# UiAutomator2 入门

## 一、关于

[UiAutomator](https://developer.android.com/training/testing/ui-automator.html)是Google提供的用来做安卓自动化测试的一个Java库，基于Accessibility服务。功能很强，可以对第三方App进行测试，获取屏幕上任意一个APP的任意一个控件属性，并对其进行任意操作，但有两个缺点：

* 测试脚本只能使用Java语言
* 测试脚本要打包成jar或者apk包上传并安装到设备上才能运行

为简化操作步骤并实现联机操作，因此有了Python uiautomator库，原理是在手机上运行了一个http rpc服务，将uiautomator中的功能开放出来，然后再将这些http接口封装成库。

UiAutomator2库从xiaocong/uiautomator进化升级而来，对原有的库的bug进行了修复，还增加了很多新的Feature。主要有以下部分：

* 设备和开发机可以脱离数据线，通过WiFi互联（基于[atx-agent](https://github.com/openatx/atx-agent)）
* 集成了[openstf/minicap](https://github.com/openstf/minicap)达到实时屏幕投频，以及实时截图
* 集成了[openstf/minitouch](https://github.com/openstf/minitouch)达到精确实时控制设备
* 修复了[xiaocong/uiautomator](https://github.com/xiaocong/uiautomator)经常性退出的问题
* 代码进行了重构和精简，方便维护
* 实现了一个设备管理平台(也支持iOS) [atxserver2](https://github.com/openatx/atxserver2)
* 扩充了toast获取和展示的功能

uiautomator2 不支持iOS测试，需要iOS自动化测试，可以转到这个库 [openatx/facebook-wda](https://github.com/openatx/facebook-wda)。

库 [~~https://github.com/NeteaseGame/ATX~~](https://github.com/NeteaseGame/ATX) 已不再维护

## 二、安装

### Requirements

* Android 4.4+
* Python 3.6+
* UiAutomator2

如果用Python2的pip安装，会安装本库的老版本0.2.3；如果用Python3.5的pip安装，会安装本库的老版本0.3.3；两者均已经不再维护

### 安装方法

[Python官方下载地址](https://www.python.org/downloads/)

Windows平台可在Powershell中输入python启动Microsoft Store一键安装

在使用pip安装UiAutomator2时若网速较慢，建议将pip安装源更换为如下地址，更换方式可参考[PIP 更换国内安装源](https://blog.csdn.net/yuzaipiaofei/article/details/80891108)：

* 阿里云 <http://mirrors.aliyun.com/pypi/simple/>
* 中国科技大学 <https://pypi.mirrors.ustc.edu.cn/simple/>
* 豆瓣 <http://pypi.douban.com/simple/>
* 清华大学 <https://pypi.tuna.tsinghua.edu.cn/simple/>
* 中国科学技术大学 <http://pypi.mirrors.ustc.edu.cn/simple/>

### 完成

准备1台开启了开发者选项的安卓手机，连接上电脑，确保执行adb devices可以看到连接上的设备。

运行pip install -U uiautomator2 安装uiautomator2

运行python -m uiautomator2 init安装包含httprpc服务的apk到手机atx-agent, minicap, minitouch （在过去的版本中，这一步是必须执行的，但是从1.3.0之后的版本，当运行python代码u2.connect()时就会自动安装了）

命令行运行python打开python交互窗口。然后将下面的命令输入到窗口中。

import uiautomator2 as u2
  
  
d = u2.connect() # connect to device
  
print(d.info)

这时看到类似下面的输出，就说明已经成功安装所需的库了。

{'currentPackageName': 'net.oneplus.launcher', 'displayHeight': 1920, 'displayRotation': 0, 'displaySizeDpX': 411, 'displaySizeDpY': 731, 'displayWidth': 1080, 'productName': 'OnePlus5', '  
screenOn': True, 'sdkInt': 27, 'naturalOrientation': True}

### 辅助工具

* uiautomatorviewer
* 这是Android SDK自带的工具，用于获取AndroidUI中的控件属性。在使用UiAutomator时，由于安卓端的uiautomator是独占资源，所以两者无法同时使用，推荐使用下面的工具
* weditor
* 基于Python，使用pip安装，可与UiAutomator更好融合。
* 安装方法：
* pip install -U weditor
* 安装好之后，就可以在命令行运行weditor --help 确认是否安装成功了。
* Windows系统可以使用命令在桌面创建一个快捷方式 weditor --shortcut，如果使用的是conda环境则快捷方式无效
* 命令行直接输入 weditor 会自动打开浏览器，输入设备的ip或者序列号，点击Connect即可。
* 参考文章：[浅谈自动化测试工具python-uiautomator2](https://testerhome.com/topics/11357)

