普通feature文件撰写不会直接用到此模式.

mode\_pv\_extract = ur'(%s+)=(%s+)' % (mode\_validch, mode\_validch)

## [多设备](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Multi-devices)支持

### 配置文件

在安卓莴苣的配置文件的phone部分可以指定最多3个设备.

android\_lettuce.conf

[phone]

is\_only\_one\_device = no

device1 = CKT Z

device2 = HIKe 828

device3 = HIKe 838

is\_only\_one\_device = yes|no

yes: 当前只连接了一台手机设备, 将不会使用设备名来选择设备

no: 当前只连接了多于一台手机设备, 将会使用设备名来选择设备

device1/device2/device3

手机设备1/2/3的名字. "adb devices"显示出来的名字.

### 语句使用

1. 检查是否有支持设备2/设备3的预定义步骤

执行al\_steps [project], 如果在输出中有"...@手机2"/"...@手机3", 表明已经有支持设备2/设备3的预定义步骤.如果没有, 那么在配置文件正确配置情况(is\_only\_one\_device = yes, 并且设备名正确配置)下, 执行al\_update [project], 会更新出多设备预定义步骤.

2. 在预定义步骤中使用"@手机2"/"@手机3"来指定执行的设备.

如果在设备1上执行, 不需要添加"@手机1", 缺省为在设备1上执行.

比如,

在手机1上点击文字7: "点击页面元素【text=7】"

在手机2上点击文字7: "点击页面元素【text=7】@手机2"

在手机3上点击文字7: "点击页面元素【text=7】@手机3"

## [页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

在很多预定义步骤中用到了通过页面元素(UI Element)属性定位页面元素的方法.

### 举例1

比如有如下的页面:

<node index="0" text="" resource-id="android:id/list" class="android.widget.ListView" ...>

<node index="0" text="WIRELESS & NETWORKS" resource-id="" class="android.widget.TextView" .../> => 目标元素1

<node index="1" text="" resource-id="" class="android.widget.LinearLayout" ...>

<node index="1" text="" resource-id="" class="android.widget.RelativeLayout" ...>

<node index="0" text="Wi‑Fi" resource-id="android:id/title" class="android.widget.TextView" .../> => 目标元素2

</node>

<node index="2" text="ON" resource-id="com.android.settings:id/switchWidget" class="android.widget.Switch" .../> => 目标元素3

</node>

...

</node>

以(u'点击页面元素【%s】$' % (mode\_pvs))预定义步骤举例:

要使用此步骤, 你应该完成【%s】属性定义, 定义(在【】中的文字)应该满足mode\_pvs要求的格式.

如果你对此格式不清楚, 你应该先了解清楚此格式的含义, 才能显出符合要求的步骤语句.

了解模式格式要求, 可以参考[安卓莴苣常用正则表达式](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Fequent_Re), 也可以阅读lettuce代码库中regex.py源码.

1. 点击目标元素1的等价语句:

点击页面元素【index=0,class=android.widget.TextView】

点击页面元素【index=0,instance=1】

点击页面元素【class=android.widget.TextView,instance=0】

2. 点击目标元素2的等价语句:

点击页面元素【text=Wi‑Fi】

点击页面元素【resourceId=android:id/title】

3. 点击目标元素3的等价语句:

点击页面元素【text=ON】

点击页面元素【index=2】

点击页面元素【resourceId=com.android.settings:id/switchWidget】

点击页面元素【resourceIdMatches=.\*/switchWidget】

### 举例2

这里区分一下"xxx", "xxxContains", "xxxMatches", "xxxStartsWith"属性的区别.

点击页面元素【text=Open】 - 点击屏幕上属性(text为Open)的页面元素

点击页面元素【textContains=ckt】 - 点击屏幕上属性(text中包含ckt的, 比如text为Open\_ckt或ckt\_button或ckt123)的页面元素

点击页面元素【textMatches=ckt\d+】 - 点击屏幕上属性(text匹配正则表达式, 比如text为ckt123会被匹配, 而ckt\_button则不会被匹配)的页面元素

点击页面元素【textStartsWith=ckt】 - 点击屏幕上属性(text中以ckt开头的, 比如text为ckt\_button或ckt123满足条件, 而Open\_ckt则不满足条件)的页面元素

### 举例3

instance实际上并不是UI Element的属性, 你在uiautomatorviewer中应该是看不到这个属性的.

可以理解为, 它是框架附加的一个虚拟属性. 它的含义是, 表示具有其他属性定义的页面元素的第几个实例.

需要注意的是, 程序以0开始计数的, 也就是说instance=0作为日常生活中所说的第1个, 以此类推.

点击页面元素【enabled=true, instance=3】 - 点击屏幕上第4个属性(enabled为true)的页面元素

### 属性列表

可供大家使用的属性有以下这些. 这些定义在预定义步骤的click.py和longclick.py中也能找到.

|  |  |
| --- | --- |
| 属性名 | 说明 |
| instance | 第几个实例 |
| text | 属性text等于(此字符串) |
| textContains | 属性text包含(此字符串) |
| textMatches | 属性text匹配(此正则表达式) |
| textStartsWith | 属性text以(此字符串)开始 |
| resourceId | 属性resourceId等于(此正则表达式) |
| resourceIdMatches | 属性resourceId匹配(此正则表达式) |
| index | 属性index等于 |
| className | 属性className等于(此字符串) |
| classNameMatches | 属性className匹配(此正则表达式) |
| description | 属性content-desc等于(此字符串) |
| descriptionContains | 属性content-desc包含(此字符串) |
| descriptionMatches | 属性content-desc匹配(此正则表达式) |
| descriptionStartsWith | 属性description以(此字符串)开始 |
| checkable | 属性checkable等于true/false(不能写作True/False) |
| checked | 属性checked等于true/false(不能写作True/False) |
| clickable | 属性clickable等于true/false(不能写作True/False) |
| longClickable | 属性longClickable等于true/false(不能写作True/False) |
| scrollable | 属性scrollable等于true/false(不能写作True/False) |
| enabled | 属性enabled等于true/false(不能写作True/False) |
| focusable | 属性focusable等于true/false(不能写作True/False) |
| focused | 属性focused等于true/false(不能写作True/False) |
| selected | 属性selected等于true/false(不能写作True/False) |
| packageName | 属性packageName等于(此字符串) |
| packageNameMatches | 属性packageName匹配(此字符串) |

## [物理键名字和键码](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Key_Keycode)

在press预定义步骤中会用到物理键和键码.

以下的定义在预定义步骤的press.py中也能找到.

### 支持的物理键名字

|  |  |
| --- | --- |
| 物理键名字 | 说明 |
| home | HOME键 |
| back | 回退键 |
| power | 电源键 |
| menu | 菜单键 |
| search | 搜索键 |
| volume\_up | 音量+键 |
| volume\_down | 音量-键 |
| volume\_mute | 静音键 |
| left | 导向左键 |
| right | 导向右键 |
| up | 导向上键 |
| down | 导向下键 |
| center | 导向居中键 |
| enter | 回车键 |
| delete(or del) | 删除键 |
| recent(recent apps) | 最近使用APP键 |
| camera | 拍照键 |

### 键码

<http://blog.csdn.net/feizhixuan46789/article/details/16801429>

"点按键码【0x07】", 键码要提供2为16进制形式的格式"0xnn".

比如, 键码为7(10进制)的数字"0", 应该写为"0x07".

键码为15(10进制)的数字"8", 应该写为"0x0f".

键码为16(10进制)的数字"9", 应该写为"0x10".

## [导出Automatorviewer需要的数据到本地](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Dump_Utility_for_Automatorviewer)

### 背景

在安卓莴苣撰写用例的过程中, 通过uiautomatorviewer工具来查看界面是经常被需要的.

但是, 安卓莴苣是发布和部署在ubuntu系统的, 为了调试脚本手机设备是挂在ubuntu系统上的. 比较不幸的消息是, uiautomatorviewer工具在ubuntu下面是不能使用的, 只能在windows环境下使用. 因此矛盾就出现啦: 手机挂在ubuntu虚拟机系统上, windows下的viewer工具是不能直接从手机设备截图和获取页面信息的.

为了从手机设备截图和获取页面信息, 你不得不从ubuntu虚拟机上摘下手机设备, 挂到windows环境下. 这很麻烦, 很不方便.

### 安卓莴苣viewer辅助工具

为了能从ubuntu虚拟机上的挂载的手机设备截图和获取页面信息, 在安卓莴苣体系中专门设计了一个安卓莴苣viewer辅助工具.

工具由2部分组成: capture4viewer.py 和 ftpget\_atwin.bat

存放位置:

/home/ckt/lettuce/tools/viewer

#### capture4viewer.py

在ubuntu下执行.

Usage: capture4viewer.py [options]

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-n PAGENAME, --pagename=PAGENAME

provide the page name

-p PATH, --path=PATH specify the store path

截图和页面信息文件保存目录[viewer home], 在capture4viewer.py中base\_path定义缺省值(/home/ckt/lettuce/viewer).

你可以在命令中使用-p指定文件保存目录; 如果没有指定, 会使用base\_path缺省值.

你可以修改capture4viewer.py中base\_path值, 改变缺省路径.

截图和页面信息文件缺省名字为mypage.

你也可以在命令中使用-n执行文件名字; 如果没有指定, 会使用mypage作为文件名字.

举例1: 　对手机设备截图(保存为/home/ckt/lettuce/viewer/myname.png), 获取页面信息(保存为/home/ckt/lettuce/viewer/myname.uix).

./capture4viewer.py -n myname

举例2: 　对手机设备截图(保存为/temp/myname.png), 获取页面信息(保存为/temp/myname.uix).

