

app 自动化测试



目录

contents

1 环境安装

4 pytest

2 adb调试及快速体验

5 PO模式

3 UIAutomatorViewer

6 参数化和allure报告

1 环境安装

- 1.1 java环境安装
- 1.2 python环境安装
- 1.3 Android SDK安装
- 1.4 安装模拟器或者真机
- 1.5 安装node.js环境
- 1.6 appium环境安装
- 1.7 安装Appium-Python-Client库
- 1.8 python的其它库
- 1.9 allure, xml转html工具

1.1 java环境安装（推荐安装jdk1.8）

配置环境变量：

自定义变量JAVA_HOME=C:\Java\jdk1.8.0_151

注意：C:\Java\jdk1.8.0_151 为jdk的安装目录

配置path：

%JAVA_HOME%\bin

%JAVA_HOME%\jre\bin

验证： 终端输入： java -version

```
C:\Users\Administrator>java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```


1.2 python环境安装

安装

官网下载对应的安装包，安装到指定目录下
配置path:

C:\Python36\

验证： 终端输入： python

A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe - python". The window content shows the following text: "Microsoft Windows [版本 10.0.17763.805]", "(c) 2018 Microsoft Corporation。保留所有权利。", "C:\Users\Administrator>python", "Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32", "Type "help", "copyright", "credits" or "license" for more information.", and ">>> _". A red rectangle highlights the "python" command in the prompt line.

```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 10.0.17763.805]
(c) 2018 Microsoft Corporation。保留所有权利。
C:\Users\Administrator>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

1.3 Android SDK 安装

解压android-sdk.zip到指定目录

配置环境变量：

自定义变量 ANDROID_HOME=C:\android-sdk

C:\android-sdk是你的解压目录

platform-tools文件夹和tools文件夹都在C:\android-sdk下

配置path：

%ANDROID_HOME%\platform-tools

%ANDROID_HOME%\tools

验证： 终端输入： adb --version



```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.17763.805]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>adb --version
Android Debug Bridge version 1.0.39
Version 0.0.1-4500957
Installed as F:\Android_SDK\android-sdk\platform-tools\adb.exe

C:\Users\Administrator>
```

1.4 真机

1. 准备一根USB线
2. 打开手机开发者模式，打开USB调试（不同手机打开方式不一样，可百度查询）

验证： 终端输入：adb devices



```
CA: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.17763.805]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>adb devices
List of devices attached
e0c0e0c5    device

C:\Users\Administrator>
```