## 三、使用方法详解

UiAutomator2可搭配使用Python unittest单元测试框架使用，使用方法可参考[unittest --- 单元测试框架 — Python 3.11.1 文档](https://docs.python.org/zh-cn/3/library/unittest.html)

### 3.1 与设备连接

1. USB线缆

* 假设设备序列号是 123456f (通过 adb devices 命令查看)
* import uiautomator2 as u2  
    
  d = u2.connect('123456f') # alias for u2.connect\_usb('123456f')  
  print(d.info)

1. Wi-Fi

* 假设设备IP是 10.0.0.1 并与计算机在同一局域网下
* import uiautomator2 as u2  
    
  d = u2.connect('10.0.0.1') # alias for u2.connect\_wifi('10.0.0.1')  
  print(d.info)

1. Wi-Fi ADB

* 需要在开发者选项中打开网络调试，此处假设设备IP是10.0.0.1，adb调试端口是5555
* import uiautomator2 as u2  
    
  d = u2.connect\_adb\_wifi("10.0.0.1:5555")  
    
  # 相当于  
  # + Shell: adb connect 10.0.0.1:5555  
  # + Python: u2.connect\_usb("10.0.0.1:5555")
* 若不添加参数地调用u2.connect()，UiAutomator2会尝试从环境变量ANDROID\_DEVICE\_IP 或 ANDROID\_SERIAL 中获取设备序列号。  
  If this environment variable is empty, uiautomator will fall back to connect\_usb and you need to make sure that there is only one device connected to the computer.

### 3.2 熟悉Android常用UI控件

### 3.3 通过坐标的手势交互

| 手势 | 函数调用 | 必选参数 | 可选参数 | 示例 | 补充 |
| --- | --- | --- | --- | --- | --- |
| 点击 | d.click() | x, y（px） |  | d.click(50, 100) | 物理像素点定位 |
|  |  | %x, %y （%） |  | d.click(0.1, 0.2) | 百分比定位，基于屏幕大小 |
| 双击 | d.double\_click() | x,y (px/%) | duration | d.double\_click(50, 100, 0.2) | 默认点击间隔为100ms |
| 长按 | d.long\_click() | x,y (px/%) | duration | d.long\_click(50, 100, duration=0.5) | 默认持续时间为500ms |
| 滑动 | d.swipe() | startX,startY,endX,endY | duration | 同上 | 默认持续时间为500ms |
| 滑动扩展功能 | d.swipe\_ext() | "right" / "up" / "left" / "down" |  | d.swipe\_ext("up") | 上划 |
|  |  |  | scale | d.swipe\_ext("up", scale=0.9) | 默认滑动距离为屏幕高度（宽度）的90% |
|  |  |  | box | d.swipe\_ext("up", box=(0,0,100,100)) | 在 (0,0) -> (100, 100) 这个区域做滑动 |
| 按住并拖动 | d.drag() | startX,startY,endX,endY | duration | d.drag(0,0,100,100,duration=0.5)``d.drag(0,0,100,100) | 默认执行时长为0.5秒 |
| 九宫格解锁 | d.swipe\_points() | 由点坐标（元组/列表，px/%）组成的列表 | duration | d.swipe\_points([(0, 0), (100, 100), (200, 200)], 0.5)) | 默认执行时长为0.5秒 |

### 3.4 控件定位

#### 支持的选择器

ui2支持 android 中 UiSelector 类中的所有定位方式，详细可以在这个网址查看<https://developer.android.com/reference/android/support/test/uiautomator/UiSelector>