./capture4viewer.py -n myname -p /temp

#### ftpget\_atwin.bat

在windows下执行, 把在ubuntu获得的.png和.uix文件ftp到windows本地.

首先, 你需要把ftpget\_atwin.bat放置到windows本地.

方法1. 尝试使用你ssh客户端自带的sftp工具, sftp客户端可以方便传输文件.

比如, SecureCRT自带一个命令行sftp工具, 可以参考<http://www.cnblogs.com/xwdreamer/p/3673397.html>

比如, xshell客户端, 尝试Ctrl+Alt+F看能不能调出sftp工具.

方法2. 如果方法1中的工具没有安装, 那么, 你可以尝试通过ftp客户端把.bat下载到windows.

ubuntu都打开了ftp服务, 你可以使用账号root:123123.

方法3. 你可以在ubuntu直接cat ftpget\_atwin.bat, 把内容复制粘贴到windows下的.bat文件.

然后, 你应该在ftpget\_atwin.bat设置ftp服务器ip地址(即server), ftp服务器上的文件目录(即remote), 保存到本地的目录(local)

...

rem ftp服务器ip地址

set server=10.120.2.242

rem ftp服务器上文件所在目录

set remote=/home/ckt/lettuce/viewer

rem 保存到本地的目录

set local=d:\tmp\_\_\viewer

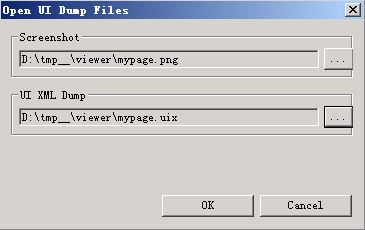
...

最后, 双击ftpget\_atwin.bat, ubuntu远端目录中的.png和.uix文件就会被ftp到本地指定的目录.

### 在uiautomatorviewer中查看页面

首先, 点击uiautomatorviewer工具左上角的open命令按钮

然后, 在ScreenShot中选择对应的.png; 在UI XML Dump中选择对应的.uix.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Uiautomatorviewer_open.jpg)

最后, 确认打开即可看到界面和元素信息.

## [图片比较工具](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Image_Tool)

### 背景

首先, 在安卓莴苣中, 我们并不推荐采用截图比较. 因为这样做的缺点是显而易见的:　比较的鲁棒性差, 并不稳定; 图片比较也很耗时.

所以, 检查点优先采用的还是基于页面元素信息的比较, 而不是截图像素的比较.

但是, 会存在页面元素无法获得的情况, 这样就不得不进行截图的比较啦.

进行截图比较有这么几步:

1. 准备预期的图片

2. 如果希望比较截图中的某一个区域, 那么你需要指定区域. 如果只是对整个截图进行比较, 你可以跳过本步骤.

3. 修改配置文件, 进行图片比对.

### 准备预期的图片

#### 预定义步骤: 截屏并比较【%s】

如果你是想进行图片比对操作, 那么你选择的预定义步骤应该是"截屏并比较【%s】", 而不是"截屏【%s】". 具体的步骤含义, 请参看[【WiKi】【2014-08-28】安卓莴苣预定义步骤之image(截图)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Image).

#### 关闭图片比对

因为最开始是没有预期图片的, 一旦进行图片比对, 就会失败导致执行不会继续, 所以准备预期图片阶段, 你应该让框架不做任何图片比对操作.

关闭图片比对, 在/home/ckt/lettuce/android\_lettuce.conf设置如下

...

[image]

is\_compared = no

...

#### 截取图片

设置完成后, "截屏并比较【%s】"就退化为"截屏【%s】"啦.

执行一遍安卓莴苣, 截图就会把保存在[your project]/captures目录下面啦.

#### 准备预期图片

仔细检查你的captures下的图片, 如果的确是你预期的内容, 你就可以把captures下面的内容拷贝到images目录, 成为你用例预期的图片啦.

为了方便你拷贝captures下的内容到images目录, 在al系列命令中为大家准备了al\_image命令用来执行这些拷贝或删除操作, 具体内容请参考[【WiKi】【2014-08-22】安卓莴苣系统:目录和常用命令](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Directories).

一般地, 你可以执行al\_image -u [your-project-name], 就会把captures下面的内容拷贝到images目录.

captures和images下的目录结构应该是相同的, 都为"[feature]/[scenario]/图片". 在captures和images下具有相同相对路径的文件就会进行比对.

举例说明, 如果你的特性名为计算器, 场景为加法, 在加法场景中执行"截屏并比较【sum】",

场景: 加法

点击文本【"7"】

点击文本【"+"】

点击文本【"8"】

点击文本【"="】

截屏并比较【sum】

那么, 你的图片目录结构和内容应该这样:

images/计算器/加法/sum.png, 为你应该准备的预期截图

captures/计算器/加法/sum.png, 为每次你执行步骤后的截图

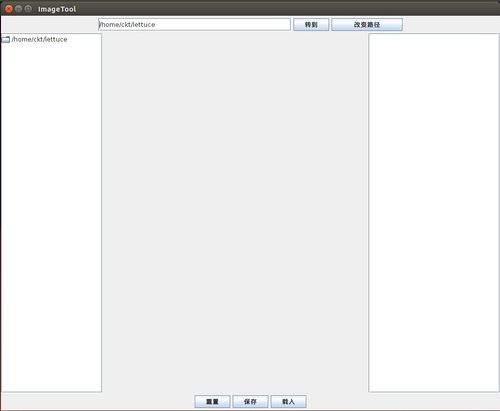
### 指定比较区域

如果希望比较截图中的某一个区域, 那么你需要指定区域. 如果只是对整个截图进行比较就可以, 你可以跳过本步骤.

为了指定截图中的某一个区域, 在安卓莴苣体系中设计了一个专门的区域选择工具, al\_imgtool.

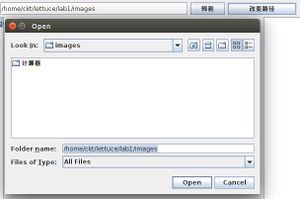
#### 启动工具

在x窗口环境中, 执行al\_imgtool, 启动区域选择工具.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Al_imgtool_1.jpg)

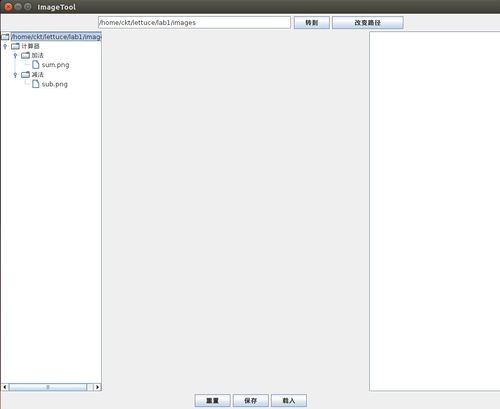
#### 指定目录

缺省的路径在/home/ckt/lettuce, 你需要点击"改变路径", 选择到你要进行区域处理的目录, 一般为你项目下的images目录.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Al_imgtool_2.jpg)

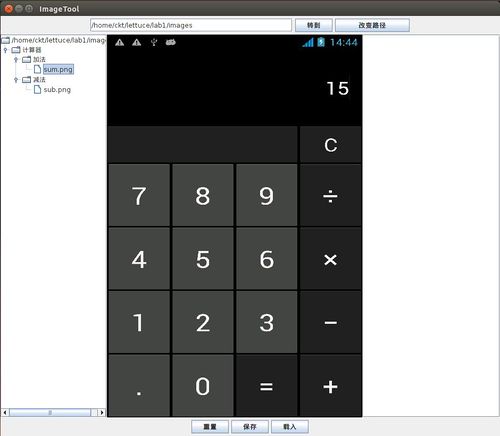
在路径编辑框中, 你也可以直接编辑, 而不使用"改变路径"目录选择对话框进行.

你需要点击"转到", 新路径下的内容才会更新到左侧的目录树. 目录树中只会对.png的内容进行显示.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Al_imgtool_3.jpg)

#### 选择图片

双击你想要选择区域的.png图片, 图片会显示在界面中央.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Al_imgtool_4.jpg)

#### 选择区域

使用鼠标选择你想要比较的区域. 你可以同时选择多个区域, 考虑到图片比较的高耗时, 请只选择必须的区域.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Al_imgtool_5.jpg)

在选择区域后, 区域矩形信息会显示到右侧列表中. 如果你不满意选择, 你也可以升级区域信息, 去掉改区域.

在区域选定后, 请点击"保存". 另外, "重置"为清空保存的区域信息, 然后重新开始选择区域; "载入"为在你对区域列表进行增删后, 还没有保存前, 你可以放弃之前的增删, 重新载入之前保存的区域信息.

保存好你此张图片的区域信息后, 如果进行此张图片比对, 安卓莴苣就会对区域进行比对, 而不再是对整个图片进行比对.

### 图片比对

#### 打开图片比对

在准备预期图片时候, 关闭了图片比对功能. 现在我们重新打开图片对比, 在以后的执行过程中, "截屏并比较【%s】"就会开始进行图片或区域对比啦.

打开图片比对, 在/home/ckt/lettuce/android\_lettuce.conf设置如下

...

[image]

is\_compared = yes

...

## [基于坐标的录制工具](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Recorder_Tool)

### 背景

有些时候, 应用界面是通过图形库画出来的, 比如HiCam机身很多介绍就是OpenGL绘制出来的, 不支持定位页面元素. 这种情况下, 基于坐标的操作和截图的比较就是安卓莴苣不得不选择的方法.

基于坐标的操作, 通过截图后在图形工具中查看坐标点进行步骤撰写是一种方法, 但是这种方法非常低效, 也不方便.