1.5 node.js安装

官网下载: node-v8.11.3-x64.msi

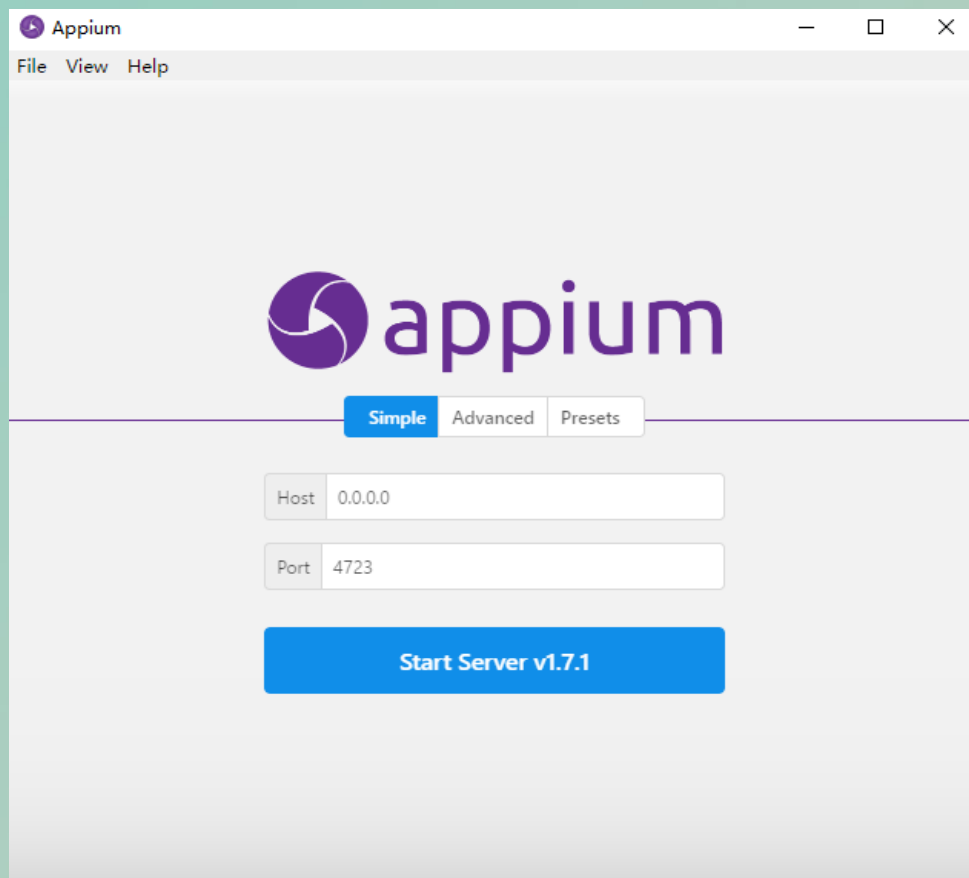
按照提醒一步一步安装

检验: `node -v`

1.6 appium环境安装

安装客户端 -启动Appium服务

直接双击安装appium-desktop-Setup-1.2.7.exe即可



1.7 安装Appium-Python-Client库

命令行安装:

```
pip install Appium-Python-Client==0.40
```

1.8 python其他的库

- pytest, 推荐安装3.6.1版本

```
pip install pytest==3.6.1
```

- pytest-html, 推荐安装1.19.0版本

```
pip install pytest-html==1.19.0
```

- pytest-ordering, 推荐安装0.5版本

```
pip install pytest-ordering==0.5
```

pytest-rerunfailures, 推荐安装4.1版本

```
pip install pytest-rerunfailures
```

PyYAML库安装, 推荐安装3.11版本

```
pip install PyYAML==4.2b4
```

pytest-allure-adaptor, 推荐安装1.7.10

```
pip install pytest-allure-adaptor==1.7.10
```

1.9 allure, xml转html工具

1. 解压压缩包allure-2.5.0.zip
2. 将压缩包内的bin目录配置到path系统环境变量

D:\app_tools\allure-2.6.0\bin

验证 allure --version



```
CA: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.17763.805]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>allure --version
2.6.0
C:\Users\Administrator>
```