整体内容如下,所有的属性可以通过weditor查看到。

| 名称 | 描述 |
| --- | --- |
| text | text是指定文本的元素 |
| textContains | text中包含有指定文本的元素 |
| textMatches | text符合指定正则的元素 |
| textStartsWith | text以指定文本开头的元素 |
| className | className是指定类名的元素 |
| classNameMatches | className类名符合指定正则的元素 |
| description | description是指定文本的元素 |
| descriptionContains | description中包含有指定文本的元素 |
| descriptionMatches | description符合指定正则的元素 |
| descriptionStartsWith | description以指定文本开头的元素 |
| checkable | 可检查的元素，参数为True，False |
| checked | 已选中的元素，通常用于复选框，参数为True，False |
| clickable | 可点击的元素，参数为True，False |
| longClickable | 可长按的元素，参数为True，False |
| scrollable | 可滚动的元素，参数为True，False |
| enabled | 已激活的元素，参数为True，False |
| focusable | 可聚焦的元素，参数为True，False |
| focused | 获得了焦点的元素，参数为True，False |
| selected | 当前选中的元素，参数为True，False |
| packageName | packageName为指定包名的元素 |
| packageNameMatches | packageName为符合正则的元素 |
| resourceId | resourceId为指定内容的元素 |
| resourceIdMatches | resourceId为符合指定正则的元素 |

#### 选择器定位示例

element = d(上述列表中的属性=字符串值)

例：element = d(resourceId="com.google.android.dialer:id/dialpad\_fab")

返回类型为 <class 'uiautomator2.\_selector.UiObject'>

例如<uiautomator2.\_selector.UiObject object at 0x000001D938F596D0> (即使找不到元素也返回值)

#### 判断控件是否存在

d(上述列表中的属性=字符串值).exists()

返回类型为bool型

### 3.5 高级控件定位

#### 子元素定位

d(上述列表中的属性=字符串值).child()

d(上述列表中的属性=字符串值).child\_by\_text()

d(上述列表中的属性=字符串值).child\_by\_description()

d(上述列表中的属性=字符串值).child\_by\_instance()

例：

# 查找类名为android.widget.ListView下的Bluetooth元素  
d(className="android.widget.ListView").child(text="Bluetooth")

child\_by\_description is to find children whose grandchildren have  
the specified description, other parameters being similar to child\_by\_text.

child\_by\_instance is to find children with has a child UI element anywhere  
within its sub hierarchy that is at the instance specified. It is performed  
on visible views **without scrolling**.

#### 兄弟元素定位

d(上述列表中的属性=字符串值).sibling()

#查找与google同一级别，类名为android.widget.ImageView的元素  
d(text="Google").sibling(className="android.widget.ImageView")

#### 相对位置定位

相对定位支持left, right, top, bottom即在某个元素的前后左右

d(A).left(B) # 选择A左边的B  
d(A).right(B) # 选择A右边的B  
d(A).up(B) # 选择A上边的B  
d(A).down(B) # 选择A下边的B  
  
# 选择 WIFI 右边的开关按钮  
d(text='Wi‑Fi').right(resourceId='android:id/widget\_frame')

#### 多属性定位

在定位控件时添加多个属性参数即可

d(className="android.widget.ImageView", resourceId="xxxxxxxx")

#### 链式定位

d(className="android.widget.ListView", resourceId="android:id/list") \  
 .child\_by\_text("Wi‑Fi", className="android.widget.LinearLayout") \  
 .child(className="android.widget.Switch") \  
 .click()

#### 多个相同属性控件定位

有时屏幕可能包含多个具有相同属性的控件，例如text属性，那么必须使用选择器中的 "instance"属性来选择一个合格的实例，如下所示。

d(text="Add new", instance=0) # which means the first instance with text "Add new"

此外，uiautomator2提供了一个类似列表的API（类似于jQuery）：

# get the count of views with text "Add new" on current screen  
d(text="Add new").count  
  
# same as count property  
len(d(text="Add new"))  
  
# get the instance via index  
d(text="Add new")[0]  
d(text="Add new")[1]  
...  
  
# iterator  
for view in d(text="Add new"):  
 view.info

**Notes**: when using selectors in a code block that walk through the result list, you must ensure that the UI elements on the screen keep unchanged. Otherwise, when Element-Not-Found error could occur when iterating through the list.

