Your browser **doesn't support the features required** by impress.js, so you are presented with a simplified version of this presentation.

For the best experience please use the latest **Chrome** or **Safari** browser. Firefox 10 (to be released soon) will also handle it.

pypi: uiautomator

Jul 2013 · He, Xiaocong

什么是Android UiAutomator ?

Android UiAutomator : http://developer.android.com/tools/help/uiautomator/index.

优点

- 能根据文本，描述等属性定位需要操作的UI元素
- 支持多种UI操作，包括点击，按键，托拽，手势等
- 支持自定义Watcher来处理测试过程中的意外，如呼入电话，ANR对话框
- 可以对整个设备进行跨应用（进程）的测试
- 使用强大的Java语言，对Android开发者友好

缺点

- 只支持API level 16+ (Android 4.1+)
- 编译，运行的步骤复杂（Java的特点…）

- 使用笨重的Java语言…
- 这不是单元测试的工具

pypi: uiautomator

https://github.com/xiaocong/uiautomator

安装

```
$ [sudo] pip install uiautomator
```

初步使用uiautomator

```
from uiautomator import device as d

d.screen.on()
d(resourceId="android:id/glow_pad_view").swipe.down()
d(text="Camera").click()
d.press.back()
d.press.power()
```

在原生Android 4.3 锁屏界面运行如上代码得到如下结果：

- 点亮屏幕
- 向下滑动解锁图标
- 点击文本”Camera”所在的区域
- 按”回退(back)”键

- 按"回退(back)"键
- 按"电源(power)"键，等价于灭屏操作

设备操作

- 获取设备信息
- 点亮/熄灭屏幕
- 坐标点击
- 屏幕托拽操作
- 按键操作
- 获得/设置设备的Orientation
- 截取屏幕图片
- Dump界面的层次结构
- 打开通知和快速设置界面
- 设置Watcher

设备操作 - 获取设备信息

```
from uiautomator import device as d

d.info
```

获得如下信息：

```
{ u'displayRotation': 0,
  u'displaySizeDpY': 640,
  u'displaySizeDpX': 360,
  u'currentPackageName': u'com.android.launcher',
  u'productName': u'takju',
  u'displayWidth': 720,
  u'sdkInt': 18,
  u'displayHeight': 1184,
  u'naturalOrientation': True
}
```

设备操作 - 屏幕

```
from uiautomator import device as d

# 点亮屏幕
d.screen.on()
# 关闭屏幕
d.screen.off()
```

类似于：

```
d.wakeup()   # screen on
d.sleep()  # screen off
```

设备操作 - 坐标点击

```
from uiautomator import device as d

# 点击屏幕坐标点(x, y)
d.click(x, y)
```

设备操作 - 托拽

```
from uiautomator import device as d

# 从屏幕坐标点(sx, sy)托拽到(ex, ey)
d.drag(sx, sy, ex, ey)
```

设备操作 - 按软/硬键

```
# 按 Home 键
d.press.home()
# 下面与上面等价
d.press("home")
```

下面是所有按键的定义：

- home, back, recent, menu, search
- left, right, top, bottom, center
- enter, delete(del), power, camera
- volume_up, volume_down, volume_mute

设备操作 - Orientation

```
# 获得 orientation
d.orientation

# 设置 Orientation
d.orientation = "left"  # or "right", "natural"

# 冻结/解除冻结屏幕rotation
d.freeze_rotation()
d.freeze_rotation(False)
```

支持的Orientation包括：

- right or r
- left or l
- natural or n
- upsidedown or u(不能被设置)

设备操作 - 截屏和dump

截屏

```
# 截屏并保存为本地 home.png
d.screenshot("home.png")
```

Dump

```
# Dump hierarchy 到本地文件 home.xml
d.dump("home.xml")
```

设备操作 - 打开通知和快速设置

```
# 打开通知
d.open.notification()

# 打开快速设置
d.open.quick_settings()
```