2 adb调试及快速体验

2.1 概念

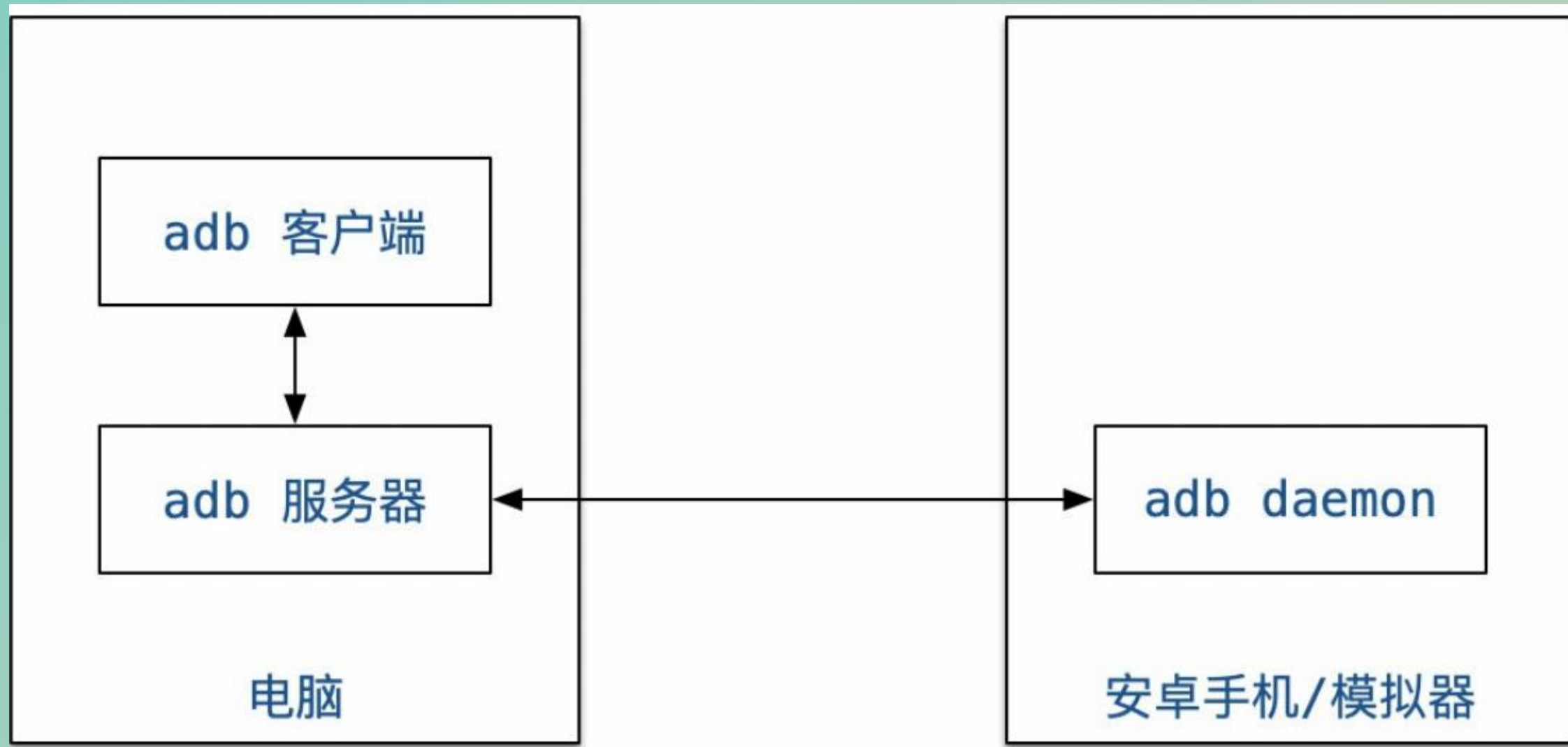
ADB 全名 Android Debug Bridge，是一个调试工具

2.2 构成及工作原理

Client端：运行在开发机器中，即你的开发电脑，用来发送 adb 命令

Daemon守护进程：运行在调试设备中，手机或模拟器，用来接收并执行 adb命令

Server端：同样运行在开发机器中，用来管理 Client 端和手机的 Daemon 之间的通信



2.3 adb常用命令

获取包名和界面名（windows）：

```
adb shell dumpsys window windows | findstr mFocusedApp
```

文件传输：

```
adb push 电脑的文件路径 手机的文件夹路径
```

```
adb pull 手机的文件路径 电脑的文件夹路径
```

获取手机日志：

```
adb logcat
```

安装/卸载app：

```
adb install 路径/xx.apk          adb uninstall 包名
```

获取当前电脑连接的设备:adb devices

进入到安卓手机内部的linux系统命令行:adb shell

启动adb服务器:adb start-server

停止adb服务器:adb kill-server

2.4 快速体验app自动化（test1）

```
import time
from appium import webdriver
desired_caps = dict()
desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '9'
desired_caps['deviceName'] = '192.168.56.101:5555'
desired_caps["noReset"] = 'True'
desired_caps['appPackage'] = 'com.xiaomi.xiaoilite'
desired_caps['appActivity'] = '.presenter.info.PersonalInfoActivity'
driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)
driver.implicitly_wait(3)
driver.find_element_by_id("com.xiaomi.xiaoilite:id/tv_user_name").click()
driver.find_element_by_id("username").send_keys("ahh")
driver.find_element_by_id("pwd").send_keys("jds")
driver.find_element_by_id("login-button").click()
time.sleep(6)
driver.quit()
```