#### XPATH定位

Java uiautoamtor中默认是不支持xpath的，所以这里属于扩展的一个功能。速度不是这么的快。

详情见<XPATH.md>

For example: 其中一个节点的内容

<android.widget.TextView  
 index="2"  
 text="05:19"  
 resource-id="com.netease.cloudmusic:id/qf"  
 package="com.netease.cloudmusic"  
 content-desc=""  
 checkable="false" checked="false" clickable="false" enabled="true" focusable="false" focused="false"  
 scrollable="false" long-clickable="false" password="false" selected="false" visible-to-user="true"  
 bounds="[957,1602][1020,1636]" />

xpath定位和使用方法

\*有些属性的名字有修改需要注意

description -> content-desc  
resourceId -> resource-id

常见用法

# wait exists 10s  
d.xpath("//android.widget.TextView").wait(10.0)  
  
# find and click  
d.xpath("//\*[@content-desc='分享']").click()  
  
# check exists  
if d.xpath("//android.widget.TextView[contains(@text, 'Se')]").exists:  
 print("exists")  
   
# get all text-view text, attrib and center point  
for elem in d.xpath("//android.widget.TextView").all():  
 print("Text:", elem.text)  
 # Dictionary eg:   
 # {'index': '1', 'text': '999+', 'resource-id': 'com.netease.cloudmusic:id/qb', ......}  
 print("Attrib:", elem.attrib)  
 # Coordinate eg: (100, 200)  
 print("Position:", elem.center())

### 3.6 控件交互

#### 获取控件信息

| 方法 | 描述 | 返回类型 | 备注 | 示例 |
| --- | --- | --- | --- | --- |
| exists() | 判断元素是否存在 | bool | @property | e\_ex = d(text="abc").exists |
| info() | 返回元素的所有信息 | dict | @property | e\_info = d(text="abc").info |
| get\_text() | 返回元素文本 | str |  | e\_text = d(text="abc").get\_text() |
| set\_text("text") | 设置元素文本 | None |  | d(text="abc").set\_text("text") |
| clear\_text() | 清空元素文本 | None |  | d(text="abc").clear\_text() |
| center() | 返回元素的中心点位置 | (x,y) | 基于整个屏幕的点 | e\_center = d(text="abc").center() |
| screenshot() | 获取控件的屏幕截图 |  |  | im = d(text="Settings").screenshot() im.save("settings.jpg") |

@property表示将方法视为属性，可直接获取，例如d(text="abc").exists 而非 d(text="abc").exists()

#### 点击控件

d(text="Settings").click()

| 方法 | 参数(方括号表示可选) | 返回值 | 示例 | 解释 |
| --- | --- | --- | --- | --- |
| click() | [timeout] |  | d(text="Settings").click(timeout=10) | 在10秒内等待控件出现。默认为0秒 |
|  | [offset] |  | d(text="Settings").click(offset=(0.5, 0.5)) | 参数值为二元组，表示点击的点在相对控件左上角的水平、垂直偏移值（百分比）。默认为控件中心点（0.5, 0.5） |
|  |  |  | d(text="Settings").click(offset=(0, 1)) | 点击Settings控件的右上角 |
| click\_exists() | [timeout] |  | d(text='Skip').click\_exists(timeout=10.0) | 在10秒内存在时点击，默认超时0秒 |
| click\_gone() | [maxentry] | bool | d(text="Skip").click\_gone(maxretry=10, interval=1.0) | 不断点击直到消失。maxentry为点击次数 |
|  | [interval] |  |  | interval为点击间隔 |
| long\_click() | [duration] |  | d(text="Settings").long\_click(duration=3) | 按下的时长 |
|  | [timeout] |  | d(text="Settings").long\_click(timeout=3) | 等待控件出现的时长 |

#### 控件的手势交互

**拖拽**

将Settings拖拽到屏幕上(x, y)点处，拖拽时长为0.5s:

d(text="Settings").drag\_to(x, y, duration=0.5)

将Settings拖拽到Clock的中点处，拖拽时长为0.25s：

d(text="Settings").drag\_to(text="Clock", duration=0.25)

**滑动**

滑动支持四个方向,即上下左右（right/left/up/down）。

1 steps 大约是 5ms, 所以 20 steps 大约是 0.1s

d(text="Settings").swipe("right")  
d(text="Settings").swipe("left", steps=10)

**Two-point gesture from one point to another**

d(text="Settings").gesture((sx1, sy1), (sx2, sy2), (ex1, ey1), (ex2, ey2))

**双指放大/缩小**

支持两种手势:

* In, 缩小
* Out, 放大

# from edge to center.  
d(text="Settings").pinch\_in(percent=100, steps=10)  
# from center to edge  
d(text="Settings").pinch\_out()

注意：pinch操作从Android 4.3开始支持

**等待，直到控件出现/消失**

# wait until the ui object appears  
d(text="Settings").wait(timeout=3.0) # return bool  
# wait until the ui object gone  
d(text="Settings").wait\_gone(timeout=1.0)