为了快速的获得基于坐标操作的信息, 我专门定制了用于安卓莴苣的基于坐标操作步骤录制工具.

### 环境要求

本工具要求:

1. Java环境
2. 图形界面环境
3. 安卓设备已经通过usb连接, 并且能被adb识别
4. 安卓设备中已经安装monkey服务(通常无需操作)

### 启动

#### ubuntu启动

在安卓莴苣环境中, 多种途径可以启动录制工具:

1. 直接使用al\_recorder启动.

如果你的环境中还没有此命令, 请执行添加命令:

/home/ckt/lettuce/tools/scripts/allink.sh

2. 使用启动脚本

/home/ckt/lettuce/tools/rectool/start\_rectool.sh

3. 直接用java运行alrec.jar

java -jar <alrec.jar的全路径>

#### windows启动

在windows下也可以使用录制工具:

只可以通过java运行alrec.jar来启动工具

java -jar <alrec.jar的全路径>

### 使用

1. 录制工具启动后界面如图

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Alrec1.jpg)

2. 在界面中执行想要进行的操作

截屏预览区:触摸/长按/拖动操作

命令输入区: 等待/输入编辑框内容/执行shell命令/启动Activity/截屏

物理按键区: 如图列出的物理按键/点亮屏幕(WAKE)

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Alrec2.jpg)

3. 在已录制步骤列表中, 会记录已经记录下出操作步骤

你可以双击某步骤进行删除; 选中某步骤, 下一个步骤将会添加到选中步骤的后面

4. 录制完成后, 可以从菜单中选择命令:文件/导出/安卓莴苣, 把录制的步骤导出未安卓莴苣特性文件.

导出的特性文件将存放在[启动录制工具时所在目录]下的"al"目录中

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Alrec3.jpg)

5. 如果想修改之前导出的安卓莴苣特性文件中的步骤, 可以从菜单中选择命令:文件/导入/安卓莴苣, 把保存的步骤导入录制工具进行编辑.

能导入的步骤只能为录制工具可以录制保存的预定义步骤.

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Alrec4.jpg)

# 预定义步骤参考

## [预定义步骤之click(点击)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Click)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### click.py

@step(u'点击页面元素【%s】$' % (mode\_pvs))

点击页面元素【text=Open】 - 点击屏幕上属性(text为Open)的页面元素

点击页面元素【text=On, index=1】 - 点击屏幕上属性(index为1, text为On)的页面元素

点击页面元素【text=8,index=1,enabled=true】 - 点击屏幕上属性(index为1, text为8, enabled为true)的页面元素

点击页面元素【enabled=true, instance=3】 - 点击屏幕上第4个(程序以0开始作为日常生活中所说的第1个)属性(enabled为true)的页面元素

@step(u'点击文本【%s】$' % (mode\_text))

点击文本【"Wifi & Network"】 - 点击屏幕上属性(text为Wifi & Network)的页面元素

@step(u'点击描述【%s】$' % (mode\_text))

点击文本【"the buttons's description"】 - 点击屏幕上属性(content-desc为the buttons's description)的页面元素

@step(u'点击页面元素，父【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

点击页面元素，父【text=wifi】自【index=2】 - 点击屏幕上页面元素, 条件为其父节点有属性特征(text=wifi), 自己有属性特征(index=2)

@step(u'点击页面元素，兄【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

点击页面元素，兄【text=wifi,instance=1】自【index=2】 - 点击屏幕上页面元素, 条件为其兄弟节点为第2个(从0开始数的第1个)属性特征(text=wifi), 自己有属性特征(index=2)

@step(u'点击【%s】上边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

点击【text=7】上边的【index=3】页面元素 - 点击屏幕上页面元素, 其物理位置位于元素(text=7)上边, 自己有属性特征(index=3)

@step(u'点击【%s】下边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

点击【text=7】下边的【index=3】页面元素 - 点击屏幕上页面元素, 其物理位置位于元素(text=7)下边, 自己有属性特征(index=3)

@step(u'点击【%s】左边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

点击【text=7】下边的【index=3】页面元素 - 点击屏幕上页面元素, 其物理位置位于元素(text=7)左边, 自己有属性特征(index=3)

@step(u'点击【%s】右边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

点击【text=7】下边的【index=3】页面元素 - 点击屏幕上页面元素, 其物理位置位于元素(text=7)右边, 自己有属性特征(index=3)

@step(u'点击页面坐标【%s】$' % (mode\_position))

点击页面坐标【100,50】 - 点击坐标点(x=100,y=50)

@step(u'点击可滚动框架【%s】内的特征文字【%s】页面元素$' % (mode\_pvs, mode\_text))

点击可滚动框架【className=android.widget.ListView,index=2】内的特征文字【"Mr. Yu"】页面元素 - 在可滚动的框架(index=2,视图类为android.widget.ListView)里找到属性特征(text为Mr. Yu)的页面元素进行点击

@step(u'点击可滚动框架【%s】内的特征描述【%s】页面元素$' % (mode\_pvs, mode\_text))

点击可滚动框架【className=android.widget.ListView,instance=2】内的特征描述【"Mine"】页面元素 - 在可滚动的框架(第3个视图类为android.widget.ListView)里找到属性特征(content-desc为Mine)的页面元素进行点击

@step(u'点击可滚动框架【%s】内的特征序号【%s】页面元素$' % (mode\_pvs, mode\_int))

不推荐使用本预定义语句

@step(u'点击可滚动框架【%s】内的页面元素【%s】且其子孙结点含text值【%s】$' % (mode\_pvs, mode\_pvs, mode\_text))

点击可滚动框架【className=android.widget.ListView,index=2】内的页面元素【checked=true】且其子孙结点含text值【"Mr. Yu"】 -

此语句为"点击可滚动框架【%s】内的特征文字【%s】页面元素"的完备写法. 在可滚动的框架(index=2,视图类为android.widget.ListView)里找到属性特征(checked=true)的页面元素A, 且元素A其自己或其子孙中包含属性(text为Mr. Yu), 点击页面元素A

@step(u'点击可滚动框架【%s】内的页面元素【%s】且其子孙结点含description值【%s】$' % (mode\_pvs, mode\_pvs, mode\_text))

点击可滚动框架【className=android.widget.ListView,index=2】内的页面元素【checked=true】且其子孙结点含description值【"Mr. Yu"】 -

此语句为"点击可滚动框架【%s】内的特征描述【%s】页面元素"的完备写法. 在可滚动的框架(index=2,视图类为android.widget.ListView)里找到属性特征(checked=true)的页面元素A, 且元素A其自己或其子孙中包含属性(content-desc为Mr. Yu), 点击页面元素A

@step(u'点击不可滚动框架【%s】内的页面元素【%s】且其子孙结点含instance序号【%s】$' % (mode\_pvs, mode\_pvs, mode\_int))

点击可滚动框架【className=android.widget.ListView,index=2】内的页面元素【checked=true】且其子孙结点含instance值【2】 -

此语句为"点击可滚动框架【%s】内的特征序号【%s】页面元素"的完备写法. 在不可滚动的框架(index=2,视图类为android.widget.ListView)里找到属性特征(checked=true)的页面元素A, 且元素A其子孙中包含属性(instance为2,即能数到第3个), 点击页面元素A

@step(u'点击页面元素左上角【%s】$' % (mode\_pvs))

和"点击页面元素【%s】$"类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击页面元素左上角，父【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

和"点击页面元素，父【%s】自【%s】$类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击页面元素左上角，兄【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

和"点击页面元素，兄【%s】自【%s】$"类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击【%s】上边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】上边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击【%s】下边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】下边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击【%s】左边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】左边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击【%s】右边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】右边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素左上角

@step(u'点击页面元素右下角【%s】$' % (mode\_pvs))

和"点击页面元素【%s】"类似, 前者点击中心, 此步骤点击页面元素右下角

@step(u'点击页面元素右下角，父【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

和"点击页面元素，父【%s】自【%s】$"类似, 前者点击中心, 此步骤点击页面元素右下角

@step(u'点击页面元素右下角，兄【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

和"点击页面元素，兄【%s】自【%s】$"类似, 前者点击中心, 此步骤点击页面元素右下角

@step(u'点击【%s】上边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】上边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素右下角

@step(u'点击【%s】下边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】下边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素右下角

@step(u'点击【%s】左边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】左边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素右下角

@step(u'点击【%s】右边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

和"点击【%s】右边的【%s】页面元素$"类似, 前者点击中心, 此步骤点击页面元素右下角

## [预定义步骤之longclick(长按)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Longclick)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### longclick.py

长按和点击click预定义步骤非常相似, 请参考[【WiKi】【2014-08-28】安卓莴苣预定义步骤之click](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Click)

@step(u'长按页面元素【%s】$' % (mode\_pvs))

@step(u'长按页面元素，父【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

@step(u'长按页面元素，兄【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】上边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】下边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】左边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】右边的【%s】页面元素$' % (mode\_pvs, mode\_pvs))

@step(u'长按页面坐标【%s】$' % (mode\_position))

@step(u'长按可滚动框架【%s】内的特征文字【%s】页面元素$' % (mode\_pvs, mode\_text))

@step(u'长按可滚动框架【%s】内的特征描述【%s】页面元素$' % (mode\_pvs, mode\_text))

@step(u'长按不可滚动框架【%s】内的特征序号【%s】页面元素$' % (mode\_pvs, mode\_int))

@step(u'长按可滚动框架【%s】内的页面元素【%s】且其子孙结点含text值【%s】$' % (mode\_pvs, mode\_pvs, mode\_text))