3 UIAutomatorViewer

3.1 概念

UIAutomatorViewer用来扫描和分析Android应用程序的UI控件的工具。

3.2 使用步骤

1. 进入SDK目录下的目录

windows在tools目录下，打开uiautomatorviewer.bat

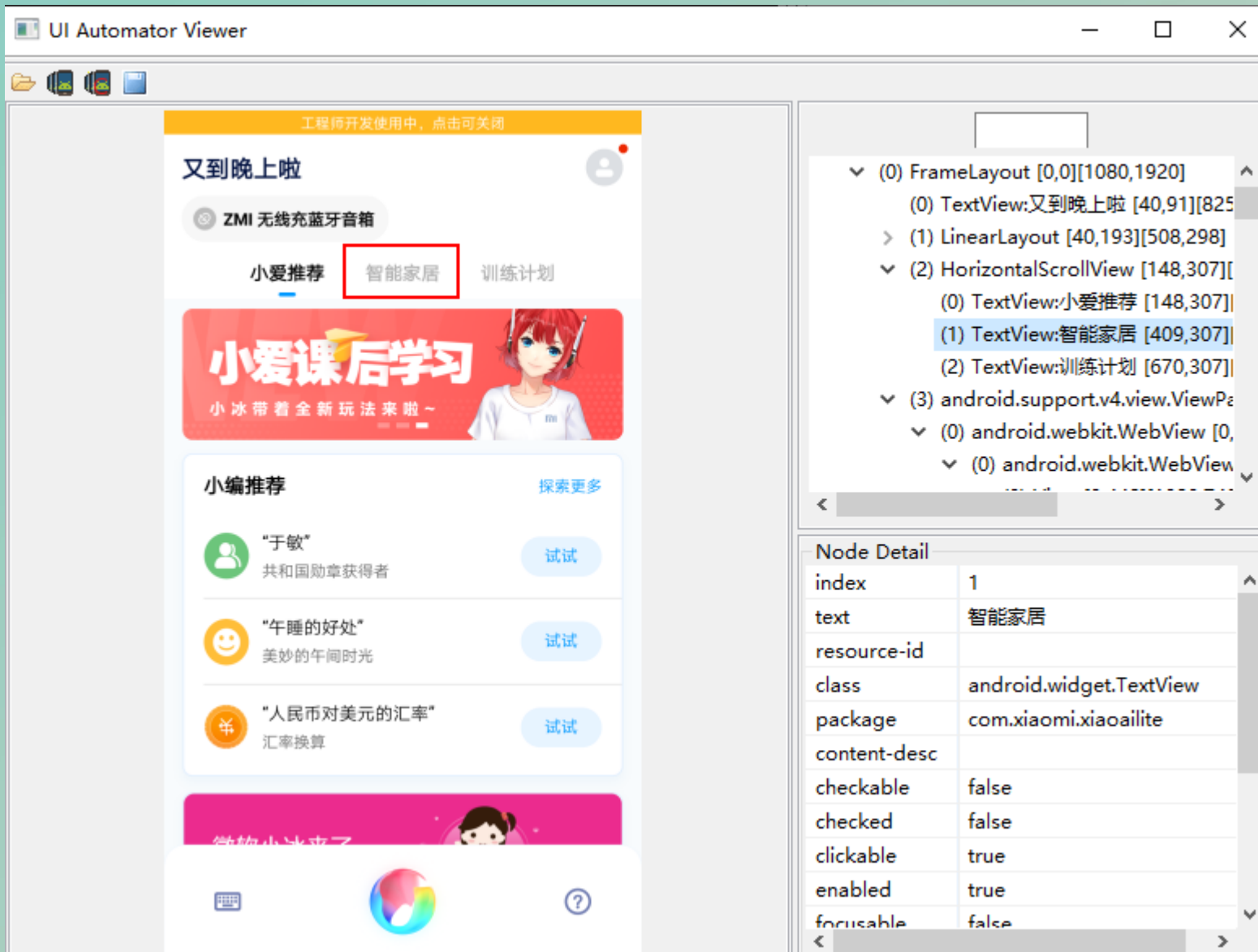
2. 电脑连接真机或打开android模拟器

3. 启动待测试app

4. 点击 uiautomatorviewer的左上角Device Screenshot（从左数第二个按钮）

5. 点击希望查看的控件

6. 查看右下角 Node Detail 相关信息



3.3 元素定位

定位一个元素（返回单个元素）：

id: `driver.find_element_by_id(id_value)`

class: `driver.find_element_by_class_name(class_value)`

xpath: `driver.find_element_by_xpath(xpath_value)`

定位一组元素（返回一个列表）：

id: `driver.find_elements_by_id(id_value)`

class: `driver.find_elements_by_class_name(class_value)`

xpath: `driver.find_elements_by_xpath(xpath_value)`

3.4 对元素进行操作

点击: `click()`

输入: `send_keys()`

清除: `clear()`

3.5 元素等待(隐式全局, 显示单个元素)

隐式等待（针对所有定位元素的超时时间设置为同一个值的时候）：

等待元素加载指定的时长，超出时长抛出NoSuchElementException异常

语法：driver.implicitly_wait(5)

显示等待（针对所有定位元素的超时时间设置为不同的值的时候）：

等待元素加载指定的时长，超出时长抛出TimeoutException异常

创建WebDriverWait对象

```
wait = WebDriverWait(driver, timeout, poll_frequency=1)
```

获取元素并设置超时时间和频率

```
wait.until(lambda x: x.find_element_by_xpath("")).click()