默认等待时间是20s，详情见**global settings**

**滚动**

Possible properties:

* horiz(水平滚动) or vert(垂直滚动)
* forward or backward or toBeginning or toEnd

# fling forward(default) vertically(default)   
d(scrollable=True).fling()  
# fling forward horizontally  
d(scrollable=True).fling.horiz.forward()  
# fling backward vertically  
d(scrollable=True).fling.vert.backward()  
# fling to beginning horizontally  
d(scrollable=True).fling.horiz.toBeginning(max\_swipes=1000)  
# fling to end vertically  
d(scrollable=True).fling.toEnd()

### 3.7 物理按键

#### 按键

d.press("home") # 按”主页“键  
d.press(0x07, 0x02) # press keycode 0x07('0') with META ALT(0x02)

按键定义参见：[Android KeyEvnet(可能需要魔法上网)](https://developer.android.com/reference/android/view/KeyEvent.html)

#### 目前可用的参数列表

* home
* back
* menu
* recent (recent apps)
* left
* right
* up
* down
* center
* search
* enter
* delete ( or del)
* volume\_up
* volume\_down
* volume\_mute
* camera
* power

### 3.8 Watcher

顾名思义，watcher是一个观察者，它可以一直检测设备页面中的变化，比如某个文本/描述/类名等的**出现**或者**改变**，然后进行下一步的点击/按钮操作，通常用于关闭无法预测的弹窗。

#### 注册watcher

注册一个**名字为**“WATCHER\_NAME”的watcher，**当存在**UiSelector满足“text='确认'”的时候，**执行**点击操作

d.watcher("WATCHER\_NAME").when(text="确认").click(text="确认")  
# Same as  
d.watcher("ALERT").when(text="OK").click()

注册完watcher后它并没有运行，没有任何效果。

例子：

# 常用写法，注册匿名监控  
d.watcher.when("安装").click()

# 注册名为ANR的监控，当出现ANR和Force Close时，点击Force Close  
d.watcher("ANR").when(xpath="ANR").when("Force Close").click()

# 其他回调例子  
d.watcher.when("抢红包").press("back")  
d.watcher.when("//\*[@text = 'Out of memory']").call(lambda d: d.shell('am force-stop com.im.qq'))

# 回调说明  
def click\_callback(d: u2.Device):  
 d.xpath("确定").click() # 在回调中调用不会再次触发watcher

d.xpath("继续").click() # 使用d.xpath检查元素的时候，会触发watcher（目前最多触发5次）

#### 启动watcher

| 方法 | 参数 | 示例 | 备注 |
| --- | --- | --- | --- |
| d.watcher.start() | interval: float = 2.0 | d.watcher.start(2.0) | 默认监控间隔2.0s |
| d.watcher.run() |  |  | 强制启动所有watcher |

所有watcher将被开启，并**一直运行在手机后台**，在你不想使用它的时候，需要使用*d.watchers.watched = False*手动关闭；当然，也可以使用**remove**方法移除某个watcher，如此它便不会再生效

watcher的实现原理：目前采用了后台运行了一个线程的方法(依赖threading库），然后每隔一段时间dump一次hierarchy，匹配到元素之后执行相应的操作。

#### 移除watcher

| 方法 | 参数 | 示例 | 备注 |
| --- | --- | --- | --- |
| d.watcher.remove() | name: str | d.watcher.remove("ANR") | 移除名为ANR的watcher |
|  | None | d.watcher.remove() | 移除所有watcher |
| d.watcher.stop() | 无 |  | 停止监控 |
| d.watcher.reset() | 无 |  | 停止并移除所有监控 |

### 3.9 执行Shell命令

#### 执行阻塞命令

| 方法 | 返回值 | 参数 | 示例 | 备注 |
| --- | --- | --- | --- | --- |
| d.shell() | output, exit\_code 输出, 返回代码 | cmdargs: Union[str, List[str]] | d.shell("pwd") | 命令行，可以是字符串或字符串数组 |
|  |  | stream: bool = False | d.shell("pwd", stream=True) | 输入输出流的形式输出，缺省值False |
|  |  | timeout: int = 60 | d.shell("pwd", timeout=60) | 超时，缺省值60s |

Note: timeout support require atx-agent >=0.3.3

例子：

output, exit\_code = d.shell("pwd", timeout=60) # timeout 60s (Default)  
# output: "/\n", exit\_code: 0  
# Similar to command: adb shell pwd  
  