@step(u'长按可滚动框架【%s】内的页面元素【%s】且其子孙结点含description值【%s】$' % (mode\_pvs, mode\_pvs, mode\_text))

@step(u'在框架【%s】内，长按页面元素【%s】且子孙结点含instance序号【%s】$' % (mode\_pvs, mode\_pvs, mode\_int)) @step(u'长按页面元素左上角【%s】$' % (mode\_pvs))

@step(u'长按页面元素左上角，父【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

@step(u'长按页面元素左上角，兄【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】上边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】下边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】左边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】右边的【%s】页面元素左上角$' % (mode\_pvs, mode\_pvs))

@step(u'长按页面元素右下角【%s】$' % (mode\_pvs))

@step(u'长按页面元素右下角，父【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

@step(u'长按页面元素右下角，兄【%s】自【%s】$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】上边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】下边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】左边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

@step(u'长按【%s】右边的【%s】页面元素右下角$' % (mode\_pvs, mode\_pvs))

## [预定义步骤之image(截图)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Image)

### image.py

@step(u'截屏【%s】$' % (mode\_word))

截屏【my\_picture】 - 截取屏幕, 在captures目录中保存到路径"[feature]/[scenario]/my\_picture.png"

@step(u'截屏并比较【%s】$' % (mode\_word))

截屏并比较【my\_picture】 - 截取屏幕, 在captures目录中保存到路径"[feature]/[scenario]/my\_picture.png";

并且和images目录中的"[feature]/[scenario]/my\_picture.png"进行比对

## [预定义步骤之doubleclick(双击)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Doubleclick)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### doubleclick.py

@step(u'快速双击页面元素【%s】$' % (mode\_pvs))

快速双击页面元素【text=open】 - 快速双击特征(text为open)的页面元素

@step(u'快速双击页面坐标【%s】$' % (mode\_position))

快速双击页面坐标【540,960】- 快速双击坐标(x=540,y=960)

## [预定义步骤之fs(File System,文件系统操作)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Fs)

### fs.py

@step(u'执行shell命令【%s】$' % (mode\_text))

执行shell命令【"mkdir /sdcard/ykm"】 - 在手机上执行shell命令: mkdir /sdcard/ykm, 效果为在手机上创建/sdcard/ykm的目录

@step(u'设置路径【%s】文件数检查点$' % (mode\_file))

设置路径【/sdcard/ykm】文件数检查点 - 记录下/sdcard/ykm下的文件数. 此预定义步骤为"路径【%s】文件数相对上个检查点应该增加/减少【%s】$"步骤的前置步骤.

@step(u'路径【%s】文件数相对上个检查点应该增加【%s】$' % (mode\_file, mode\_int))

路径【/sdcard/ykm】文件数相对上个检查点应该增加【2】 - 希望相对上一个文件数检查点, /sdcard/ykm目录下的文件数+2

@step(u'路径【%s】文件数相对上个检查点应该减少【%s】$' % (mode\_file, mode\_int))

路径【/sdcard/ykm】文件数相对上个检查点应该减少【2】 - 希望相对上一个文件数检查点, /sdcard/ykm目录下的文件数-2

@step(u'设置路径【%s】下【%s】类型文件数检查点$' % (mode\_file, mode\_word))

设置路径【/sdcard/ykm】下【txt】类型文件数检查点 - 记录下/sdcard/ykm下的txt后缀的文件数. 此预定义步骤为"路径【%s】下【%s】类型文件数相对上个检查点应该增加/减少【%s】$$"步骤的前置步骤.

@step(u'路径【%s】下【%s】类型文件数相对上个检查点应该增加【%s】$' % (mode\_file, mode\_word, mode\_int))

路径【/sdcard/ykm】下【txt】类型文件数相对上个检查点应该增加【2】 - 希望相对上一个文件数检查点, /sdcard/ykm目录下的.txt文件数+2

@step(u'路径【%s】下【%s】类型文件数相对上个检查点应该减少【%s】$' % (mode\_file, mode\_word, mode\_int))

路径【/sdcard/ykm】下【txt】类型文件数相对上个检查点应该减少【2】 - 希望相对上一个文件数检查点, /sdcard/ykm目录下的.txt文件数-2

@step(u'路径【%s】下应该存在文件【%s】$' % (mode\_file, mode\_file))

路径【/sdcard/ykm】下应该存在文件【1.txt】 - 希望/sdcard/ykm下有文件1.txt

@step(u'删除文件【%s】$' % (mode\_file))

删除文件【/sdcard/ykm/1.txt】 - 删除文件/sdcard/ykm/1.txt

@step(u'删除路径【%s】下所有文件$' % (mode\_file))

删除路径【/sdcard/ykm】下所有文件 - 删除/sdcard/ykm下所有文件, 但不删除目录/sdcard/ykm

@step(u'删除目录【%s】$' % (mode\_file))

删除目录【/sdcard/ykm】 - 删除目录/sdcard/ykm

@step(u'文件或目录【%s】改为【%s】$' % (mode\_file, mode\_file))

文件或目录【/sdcard/ykm】改为【/sdcard/zero】 - 目录/sdcard/ykm改为/sdcard/zero

文件或目录【/sdcard/zero/1.txt】改为【/sdcard/zero/2.txt】 - 文件/sdcard/zero/1.txt改为/sdcard/zero/2.txt

## [预定义步骤之press(物理键和键码)](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_PDS_Press)

可用的物理键名字和键码请参考[【WiKi】【2014-08-22】安卓莴苣系统:物理键名字和键码](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_Key_Keycode)

### press.py

@step(u'点按物理键【%s】$' % (mode\_word))

点按物理键【recent】 - 按RecentAPP键, 调用最新使用程序界面

@step(u'点按键码【%s】$' % (mode\_keycode))

点按键码【0x07】 - 按键码为7的键, 相当于按"0"键

@step(u'快速双按物理键【%s】$' % (mode\_word))

快速双按物理键【back】 - 快速按2次回退键

@step(u'快速双按键码【%s】$' % (mode\_keycode))

快速双按键码【0x07】 - 快速按2次键码为7的键

@step(u'同时点按键码【%s】和【%s】$' % (mode\_keycode, mode\_keycode))

同时点按键码【0x07】和【0x02】 - 同时按键码为7和2的键, 即按alt+0

## [预定义步骤之drag(拖动)](http://10.120.10.100:9002/Test_Knowledge_Share_Zhao_Qi_Android_Lettuce_PDS_Drag)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### drag.py

@step(u'从坐标【%s】拖动至坐标【%s】$' % (mode\_position, mode\_position))

从坐标【100,200】拖动至坐标【500,600】 - 手指从屏幕的坐标点(100,200)拖动至坐标(500,600)

@step(u'以步长【%s】从坐标【%s】拖动至坐标【%s】$' % (mode\_int, mode\_position, mode\_position))

以步长【100】从坐标【100,200】拖动至坐标【500,600】 - 手指拖动100步从屏幕的坐标点(100,200)拖动至坐标(500,600)

@step(u'拖动页面元素【%s】至坐标【%s】$' % (mode\_pvs, mode\_position))

拖动页面元素【text=设置】至坐标【400,500】 - 手指拖动页面上text为"设置"的页面元素至坐标点(400,500)

拖动页面元素【className=com.android.mms】至坐标【400,500】 - 手指拖动页面上className为"com.android.mms"的页面元素至坐标点(400,500)

@step(u'以步长【%s】拖动页面元素【%s】到坐标【%s】$' % (mode\_int, mode\_pvs, mode\_position))

以步长【100】拖动页面元素【text=设置】到坐标【400,500】 - 手指拖动100步，将页面元素拖动到坐标(400,500)

@step(u'拖动页面元素【%s】至页面元素【%s】$' % (mode\_pvs, mode\_pvs))

拖动页面元素【text=相机】只页面元素【className=com.android.mms】 - 拖动页面text为“相机”的元素至className为“com.android.mms”的位置。

@step(u'拖动文本【%s】至坐标【%s】$' % (mode\_text, mode\_position))

拖动文本【"短信"】至坐标【100,400】 - 将文本为“短信”的元素拖动至坐标点(100,400)

@step(u'拖动文本描述【%s】至坐标【%s】$' % (mode\_text, mode\_position))

拖动文本描述【"应用"】至坐标【200,300】 - 拖动description 为“应用”的元素至坐标点(200,300)

@step(u'拖动文本【%s】至另一个文本【%s】$' % (mode\_text, mode\_text))

拖动文本【"相机"】至另一个文本【"设置"】 - 将text为"相机"的对象拖动至text为"设置"的位置

@step(u'拖动文本描述【%s】至一个文本【%s】$' % (mode\_text, mode\_text))

拖动文本描述【"应用"】只一个文本【"相机"】 - 将description为"应用"的对象拖动至text为"相机"的位置

@step(u'拖动文本描述【%s】至另一个文本描述【%s】$' % (mode\_text, mode\_text))

拖动文本描述【"应用"】只另一个文本描述【"信息"】 - 将description为"应用"的对象拖动至description为"相机"的位置

@step(u'拖动文本【%s】至一个文本描述【%s】$' % (mode\_text, mode\_text))

拖动文本【"相机"】只一个文本描述【"应用"】 - 将text为"应用"的对象拖动至description为"相机"的位置

## [预定义步骤之fling(快速滑动)](http://10.120.10.100:9002/Test_Knowledge_Share_Zhao_Qi_Android_Lettuce_PDS_Fling)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### fling.py

@step(u'手指快速向左滑动屏幕$')

手指快速向左滑动屏幕 - fling模拟的是真实的手指操作，快速向左滑动屏幕

@step(u'手指快速向右滑动屏幕$')

和“手指快速向左滑动屏幕”类似，此处为向右

@step(u'手指快速向上滑动屏幕$')