```

参数：

driver: 驱动对象

timeout: 超时的时长，单位：秒

poll_frequency: 检测间隔时间，默认为0.5秒

返回值：

WebDriverWait对象

```
WebDriverWait(driver, timeout, poll_frequency=0.5)
```

3.6 滑动和拖拽事件

swipe滑动事件（test1）：

从一个坐标位置滑动到另一个坐标位置，只能是两个点之间的滑动。

```
# 从一个坐标位置滑动到另一个坐标位置，只能是两个点之间的滑动
# 参数：
#     start_x:    起点X轴坐标
#     start_y:    起点Y轴坐标
#     end_x:      终点X轴坐标
#     end_y:      终点Y轴坐标
#     duration:   滑动这个操作一共持续的时间长度，单位：ms
driver.swipe(start_x, start_y, end_x, end_y, duration=None)
```

距离相同时，持续时间越长，惯性越小

持续时间相同时，手指滑动的距离越大，实际滑动的距离也就越大

3.6 滑动和拖拽事件

drag_and_drop 拖拽事件（test1）：

从一个元素滑动到另一个元素，第二个元素替代第一个元素原本屏幕上的位置。

```
# 从一个元素滑动到另一个元素，第二个元素替代第一个元素原本屏幕上的位置
# 参数：
#     origin_el:      滑动开始的元素
#     destination_el: 滑动结束的元素
driver.drag_and_drop(origin_el, destination_el)
```

不能设置事件，没有惯性。

3.6 手机操作

获取手机分辨率

语法: `driver.get_window_size()`

手机截图

语法: `# 参数:`
`# filename: 指定路径下, 指定格式的图片`
`get_screenshot_as_file(filename)`

`driver.get_screenshot_as_file(os.getcwd() + os.sep + './screen.png')`

获取和设置手机网络:

获取: `driver.network_connection`

Possible values:

<i>Value (Alias)</i>	<i> Data</i>	<i> Wifi</i>	<i> Airplane Mode</i>
<i>-----</i>			
<i>0 (None)</i>	<i> 0</i>	<i> 0</i>	<i> 0</i>
<i>1 (Airplane Mode)</i>	<i> 0</i>	<i> 0</i>	<i> 1</i>
<i>2 (Wifi only)</i>	<i> 0</i>	<i> 1</i>	<i> 0</i>
<i>4 (Data only)</i>	<i> 1</i>	<i> 0</i>	<i> 0</i>
<i>6 (All network on)</i>	<i> 1</i>	<i> 1</i>	<i> 0</i>

3.6 手机操作

设置手机网络

```
# 设置手机网络
# 参数:
#     connectionType: 网络类型
driver.set_network_connection(connectionType)
```

设置飞行模式: `driver.set_network_connection(1)`

发送键到设备（可百度查对应的键）

```
# 发送键到设备
# 参数:
#     keycode: 发送给设备的关键代码
#     metastate: 关于被发送的关键代码的元信息，一般为默认值
driver.press_keycode(keycode, metastate=None)
```

3.6 手机操作

操作手机通知栏

```
# 打开手机通知栏  
driver.open_notifications()
```

注：

appium官方并没有为我们提供关闭通知的api，那么现实生活中怎么关闭，就怎样操作就行，比如，手指从下往上滑动，或者，按返回键

4 pytest (test2)

4.1 概念:

pytest是python的一种单元测试框架，同自带的Unittest测试框架类似，相比于Unittest框架使用起来更简洁，效率更高。

安装: `pip3 install pytest`

4.2 安装校验:

1. 进入命令行
2. 输入命令 `pytest --version` 会展示当前已安装版本

4.3 执行: `pytest -s test_login.py`

. 表示成功

F 表示失败

4.4 setup 和 teardown

pytest在运行自动化脚本的前后会执行两个特殊的方法，分别是setup和teardown。在执行脚本之前会执行setup方法，在执行脚本之后会执行teardown方法。有了这两个方法，我们可以在setup中进行获取驱动对象的操作，在teardown中进行关闭驱动对象的操作。

函数级别方法：

运行于测试方法的始末，运行一次测试函数会运行一次setup和teardown。

类级别方法：

运行于测试方法的始末，运行一次测试函数会运行一次 setup 和 teardown。

```
import pytest
class TestLogin:
    # 函数级开始
    def setup(self):
        print("————>setup_method")
    # 函数级结束
    def teardown(self):
        print("————>teardown_method")
    def test_a(self):
        print("————>test_a")
    def test_b(self):
        print("————>test_b")
```

```
class TestLogin:
    # 测试类级开始
    def setup_class(self):
        print("————>setup_class")
    # 测试类级结束
    def teardown_class(self):
        print("————>teardown_class")
    def test_a(self):
        print("————>test_a")
    def test_b(self):
        print("————>test_b")
```

4.4 配置文件（test3）

使用配置文件后可以快速的使用配置的项来选择执行哪些测试模块。

使用方法：

1. 项目下新建 scripts 模块
2. 将测试脚本文件放到 scripts 中
3. pytest 的配置文件放在自动化项目目录下
4. 名称为 pytest.ini
5. 命令行运行时会使用该配置文件中的配置
6. 第一行内容为 [pytest]

文件内容：

有图

```
[pytest]
# 添加命令行参数
addopts = -s
# 文件搜索路径
testpaths = ./scripts
# 文件名称
python_files = test_*.py
# 类名称
python_classes = Test*
# 方法名称
python_functions = test_*
```

4.4 测试报告 (test3)

自动化测试脚本最终执行是通过还是不通过，需要通过测试报告进行体现。

安装：pip3 install pytest-html 进行安装

使用：在配置文件中的命令行参数中增加 --html=用户路径/report.html

```
[pytest]
addopts = -s --html=report/report.html --reruns 0
testpaths = ./scripts
python_files = test_*.py
python_classes = Test*
python_functions = test_*
```

4.5 控制函数执行顺序 (test4)