# Since `shell` function return type is `namedtuple("ShellResponse", ("output", "exit\_code"))`  
# so we can do some tricks  
output = d.shell("pwd").output  
exit\_code = d.shell("pwd").exit\_code

The first argument can be list. for example

output, exit\_code = d.shell(["ls", "-l"])  
# output: "/....", exit\_code: 0

此函数返回stdout与stderr合并的字符串。  
如果命令是阻塞命令，shell也会阻塞直到命令完成或超时生效。在执行命令期间不会接收到部分输出。这个API不适合长时间运行的命令。给出的shell命令在类似于adb shell的环境中运行，其Linux权限级别为adb或shell(高于应用程序权限)。

#### 执行非阻塞命令

添加参数 stream=True 会返回 requests.models.Response 对象. 更多详情： [Requests stream](http://docs.python-requests.org/zh_CN/latest/user/quickstart.html#id5)

r = d.shell("logcat", stream=True)  
deadline = time.time() + 10 # run maxium 10s  
try:  
 for line in r.iter\_lines(): # r.iter\_lines(chunk\_size=512, decode\_unicode=None, delimiter=None)  
 if time.time() > deadline:  
 break  
 print("Read:", line.decode('utf-8'))  
finally:  
 r.close() # this method must be called

要结束命令，使用 r.close() .

### 3.10 获取设备信息

| 方法 | 返回类型: 返回值 |
| --- | --- |
| d.info | dict: 包名、屏幕分辨率、屏幕旋转、屏幕状态、制造商、SDK版本等 |
| d.device\_info | dict: 设备信息、SDK、分辨率、电池、内存、存储、CPU等 |
| d.window\_size | tuple: 屏幕分辨率 |
| d.wlan\_ip | str: IP地址 |
| d.app\_current() | dict: 前台应用信息 |
| d.serial | str: 设备序列号 |

### 3.11 控制屏幕

#### 灭屏/亮屏

d.screen\_on() # turn on the screen  
d.screen\_off() # turn off the screen

#### 锁定/解锁

x d.unlock()  
# This is equivalent to  
# 1. launch activity: com.github.uiautomator.ACTION\_IDENTIFY  
# 2. press the "home" key

#### 获取屏幕状态

d.info.get('screenOn')

仅在Android >= 4.4版本上支持

#### 屏幕旋转

设置旋转方向：

The possible orientations:

* natural or n
* left or l
* right or r
* upsidedown or u (can not be set)

# retrieve orientation. the output could be "natural" or "left" or "right" or "upsidedown"  
orientation = d.orientation  
  
# WARNING: not pass testing in my TT-M1  
# set orientation and freeze rotation.  
# notes: setting "upsidedown" requires Android>=4.3.  
d.set\_orientation('l') # or "left"  
d.set\_orientation("l") # or "left"  
d.set\_orientation("r") # or "right"  
d.set\_orientation("n") # or "natural"

控制自动旋转：

# freeze rotation  
d.freeze\_rotation()  
# un-freeze rotation  
d.freeze\_rotation(False)

### 3.12 应用管理

#### 安装

仅支持从URL链接安装apk

d.app\_install('http://some-domain.com/some.apk')

#### 启动应用

# 默认的这种方法是先通过atx-agent解析apk包的mainActivity，然后调用am start -n $package/$activity启动  
d.app\_start("com.example.hello\_world")  
  
# 使用 monkey -p com.example.hello\_world -c android.intent.category.LAUNCHER 1 启动  
# 这种方法有个副作用，它自动会将手机的旋转锁定给关掉  
d.app\_start("com.example.hello\_world", use\_monkey=True) # start with package name  
  
# 通过指定main activity的方式启动应用，等价于调用am start -n com.example.hello\_world/.MainActivity  
d.app\_start("com.example.hello\_world", ".MainActivity")

#### 停止应用

# equivalent to `am force-stop`, thus you could lose data  
d.app\_stop("com.example.hello\_world")   
# equivalent to `pm clear`  
d.app\_clear('com.example.hello\_world')

#### 停止所有应用

# stop all  
d.app\_stop\_all()  
# stop all app except for com.examples.demo  
d.app\_stop\_all(excludes=['com.examples.demo'])

#### 获取应用信息

d.app\_info("com.examples.demo")
  