和“手指快速向左滑动屏幕”类似，此处为向上

@step(u'手指快速向下滑动屏幕$')

和“手指快速向左滑动屏幕”类似，此处为向下

@step(u'手指快速向上边起始位置滑动$')

手指快速向上边起始位置滑动 - 模拟手指真实快速操作，向上边起始位置滑动，即从下方向上方开始的位置快速滑动

@step(u'手指快速向下边结束位置滑动$')

和“手指快速向上边起始位置滑动” 类似

@step(u'手指快速向左边起始位置滑动$')

和“手指快速向上边起始位置滑动” 类似

@step(u'手指快速向右边结束位置滑动$')

和“手指快速向上边起始位置滑动” 类似

@step(u'手指快速向左滑动框架【%s】$' % (mode\_pvs))

手指快速向左滑动框架【className=android.widget.ListView】 - 在框架【className=android.widget.ListView】内快速向左滑动

@step(u'手指快速向右滑动框架【%s】$' % (mode\_pvs))

和“手指快速向左滑动框架【%s】”类似 - 在框架内快速向右滑动

@step(u'手指快速向上滑动框架【%s】$' % (mode\_pvs))

和“手指快速向左滑动框架【%s】”类似 - 在框架内快速向上滑动

@step(u'手指快速向下滑动框架【%s】$' % (mode\_pvs))

和“手指快速向左滑动框架【%s】”类似 - 在框架内快速向下滑动

@step(u'手指快速向上边起始位置滑动框架【%s】$' % (mode\_pvs))

手指快速向上边其实位置滑动框架【className=android.widget.ListView】 - 向起始位置滑动框架

@step(u'手指快速向下边结束位置滑动框架【%s】$' % (mode\_pvs))

和“手指快速向上边起始位置滑动框架【%s】”类似

@step(u'手指快速向左边起始位置滑动框架【%s】$' % (mode\_pvs))

和“手指快速向上边起始位置滑动框架【%s】”类似

@step(u'手指快速向右边结束位置滑动框架【%s】$' % (mode\_pvs))

和“手指快速向上边起始位置滑动框架【%s】”类似

## [预定义步骤之scroll(滚动)](http://10.120.10.100:9002/Test_Knowledge_Share_Zhao_Qi_Android_Lettuce_PDS_Scroll)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### scroll.py

@step(u'手指向左滚动屏幕$')

手指向左滚动屏幕 - 滚动屏幕

@step(u'手指向右滚动屏幕$')

和“手指向左滚动屏幕”类似 - 向右滚动

@step(u'手指向上滚动屏幕$')

和“手指向左滚动屏幕”类似 - 向上滚动

@step(u'手指向下滚动屏幕$')

和“手指向左滚动屏幕”类似 - 向下滚动

@step(u'以步长【%s】手指向左滚动$' % (mode\_int))

以步长【100】手指向左滚动 - 手指用100步完成向左滚动的操作

@step(u'以步长【%s】手指向右滚动$' % (mode\_int))

和“以步长【%s】手指向左滚动”类似

@step(u'以步长【%s】手指向上滚动$' % (mode\_int))

和“以步长【%s】手指向左滚动”类似

@step(u'以步长【%s】手指向下滚动$' % (mode\_int))

和“以步长【%s】手指向左滚动”类似

@step(u'手指向左边起始位置滚动$')

手指向左边起始位置滚动 - 向起始位置滚动

@step(u'手指向右边结束位置滚动$')

和“手指向左边起始位置滚动”类似

@step(u'手指向上边起始位置滚动$')

和“手指向左边起始位置滚动”类似

@step(u'手指向下边结束位置滚动$')

和“手指向左边起始位置滚动”类似

@step(u'手指向左滚动框架【%s】$' % (mode\_pvs))

手指向左滚动框架【className=android.view.View】 - 以className为框架

手指向左滚动框架【text=时钟】 - 以text为框架

@step(u'手指向右滚动框架【%s】$' % (mode\_pvs))

和“手指向左滚动框架【%s】”类似

@step(u'手指向上滚动框架【%s】$' % (mode\_pvs))

和“手指向左滚动框架【%s】”类似

@step(u'手指向下滚动框架【%s】$' % (mode\_pvs))

和“手指向左滚动框架【%s】”类似

@step(u'以步长【%s】手指向左滚动框架【%s】$' % (mode\_int, mode\_pvs))

以步长【100】手指向左滚动框架【text=时钟】 - 以text为框架，用100步完成滚动操作

@step(u'以步长【%s】手指向右滚动框架【%s】$' % (mode\_int, mode\_pvs))

和“以步长【%s】手指向左滚动框架【%s】”类似

@step(u'以步长【%s】手指向上滚动框架【%s】$' % (mode\_int, mode\_pvs))

和“以步长【%s】手指向左滚动框架【%s】”类似

@step(u'以步长【%s】手指向下滚动框架【%s】$' % (mode\_int, mode\_pvs))

和“以步长【%s】手指向左滚动框架【%s】”类似

@step(u'手指向左边起始位置滚动框架【%s】$' % (mode\_pvs))

和“手指向左边起始位置滚动”类似 增加框架滚动范围

手指向左边起始位置滚动框架【text=短信】

@step(u'手指向右边结束位置滚动框架【%s】$' % (mode\_pvs))

和“手指向左边起始位置滚动框架【%s】”类似

@step(u'手指向上边起始位置滚动框架【%s】$' % (mode\_pvs))

和“手指向左边起始位置滚动框架【%s】”类似

@step(u'手指向下边结束位置滚动框架【%s】$' % (mode\_pvs))

和“手指向左边起始位置滚动框架【%s】”类似

@step(u'手指纵向滚动至页面元素【%s】$' % (mode\_pvs))

手指纵向滚动至页面元素【className=android.widget.TextView】 - 纵向滚动到页面元素className=android.widget.TextView

@step(u'手指横向滚动至页面元素【%s】$' % (mode\_pvs))

和“手指纵向滚动至页面元素【%s】”类似

@step(u'手指纵向滚动至文本【%s】$' % (mode\_text))

和“手指纵向滚动至页面元素【%s】”类似

@step(u'手指横向滚动至文本【%s】$' % (mode\_text))

和“手指纵向滚动至页面元素【%s】”类似

@step(u'手指纵向滚动框架【%s】至页面元素【%s】$' % (mode\_pvs,mode\_pvs))

手指纵向滚动框架【className=android.widget.ListView】至页面元素【text=蓝牙】 - 以className为范围，纵向滚动至text为蓝牙的位置

@step(u'手指横向滚动框架【%s】至页面元素【%s】$' % (mode\_pvs,mode\_pvs))

和“手指纵向滚动框架【%s】至页面元素【%s】”类似

@step(u'手指纵向滚动框架【%s】至文本【%s】$' % (mode\_pvs,mode\_text))

和“手指纵向滚动框架【%s】至页面元素【%s】”类似 - 滚动至文本

@step(u'手指横向滚动框架【%s】至文本【%s】$' % (mode\_pvs,mode\_text))

和“手指纵向滚动框架【%s】至页面元素【%s】”类似

@step(u'手指纵向滚动框架【%s】至文本描述【%s】$' % (mode\_pvs,mode\_text))

和“手指纵向滚动框架【%s】至页面元素【%s】”类似 - 滚动至文本描述

@step(u'手指横向滚动框架【%s】至文本描述【%s】$' % (mode\_pvs,mode\_text))

和“手指纵向滚动框架【%s】至页面元素【%s】”类似

## [预定义步骤之swipe(滑动)](http://10.120.10.100:9002/Test_Knowledge_Share_Zhao_Qi_Android_Lettuce_PDS_Swipe)

可供使用的页面元素属性, 请参考[【WiKi】【2014-08-22】安卓莴苣系统:页面元素属性](http://10.120.10.100:9002/Test_Knowledge_Share_Yu_Kemin_Android_Lettuce_UI_Element_Property)

### swpie.py

@step(u'手指向左滑动屏幕$')

手指向左滑动屏幕 - 从屏幕最右边向最左边滑动，手指轨迹为最右边至最左边

@step(u'手指向右滑动屏幕$')

和“手指向左滑动屏幕”类似

@step(u'手指向上滑动屏幕$')

和“手指向左滑动屏幕”类似

@step(u'手指向下滑动屏幕$')

和“手指向左滑动屏幕”类似

@step(u'手指以步长【%s】向左滑动屏幕$' % (mode\_int))

手指以步长【100】向左滑动屏幕 - 用100步完成向左滑动屏幕的操作

@step(u'手指以步长【%s】向右滑动屏幕$' % (mode\_int))

和“手指以步长【%s】向左滑动屏幕”类似

@step(u'手指以步长【%s】向上滑动屏幕$' % (mode\_int))

和“手指以步长【%s】向左滑动屏幕”类似

@step(u'手指以步长【%s】向下滑动屏幕$' % (mode\_int))

和“手指以步长【%s】向左滑动屏幕”类似

@step(u'手指从坐标【%s】滑动至坐标【%s】$' % (mode\_position, mode\_position))

手指从坐标【100,300】滑动至坐标【400,500】 - 滑动与拖动不同，拖动带有长按的效果，滑动仅代表单纯的屏幕滑动

@step(u'手指以步长【%s】从坐标【%s】滑动至坐标【%s】$' % (mode\_int,mode\_position,mode\_position))

手指以步长【100】从坐标【100,300】滑动至坐标【400,500】 - 用100步完成从坐标点(100,300)滑动至坐标点(400,500)的操作

@step(u'以页面元素【%s】为范围，手指从右向左滑动$' % (mode\_pvs))