设备操作 - Watcher

```
# 注册watcher
d.watcher("CLOSE-ANR-DIALOG") \
 .when(text="ANR") \
 .click(className="android.widget.Button", text="Force Close")

# 注册watcher,  当满足条件是按back和home键
d.watcher("CLOSE-ANR-DIALOG") \
 .when(text="ANR") \
 .press.back.home()
```

UI对象操作

- 获取UI对象信息
- 点击或者长点击
- 文本编辑
- 对象托拽
- 滑动
- 滚动和快速滚动
- 手势
- 等待直到出现或者消失

UI对象 - 选择器

```
# 选择text为 Settings的UI对象
d(text="Settings")
# 选择标准Button UI对象
d(className="android.widget.Button")
# UI对象属性
d(className="android.widget.Button").info
```

常用的选择器参数

- 文本: text
- 描述: description
- 类型: className
- 选择: checked
- 可滚动: scrollable
- .....

UI对象 - 点击

```
from uiautomator import device as d

# 点击text为 Settings的UI对象
d(text="Settings").click()
# 长点击text为 Settings的UI对象
d(text="Settings").long_click()
```

UI对象 - 可编辑文本

```
from uiautomator import device as d

# 获得text属性
d(className="android.widget.EditText").text
# 设置可编辑文本
d(text="Email").set_text("xiaocong@gmail.com")
# 清除可编辑文本
d(text="xiaocong@gmail.com").clear_text()
```

UI对象 - 推拽

```
from uiautomator import device as d

# 推拽一个UI对象到另外一个UI对象
d(text="Clock").drag.to(text="Calculator")
# 推拽一个UI对象到坐标(x, y)
d(text="Clock").drag.to(x, y)
```

UI对象 - Swipe

```
# 向右滑动
d(resourceId="com.android.launcher:id/workspace").swipe.right()
# 或者这样写
d(resourceId="com.android.launcher:id/workspace").swipe("left")
```

支持4个方向的滑动：left, right, up, down

UI对象 - Scroll

```
# 水平向前滚动
d(scrollable=True).scroll.horiz.forward()
# 垂直滚动到开始
d(scrollable=True).scroll.vert.toBeginning()
# 水平滚动，直到目标UI对象出现
d(scrollable=True).scroll.horiz.to(text="Clock")
```

支持2个维度的滚动：horiz和vert

支持5种的滚动目标：forward, backward, toBeginning, toEnd, to

UI对象 - Fling

```
# 水平向前快速滚动
d(scrollable=True).fling.horiz.forward()
# 垂直快速滚动到开始
d(scrollable=True).fling.vert.toBeginning()
```

支持2个维度的滚动：horiz和vert

支持4种的滚动目标：forward, backward, toBeginning, toEnd

UI对象 - 手势

```
from uiautomator import device as d, point

# 双指挤入
d(resourceId="com.android.gallery3d:id/gl_root_view") \
 .pinch.In()
# 双指挤开
d(resourceId="com.android.gallery3d:id/gl_root_view") \
 .pinch.In()
# 双点手势
d(resourceId="com.android.gallery3d:id/gl_root_view") \
```

```
.gesture(point(sx1, sy1), point(sx2, sy2)) \
.to(point(ex1, ey1), point(ex2, ey2))
```

UI对象 - 存在，等待出现或者消失

```
# 存在与否
d(text="Clock").exists
# 或者
d.exists(text="Clock")
# 等待直到出现
d(text="Clock").wait.exists(timeout=5000)
# 等待直到消失
d(text="Clock").wait.gone(timeout=5000)
```

可以集成的Python测试框架

- Python unittest: http://docs.python.org/2/library/unittest.html
- Nose: https://github.com/nose-devs/nose
- Lettuce: http://lettuce.it/

参考

- Android uiautomator: http://developer.android.com/tools/help/uiautomator/index.html
- Github Repo: https://github.com/xiaocong/uiautomator

谢谢！