安装：pip3 install pytest-ordering

使用：

1. 标记于被测试函数，@pytest.mark.run(order=x)
2. 根据order传入的参数来解决运行顺序
3. order值全为正数或全为负数时，运行顺序：值越小，优先级越高
4. 正数和负数同时存在：正数优先级高

4.6 失败重试 (test5)

测试用例在运行过程中可能出现网络不稳定而导致失败

安装: `pip3 install pytest-rerunfailures`

使用: 在配置文件中的命令行参数中增加 `--reruns n`

```
[pytest]
addopts = -s --html=report/report.html --reruns 3
testpaths = ./script
python_files = test_*.py
python_classes = Test*
python_functions = test_*
```

4.7 跳过测试函数 (test6)

```
# 跳过测试函数
# 参数:
#     condition: 跳过的条件, 必传参数
#     reason: 标注原因, 必传参数
@pytest.mark.skipif(condition, reason=None)
```

4.8 参数化 (test6)

```
# 数据参数化
# 参数:
#     argnames: 参数名
#     argvalues: 参数对应值, 类型必须为可迭代类型, 一般使用list
@pytest.mark.parametrize(argnames, argvalues, indirect=False, ids=None, scope=None)
```

一个参数的使用方式

1. `argnames` 为字符串类型, 根据需求决定何时的参数名
2. `argvalues` 为列表类型, 根据需求决定列表元素中的内容
3. 在测试脚本中, 参数, 名字与 `argnames` 保持一致
4. 在测试脚本中正常使用

`argvalues` 列表有多少个内容, 这个脚本就会运行几次

多个参数的使用方式 (见test6)

4.8 toast (test7)

安装环境:

1. 安装node.js (使用 npm 或 node 验证)

node-v8.11.3-x64.msi(windows)

2. 安装cnpm (使用cnpm验证)

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

3. 下载 appium-uiautomator2-driver

cnpm install appium-uiautomator2-driver

4. 使用 npm install 或者 cnpm install 安装完成后, 都会提示Installed xx packages 或者 Allpackages installed 只要看到这种, 就说明成功了。如果不成功则请确保按照注意点做后, 再次使用相同的命令重试。

4.8 toast (test7)

使用:

1. 前置代码添加: `desired_caps['automationName'] = 'UiAutomator2'`
2. 使用xpath找text即可

```
@pytest.mark.parametrize(("message"), ["输入为空"])
def test_a(self, message): # test开头的测试函数
    desired_caps = dict()
    desired_caps['platformName'] = 'Android'
    desired_caps['platformVersion'] = '6'
    desired_caps['deviceName'] = '192.168.56.101:5555'
    desired_caps["noReset"] = 'True'
    # toast
    desired_caps['automationName'] = 'UiAutomator2'
    desired_caps['appPackage'] = 'com.xiaomi.xiaoilite'
    desired_caps['appActivity'] = '.presenter.main.MainTabActivity'
    driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)
    driver.implicitly_wait(3)
    driver.find_element_by_id("com.xiaomi.xiaoilite:id/iv_query_keyboard").click()
    driver.find_element_by_id("com.xiaomi.xiaoilite:id/tvSendQuery").click()
    element = WebDriverWait(driver, 5, 0.1).until(lambda x: x.find_element(By.XPATH, "//*[contains(@text, '%s')]") % message)
    print(element.text)

    time.sleep(6)
    driver.quit()
```