以页面元素【className=android.widget.TextView】为范围，手指从右向左滑动 - 以className为范围，区域内滑动

以页面元素【text=相机】为范围，手指从右向左滑动

@step(u'以页面元素【%s】为范围，手指从左向右滑动$' % (mode\_pvs))

和“以页面元素【%s】为范围，手指从右向左滑动”类似

@step(u'以页面元素【%s】为范围，手指从下向上滑动$' % (mode\_pvs))

和“以页面元素【%s】为范围，手指从右向左滑动”类似

@step(u'以页面元素【%s】为范围，手指从上向下滑动$' % (mode\_pvs))

和“以页面元素【%s】为范围，手指从右向左滑动”类似

@step(u'以页面元素【%s】为范围，步长为【%s】手指从右向左滑动$' % (mode\_pvs,mode\_int))

和“以页面元素【%s】为范围，手指从右向左滑动”类似 加入步长条件

@step(u'以页面元素【%s】为范围，步长为【%s】手指从左向右滑动$' % (mode\_pvs,mode\_int))

和“以页面元素【%s】为范围，手指从右向左滑动”类似 加入步长条件

@step(u'以页面元素【%s】为范围，步长为【%s】手指从下向上滑动$' % (mode\_pvs,mode\_int))

和“以页面元素【%s】为范围，手指从右向左滑动”类似 加入步长条件

@step(u'以页面元素【%s】为范围，步长为【%s】手指从上向下滑动$' % (mode\_pvs,mode\_int))

和“以页面元素【%s】为范围，手指从右向左滑动”类似 加入步长条件

@step(u'以文本【%s】为范围，手指从右向左滑动$' % (mode\_text))

以文本【"相机"】为范围，手指从右向左滑动 - 以text为范围

@step(u'以文本【%s】为范围，手指从左向右滑动$' % (mode\_text))

和“以文本【%s】为范围，手指从右向左滑动”类似

@step(u'以文本【%s】为范围，手指从下向上滑动$' % (mode\_text))

和“以文本【%s】为范围，手指从右向左滑动”类似

@step(u'以文本【%s】为范围，手指从上向下滑动$' % (mode\_text))

和“以文本【%s】为范围，手指从右向左滑动”类似

@step(u'以文本描述【%s】为范围，手指从右向左滑动$' % (mode\_text))

以文本描述【"应用"】为范围，手指从右向左滑动 - 以description为范围滑动

@step(u'以文本描述【%s】为范围，手指从左向右滑动$' % (mode\_text))

和“以文本描述【%s】为范围，手指从右向左滑动”类似

@step(u'以文本描述【%s】为范围，手指从下向上滑动$' % (mode\_text))

和“以文本描述【%s】为范围，手指从右向左滑动”类似

@step(u'以文本描述【%s】为范围，手指从上向下滑动$' % (mode\_text))

和“以文本描述【%s】为范围，手指从右向左滑动”类似

## [预定义步骤之assert(断言)](http://10.120.10.100:9002/Test_Knowledge_Share_Zhao_Qi_Android_Lettuce_PDS_Assert)

### assert.py

@step(u'应该可看到页面元素【%s】$' % (mode\_pvs))

应该可看到页面元素【className=android.widget.ImageView】

应该可看到页面元素【text=短信】

应该可看到页面元素【description=应用】 - 如果页面元素存在则pass，不存在则assert

@step(u'应该看不到页面元素【%s】$' % (mode\_pvs))

应该看不到页面元素【text=短信】 - 如果页面元素不存在则pass，存在则assert

@step(u'页面元素【%s】的数量应该为【%s】$' % (mode\_pvs, mode\_int))

页面元素【className=android.widget.ImageView】的数量应该为【5】 - android.widget.ImageView 在界面中的数量应该为5

@step(u'等待更新超时时间【%s】毫秒后，包为【%s】的页面应该更新$' % (mode\_int,mode\_text))

等待更新超时时间【2000】毫秒后，包为【"com.android.mms"】的页面应该更新 - 等待2秒后，com.android.mms的应用界面应该刷新，若没刷新则assert

@step(u'等待，页面元素【%s】应该消失$' % (mode\_pvs))

等待，页面元素【className=android.widget.TextView】应该消失 - 等待页面元素消失，若不消失则assert

@step(u'在【%s】毫秒内，页面元素【%s】应该消失$' % (mode\_int, mode\_pvs))

在【3000】毫秒内，页面元素【text=电话】应该消失 - 3秒内，text为电话的页面元素应该消失

@step(u'等待，页面元素【%s】应该出现$' % (mode\_pvs))

等待，页面元素【text=确定】应该出现 - 等待text为确定的页面元素出现，不出现则assert

@step(u'在【%s】毫秒内，页面元素【%s】应该出现$' % (mode\_int, mode\_pvs))

和“在【%s】毫秒内，页面元素【%s】应该消失”类似 - 此处为应该出现

@step(u'应该可看到文本【%s】$' % (mode\_text))

应该可看到文本【"相册"】 - 在界面上可找到文本为“相册”的页面元素

@step(u'应该看不到文本【%s】$' % (mode\_text))

和“应该可看到文本【%s】”类似，此处为看不到文本

@step(u'应该可找到文本描述【%s】$' % (mode\_text))

和“应该可看到文本【%s】”类似，此处为找得到文本描述

@step(u'应该找不到文本描述【%s】$' % (mode\_text))

和“应该可看到文本【%s】”类似，此处为找不到文本描述

@step(u'等待，文本【%s】应该消失$' % (mode\_text))

等待，文本【"收音机"】应该消失 - 等待文本为“收音机”的页面元素消失，若不消失则assert

@step(u'等待，文本描述【%s】应该找不到$' % (mode\_text))

和“等待，文本【%s】应该消失”类似

@step(u'在【%s】毫秒内，文本【%s】应该消失$' % (mode\_int, mode\_text))

和“等待，文本【%s】应该消失”类似， 此处为等待指定时间后判断

@step(u'在【%s】毫秒内，文本描述【%s】应该找不到$' % (mode\_int, mode\_text))

和“等待，文本描述【%s】应该消失”类似，此处为等待指定时间后判断

@step(u'等待，文本【%s】应该出现$' % (mode\_text))

和“等待，文本【%s】应该消失”类似，此处为出现

@step(u'等待，文本描述【%s】应该被找到$' % (mode\_text))

和“等待，文本描述【%s】应该找不到”类似，此处为找不到

@step(u'在【%s】毫秒内，文本【%s】应该出现$' % (mode\_int, mode\_text))

和“在【%s】毫秒内，文本【%s】应该消失”类似，此处为出现

@step(u'在【%s】毫秒内，文本描述【%s】应该被找到$' % (mode\_int, mode\_text))

和“【%s】毫秒内，文本描述【%s】应该找不到”类似，此处为能被找到

@step(u'图片【%s】与模板图片【%s】对比，误差不大于百分之【%s】$' % (mode\_file,mode\_file,mode\_int))

图片【/tmp/testA.img】与模板图片【/tmp/template.img】对比，误差不大于百分之【1】 - 两张图片进行对比，

误差不大于百分之1即为相似度最小为99%，如果对比结果为98%，则assert

## [预定义步骤之settings(设置)](http://10.120.10.100:9002/Test_Knowledge_Share_Zhao_Qi_Android_Lettuce_PDS_Settings)

### settings.py

@step(u'打开自动旋转屏幕$')

打开自动旋转屏幕 - 将自动旋转屏幕设置为开启

@step(u'关闭自动旋转屏幕$')

同“打开自动旋转屏幕” - 此处为关闭

@step(u'点亮屏幕$')

点亮屏幕 - 将屏幕唤醒

@step(u'熄灭屏幕$')

熄灭屏幕 - 关闭屏幕

@step(u'下载界面结构文件保存为【%s】$' % (mode\_text))

保存在当前lettuce执行目录下,文件名字为layout.xml

下载界面结构文件保存为【layout.xml】

下载界面结构文件保存为【./layout.xml】

保存在当前lettuce执行目录下的tools目录下,文件名字为layout.xml

下载界面结构文件保存为【tools/layout.xml】

下载界面结构文件保存为【./tools/layout.xml】

@step(u'打开通知栏$')

打开通知栏 - 将通知栏打开

@step(u'打开快速设置$')

打开快速设置 - 弹出快速设置界面

@step(u'等待窗口空闲$')

等待窗口空闲 - 等待当前窗口空闲

@step(u'等待窗口空闲超时时间【%s】毫秒$' % (mode\_int))

同“等待窗口空闲” 此处增加空闲超时时间

@step(u'清除页面元素【%s】中的文字$' % (mode\_pvs))

清除页面元素【className=android.widget.EditText】中的文字 - 清除className=android.widget.EditText的文字

@step(u'在页面元素【%s】添加文本【%s】$' % (mode\_pvs, mode\_text))

在页面元素【className=android.widget.EditText】添加文本【"edit\_text"】 - 在className=android.widget.EditText上添加文字edit\_text

@step(u'从两点【%s】和【%s】分别滑动至另外两点【%s】和【%s】$' % (mode\_position,mode\_position,mode\_position,mode\_position))

从两点【100,200】和【200,200】分别滑动至另外两点【400,500】和【300,700】

@step(u'从左右边缘同时滑向中心$')

从左右边缘同时滑向中心

@step(u'从中心同时滑向左右两边$')

从中心同时滑向左右两边

@step(u'分别以左右距边缘位置百分之【%s】处，按照步长【%s】同时滑向中心$' % (mode\_int,mode\_int))