# expect output
  
#{
  
# "mainActivity": "com.github.uiautomator.MainActivity",
  
# "label": "ATX",
  
# "versionName": "1.1.7",
  
# "versionCode": 1001007,
  
# "size":1760809
  
#}
  
  
# save app icon
  
img = d.app\_icon("com.examples.demo")
  
img.save("icon.png")

#### 列出所有正在运行的应用

d.app\_list\_running()  
# expect output  
# ["com.xxxx.xxxx", "com.github.uiautomator", "xxxx"]

#### 等待应用启动

pid = d.app\_wait("com.example.android") # 等待应用运行, return pid(int)
  
if not pid:
  
 print("com.example.android is not running")
  
else:
  
 print("com.example.android pid is %d" % pid)
  
  
d.app\_wait("com.example.android", front=True) # 等待应用前台运行
  
d.app\_wait("com.example.android", timeout=20.0) # 最长等待时间20s（默认）

Added in version 1.2.0

#### 检查并维持设备端守护进程处于运行状态

d.healthcheck()

#### Open Scheme

You can do it wire adb: adb shell am start -a android.intent.action.VIEW -d "appname://appnamehost"

Also you can do it with python code

d.open\_url("https://www.baidu.com")  
d.open\_url("taobao://taobao.com") # open Taobao app  
d.open\_url("appname://appnamehost")

### 3.13 杂项

#### 屏幕截图

adb提供的命令：

adb shell screencap /sdcard/screenshot.png

UiAutomator2提供的接口：

# take screenshot and save to a file on the computer, require Android>=4.2.
  
d.screenshot("home.jpg")
  
  
# get PIL.Image formatted images. Naturally, you need pillow installed first
  
image = d.screenshot() # default format="pillow"
  
image.save("home.jpg") # or home.png. Currently, only png and jpg are supported
  
  
# get opencv formatted images. Naturally, you need numpy and cv2 installed first
  
import cv2
  
image = d.screenshot(format='opencv')
  
cv2.imwrite('home.jpg', image)
  
  
# get raw jpeg data
  
imagebin = d.screenshot(format='raw')
  
open("some.jpg", "wb").write(imagebin)

#### 屏幕录制

adb提供的命令：

adb shell screenrecord /sdcard/screenrecore.mp4

U2没有使用上述screenrecord命令，是通过获取手机图片合成视频的方法，所以需要安装一些其他的依赖，如imageio, imageio-ffmpeg, numpy等  
因为有些依赖比较大，推荐使用镜像安装。直接运行下面的命令即可。

pip3 install -U "uiautomator2[image]" -i https://pypi.doubanio.com/simple

使用方法

d.screenrecord('output.mp4')  
  
time.sleep(10)  
# or do something else  
  
d.screenrecord.stop() # 停止录制后，output.mp4文件才能打开

录制的时候也可以指定fps（当前是20），这个值是略低于minicap输出图片的速度，感觉已经很好了，不建议你修改。

#### Toast

Show Toast

d.toast.show("Hello world")  
d.toast.show("Hello world", 1.0) # show for 1.0s, default 1.0s

Get Toast

# [Args]  
# 5.0: max wait timeout. Default 10.0  
# 10.0: cache time. return cache toast if already toast already show up in recent 10 seconds. Default 10.0 (Maybe change in the furture)  
# "default message": return if no toast finally get. Default None  
d.toast.get\_message(5.0, 10.0, "default message")  
  
# common usage  
assert "Short message" in d.toast.get\_message(5.0, default="")  
  
# clear cached toast  
d.toast.reset()  
# Now d.toast.get\_message(0) is None

#### 获取UI hierarchy

xml = d.dump\_hierarchy() # get the UI hierarchy dump content (unicoded).

#### 文件存取

Push a file to the device

# push to a folder
  
d.push("foo.txt", "/sdcard/")
  
# push and rename
  
d.push("foo.txt", "/sdcard/bar.txt")
  
# push fileobj
  
with open("foo.txt", 'rb') as f:
  
 d.push(f, "/sdcard/")
  
# push and change file access mode
  
d.push("foo.sh", "/data/local/tmp/", mode=0o755)

Pull a file from the device

d.pull("/sdcard/tmp.txt", "tmp.txt")
  
  
# FileNotFoundError will raise if the file is not found on the device
  
d.pull("/sdcard/some-file-not-exists.txt", "tmp.txt")