5 PO模式

5.1 概念

PO是 Page Object 的缩写，PO模式是自动化测试项目开发实践的最佳设计模式之一



核心思想是通过对界面元素的封装减少冗余代码，同时在后期维护中，若元素定位发生变化，只需要调整页面元素封装的代码，提高测试用例的可维护性、可读性。

5.2 用例准备

- 1.点击首页右下角键盘，不做输入，点击发送，是否有输入为空的toast弹出
- 2.用户登录
- 3.点击个人中心页设备指南按钮，是否正确跳转到设备指南页

做对比：传统方法和PO模式（test8）

6 参数化和allure报告

6.1 参数化

实现测试数据和代码分离，便于管理。

6.2 json介绍 （代码和测试数据分离test9）

JSON的全称是” JavaScript Object Notation”，是JavaScript对象表示法，它是一种基于文本，独立于语言的轻量级数据交换格式。

语法规则：

- 大括号保存对象
- 中括号保存数组
- 对象数组可以相互嵌套
- 数据采用键值对表示
- 多个数据由逗号分隔

6.2 json

json的值

- 数字（整数或浮点数）
- 字符串（在双引号中）
- 逻辑值（**true** 或 **false**）
- 数组（在中括号中）
- 对象（在大括号中）
- **null**

数据的操作：

- 1.python字典与JSON之间的转换
- 2.JSON文件读写

6.2 json

python字典转换json字符串 (json1)

```
data = {  
    'id': 1,  
    'name': 'Tom',  
    'address': '北京市海淀区',  
    'school': None  
}  
  
json_str = json.dumps(data)  
  
print(json_str)
```

json格式转换成python字典

```
json_str = '{"id": 1, "name": "Tom", "address": "北京市海淀区", "school": null}'  
dict_data = json.loads(json_str)  
print(dict_data)
```

6.2 json

读取JSON文件（json2）

```
with open(config.path+'\\data\\login.json', encoding='UTF-8') as f:  
    data = json.load(f) # 返回的数据类型为字典或列表  
  
print(data)
```

6.3 allure报告（test10）

简介：

我们自动化的结果一定是通过一个报告来进行体现。**Allure** 是一个独立的报告插件，生成美观易读的报告，目前支持 **Java**、**PHP**、**Ruby**、**Python**、**Scala**、**C#** 这些语言。

步骤概述

最终我们会生成一个 **html** 格式的报告，中间我们需要操作两步来进行。

1. 生成xml

2. 将 xml 转成 html

6.3 allure报告

1.1 生成 xml

安装: `pip3 install pytest-allure-adaptor`

使用步骤:

1. 将 `pytest` 配置文件中的命令行参数加上如下代码

`--alluredir report`

2. 编写好测试脚本后, 正常的在命令行中运行 `pytest` 即可

```
[pytest]
addopts = -s --alluredir report --reruns 1
testpaths = ./scripts
python_files = test_*.py
python_classes = Test*
python_functions = test_*
```

1.2 将 xml 转成 html

6.3 allure报告

1.2 将 xml 转成 html

安装:

1. <https://bintray.com/qameta/generic/allure2> 下载 allure-2.6.0.zip
2. 解压缩到一个目录（不经常动的目录）
3. 将压缩包内的 bin 目录配置到 path 系统环境变量
4. 右键我的电脑 - 属性 - 高级设置 - 环境变量 - 找到系统环境变量的path项 - 增加 **allure**到bin 目录
5. 在命令行中敲 **allure** 命令，如果提示有这个命令，即为成功

使用步骤:

在保证项目中的 **report** 目录下有 **xml** 文件的时候，执行以下步骤。

1. 进入 **report** 上级目录执行命令

```
allure generate report/ -o report/html --clean
```

2. **report** 目录下会生成 **html** 文件夹，**html** 下会有一个 **index.html**，右键用浏览器打开即可。

6.3 allure报告

1.3 参数和命令详解

1. `addopts = -s --alluredir report` 中的 `--alluredir report` 是什么意思?
 - `--alluredir` 后面的 `report` 为 xml 输出的目录名
 - 如果希望目录名叫 `result` 那么可以将命令行参数改为 `--alluredir result`
2. `allure generate report/ -o report/html --clean` 是什么意思?
 - `report/` 表示 xml 所在的目录
 - `-o` 表示 output 输出
 - `report/html` 表示将 `index.html` 报告生成到哪个文件夹

Python自学: <https://www.runoob.com/python3/python3-tutorial.html>

Github网站: 百度搜索github官网

谢

谢

观

看