分别以左右距边缘位置百分之【20】处，按照步长【100】同时滑向中心

@step(u'从中心以步长【%s】同时滑向左右距边缘位置百分之【%s】处$' % (mode\_int,mode\_int))、

同“分别以左右距边缘位置百分之【%s】处，按照步长【%s】同时滑向中心” 此处起点为中心

@step(u'以页面元素【%s】为范围，从左右边缘同时滑向中心$' % (mode\_pvs))

以页面元素【className=android.widget.EditText】为范围，从左右边缘同时滑向中心

@step(u'以页面元素【%s】为范围，从中心同时滑向左右两边$' % (mode\_pvs))

同“以页面元素【%s】为范围，从左右边缘同时滑向中心” 此处起点为中心

## 一些有意思的Q&A

### [wipe、scroll、drag、fling怎么区别?](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-07-25%E3%80%91swipe%E3%80%81scroll%E3%80%81drag%E4%BB%A5%E5%8F%8Afling%E7%AE%80%E8%BF%B0)

swipe、scroll、drag以及fling都有滑动的效果在其中，但什么时候该用哪一个操作，怎么区分呢。 简单描述一下各自的特点吧。

1. swipe 滑动，点到点的滑动，在一个界面内手指滑动。
2. scroll 滚动，可在一个框架中连续滚动，特征为连续。
3. drag 拖动，将一个东西从一个地方拖动到另一个地方，带有长按的效果。
4. fling 快速滑动，顾名思义，使用手指滑动，与swipe不同的是，fling是快速的滑动，如何区别呢？

打开【开发工具】中的指针位置，执行fling的脚本，会发现滑动路径的小尾巴上有一条多余的红色线条，再尝试使用 手指在屏幕上快速滑动一下，会发现也会带一个小尾巴。

### [使用lettuce执行脚本，设备名称含有空格不能使用怎么办?](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-07-09%E3%80%91%E4%BD%BF%E7%94%A8lettuce%E6%89%A7%E8%A1%8C%E8%84%9A%E6%9C%AC%EF%BC%8C%E8%AE%BE%E5%A4%87%E5%90%8D%E7%A7%B0%E5%90%AB%E6%9C%89%E7%A9%BA%E6%A0%BC%E4%B8%8D%E8%83%BD%E4%BD%BF%E7%94%A8%E7%9A%84%E8%A7%A3%E5%86%B3%E6%96%B9%E6%B3%95)

#### 问题现象

在使用lettuce执行测试脚本时，使用的设备如果名称是由多个单词或数字组成（类似HIKe 828，HIKe 848） 此时可能会不能执行。

#### 解决方法

解决方法：修改wrapper 初始化代码。

1. 找到wrapper uiautomator路径。 比如：C:\Python27\Lib\site-packages\uiautomator-0.1.30-py2.7.egg\uiautomator

2. 通过编辑工具打开\_\_init\_\_.py

3. 找到文件中291行，即 return dict([s.split() for s in out[index + len(match):].strip().splitlines() if s.strip()])

替换成 return dict([s.split('\t') for s in out[index + len(match):].strip().splitlines() if s.strip()]) 4. 保存退出即可。

### 如果对执行场景有选择的进行, 有什么建议?

#### 概述

lettuce执行时，一个Feature中可能会含有一个Scenario或多个Scenario。 如果只想执行其中指定的场景或者想执行除开某些场景之外的场景时，可以使用到lettuce的TAG功能。

#### 用法

lettuce tag语法： -t TAGS, --tag=TAGS

解释：Tells lettuce to run the specified tags only; can be used multiple times to define more tags(prefixing tags with "-" will exclude them and prefixing with "~" will match approximate words)

#### 示例

1. 现有一个feature文件。

@all

Feature: Test Tag usage

@first\_a

Scenario: print device info a

Given I power on the phone

@second\_b

Scenario: print device info b

Given I power on the phone

@second\_b

Scenario: print device info c

Given I power on the phone

@fourth\_d

Scenario: print device info d

Given I power on the phone

@fifth\_e

Scenario: print device info e

Given I power on the phone

2. 文件中的@first\_a, @second\_b 等等以@开头的皆为设定的tag，可以设定Feature的tag 也可以设定Scenario的tag。

3. 使用lettuce –t first\_a即执行指定tag为first\_a的场景

lettuce -t ~first\_a即只执行tag为first\_a的场景

lettuce –t –first\_a即执行除tag为first\_a以外的场景。

# [Jenkins使用](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-08-22%E3%80%91lettuce%E5%9C%A8Jenkins%E4%B8%AD%E7%9A%84%E4%BD%BF%E7%94%A8)

请参考 [集成到Jenkins执行](http://10.120.10.100:9002/%E3%80%90WiKi%E3%80%91%E3%80%902014-08-22%E3%80%91lettuce%E5%9C%A8Jenkins%E4%B8%AD%E7%9A%84%E4%BD%BF%E7%94%A8)章节

## ubuntu下Jenkins环境搭建

### 下载

到"<http://mirror.xmission.com/jenkins/debian/>"下载需要合适deb

### 安装

#### 安装daemon

sudo apt-get install daemon

#### 安装Jenkins

sudo dpkg -i jenkins\_1.\*\*\*\_all.deb

### 启动

#### 简单启动

安装后在/usr/share/jenkins/ 有 jenkins.war

cd /usr/share/jenkins/

java -jar jenkins.war --httpPort=8000

#### 系统启动

sudo service jenkins stop

sudo service jenkins start

### 端口配置

vi /etc/default/jenkins

找到下面的行改吧

1. port for HTTP connector (default 8080; disable with -1)

HTTP\_PORT=8000

### 构建项目

到　[http://ip:8000](http://ip:8000/) 去构建项目

## Jenkins配置基于角色的项目权限管理

本文将介绍如何配置jenkins，使其可以支持基于角色的项目权限管理。

由于jenkins默认的权限管理体系不支持用户组或角色的配置，因此需要安装第三发插件来支持角色的配置，本文将使用Role Strategy Plugin，介绍页面：<https://wiki.jenkins-ci.org/display/JENKINS/Role+Strategy+Plugin>

### 配置插件

安装插件后，进入系统设置页面，配置如下：

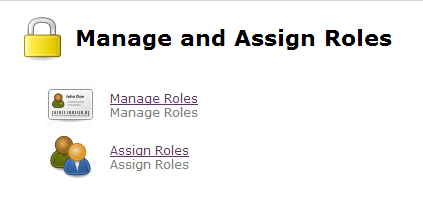
[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_001.png)

官网上安全域设置为Servlet容器代理，实际操作发现Jenkins专有用户数据库也是可以的。

### 配置权限

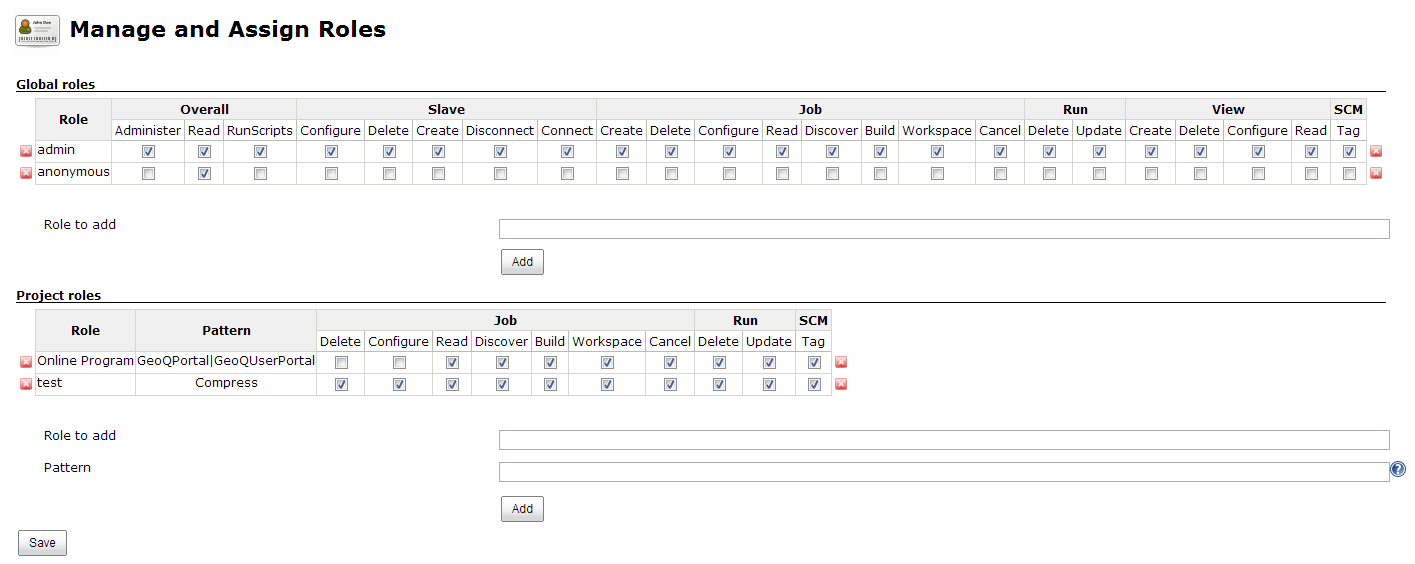
在系统管理页面点击Manage and Assign Roles进入角色管理页面：

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_002.png)

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_003.png)

### 管理角色

选择该项可以创建全局角色、项目角色，并可以为角色分配权限。

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_004.png)

如上图，分别创建了admin、anonymous两个全局角色，Online Program、test两个项目角色。

项目角色与全局角色的区别就是，项目角色只能管理项目，没有管理jenkins的权限配置。

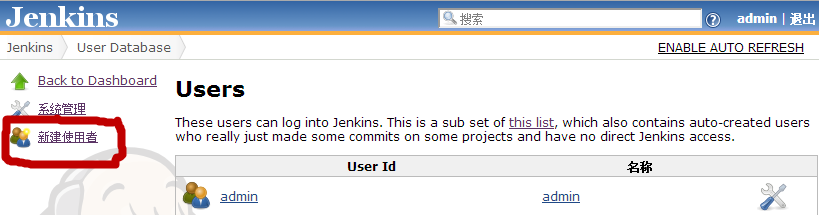
添加项目角色时，需要制定匹配项目的模式，如上图中的Pattern，官方文档介绍该选项支持正则表达式，如“Roger-.”表示所有以Roger-开头的项目，“(?i)roger-.\*”表示以roger-开头的项目并且不区分大小写，如以ABC开头的项目可以配置为“ABC|ABC.\*”，也可以使用“abc|bcd|efg”直接匹配多个项目。

### 创建用户

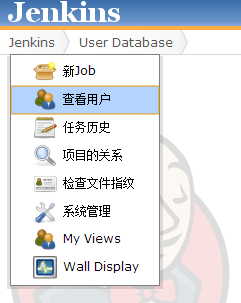
在分配角色之前需要先创建用户。

在系统管理页面，点击管理用户：

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_005.png)

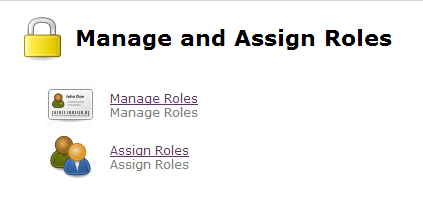
[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_006.png)

点击新建使用者可以创建新用户，如果之前有项目与scm版本管理系统（如svn、git等）连接并获取源码构建过，jenkins会从svn中读取到一些用户信息，可以在查看用户菜单中看到这些用户：

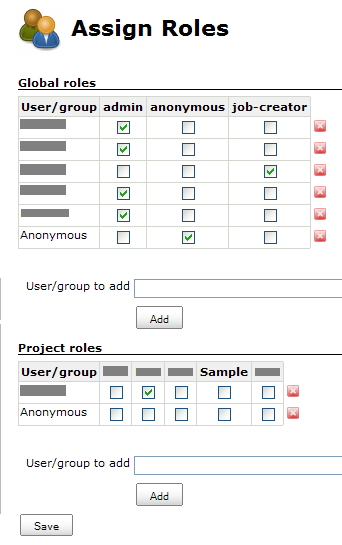
[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_007.png) [](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_008.png)

点击用户id或名称都可以修改用户信息。

### 分配角色

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_009.png)

选择Assign Roles可以为用户分配所属角色，可以分配全局角色和项目角色。

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_010.png)

如上图，将不同的用户分别分配给不同的角色，这样用户就可以具有角色所拥有的权限。

## Jenkins项目管理

### 新建项目

Jenkins首页，点击左侧的新建

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_012.jpg)

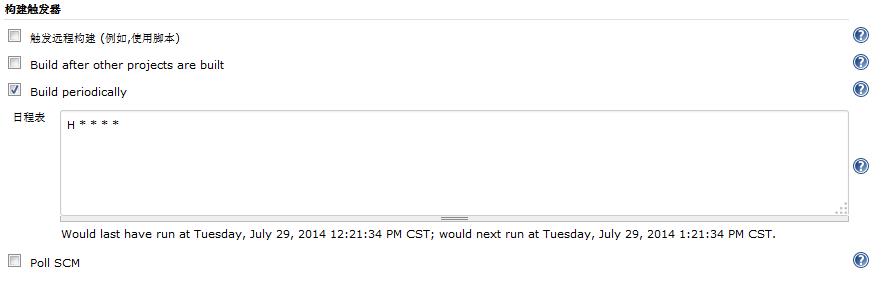
输入项目名称，选择“构建一个自由风格的软件项目”

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_013.jpg)

### 构建触发器

新增一个构建出发器

1. 选中Build periodically
2. 在日程表中输入执行计划，如每小时执行一次：

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_014.jpg)

#### 日程表参数

第一个参数代表的是分钟 minute，取值 0~59；

第二个参数代表的是小时 hour，取值 0~23；

第三个参数代表的是天 day，取值 1~31；

第四个参数代表的是月 month，取值 1~12；

最后一个参数代表的是星期 week，取值 0~7，0 和 7 都是表示星期天。

### 构建

增加一个Shell构建步骤：

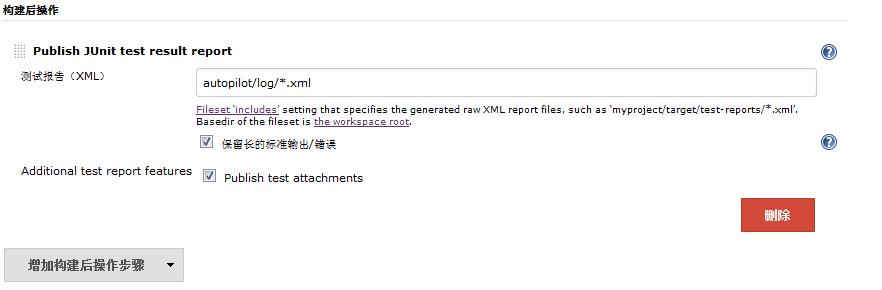
1. 点击增加构建步骤
2. 点击Execute Shell
3. 在Command中输入需要执行的shell脚本，如下图：

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_015.jpg)

### 构建后操作

通过构建后操作处理测试结果

1. 点击增加构建后操作步骤
2. 选择Publish JUnit test result report
3. 测试报告（XML）输入测试报告保存地址，如autopilot/log下的所有测试报告

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Jenkins_016.jpg)

# 附录

## 在Windows环境下安装Android Lettuce的支持环境

获取python, setuptool, pip, lettuce, uiautomator python wrapper等lettuce安装包，取下来解压可以得到下面的所有安装文件

1. 安装python

直接点击安装文件安装， 添加(C:\Python27)这样的路径到环境变量中， cmd中运行 python -V查看是否安装完成

2. 安装setuptool

直接点击安装文件安装

3. 安装pip

在pip-1.5.6 找到setup.py,cmd中使用python setup.py install

4. 安装lettuce

进入pip安装路径，找到pip.exe文件，然后在cmd中运行pip install lettuce

5. 安装wrapper

5.1 jsonrpclib-0.1.3

到解压目录下找到setup.py,cmd中运行python setup.py install

5.2 uiautomator-master

到解压目录下找到setup.py,cmd中运行python setup.py install

5.3 urllib3-1.8.2

到解压目录下找到setup.py,cmd中运行python setup.py install

6. 添加环境变量

如C:\Python27\lib\site-packages;C:\Python27\Scripts的路径到系统环境变量中

7. 验证lettuce是否可用

cmd中运行lettuce -h

## Ubuntu虚拟机不能识别Android设备的解决方法

如果Ubuntu虚拟机不能识别USB设备，请尝试

虚拟机为Virtual Box

保证虚拟机提供对设备的访问，比如：虚拟机界面窗口右下键状态栏，USB图标右键后，选择你想访问的设备

### 方法1： 添加设备号

1. 获取设备ID

lsusb找到设备ID

root@ckt:/home/ckt# lsusb

Bus 001 Device 006: ID 2a47:0c02 ==> 我的手机设备

Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet

Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

2a47是idVendor，80ee是idProduct

1. 注册到udev的rules文件

如果没有此文件，请自己创建

vim /etc/udev/rules.d/51-android.rules

SUBSYSTEM=="usb", ATTR{idVendor}="2a47", MODE="0666"

保存文件并赋予权限：

sudo chmod a+rx /etc/udev/rules.d/51-android.rules

重启udev服务

sudo /etc/init.d/udev restart

udev stop/waiting

udev start/running, process 14636

如果想添加的更详细，也可以这样：SUBSYSTEM=="usb", ATTRS{idVendor}=="2a47", ATTRS{idProduct}=="0c02",MODE="0666"

1. 启动adb的服务

重启adb服务

cd <sdk home>/platform-tools

sudo ./adb kill-server

sudo ./adb start-server

./adb devices

### 方法2： 添加通用的USB设备

1. 注册通用USB设备到udev的rules文件

如果没有此文件，请自己创建

vim /etc/udev/rules.d/51-android.rules

SYSTEM=="usb", ENV{DEVTYPE}=="usb\_device", MODE="0666"

保存文件并赋予权限：

sudo chmod a+rx /etc/udev/rules.d/51-android.rules

重启udev服务

sudo /etc/init.d/udev restart

udev stop/waiting

udev start/running, process 14636

1. 启动adb的服务

重启adb服务

cd <sdk home>/platform-tools

sudo ./adb kill-server

sudo ./adb start-server

./adb devices

## Ubuntu adb不能识别Android设备的解决方法

1. 如果Ubuntu虚拟机中还没有adb和phablet工具环境，请尝试

apt-get install phablet-tools

1. 如果安装失败，请尝试

ubuntu touch默认readonly mount，需要apt-get install安装包，请执行：

adb shell

mount -o rw,remount /

1. 如果在已经可以看见usb设备情况下，adb devices不能访问手机设备，请尝试

在ubuntu home目录（你用户的主目录）下

mkdir .android

vim .android/adb\_usb.ini

在adb\_usb.ini中想下面这样输入对应的设备号(可以用lsusb来查看)。

root@ckt:~/.android# cat adb\_usb.ini

0x2a47

设备号可以用lsusb来查看

Bus 001 Device 006: ID 2a47:0c02

==== 填此信息: 0x2a47

重启adb服务

adb kill-server

adb start-server

之后就可以用adb devices使用了