## 问题思考

前面我们都是基于线性模型来编写测试脚本，而且元素定位方式和属性值都是写死的。在业务场景简单的情况下这样写无可厚非，但是一旦遇到产品需求变更，业务逻辑比较复杂需要维护的时候就非常麻烦了，那么该如何应对这种情况呢？

## 场景案例

结合前面我们所学，测试考研帮App登录场景，按照线性模型来构造出脚本如下： 考研帮登录测试场景

**kyb\_login.py**

from appium **import** webdriver

**import** yaml

from selenium.common.exceptions **import** NoSuchElementException

**import** logging

**import** logging.config

CON\_LOG='../log/log.conf'

logging.config.fileConfig(CON\_LOG)

logging=logging.getLogger()

stream=open('../yaml/desired\_caps.yaml','r')

**data**=yaml.load(stream)

desired\_caps={}

desired\_caps['platformName']=**data**['platformName']

desired\_caps['platformVersion']=**data**['platformVersion']

desired\_caps['deviceName']=**data**['deviceName']

desired\_caps['app']=**data**['app']

desired\_caps['noReset']=**data**['noReset']

desired\_caps['unicodeKeyboard']=**data**['unicodeKeyboard']

desired\_caps['resetKeyboard']=**data**['resetKeyboard']

desired\_caps['appPackage']=**data**['appPackage']

desired\_caps['appActivity']=**data**['appActivity']

driver = webdriver.Remote('http://'+str(**data**['ip'])+':'+str(**data**['port'])+'/wd/hub', desired\_caps)

def check\_updateBtn():

logging.info("check\_updateBtn")

try:

element = driver.find\_element\_by\_id('android:id/button2')

except NoSuchElementException:

logging.info('update element is not found!')

**else**:

element.click()

def check\_skipBtn():

logging.info("check\_skipBtn")

try:

element = driver.find\_element\_by\_id('com.tal.kaoyan:id/tv\_skip')

except NoSuchElementException:

logging.info('skipBtn element is not found!')

**else**:

element.click()

check\_updateBtn()

check\_skipBtn()

logging.info('start login...')

driver.find\_element\_by\_id('com.tal.kaoyan:id/login\_email\_edittext').send\_keys('自学网2018')

driver.find\_element\_by\_id('com.tal.kaoyan:id/login\_password\_edittext').send\_keys('zxw2018')

driver.find\_element\_by\_id('com.tal.kaoyan:id/login\_login\_btn').click()

logging.info('login finished')

#### 案例分析

上面的脚本看似都比较完善，有了log采集，参数配置、启动时页面元素自动检测。但是也存在一些不足之处：

* 公共模块和业务模块混合在一起显得代码冗余等
* 测试场景单一（如果要实现如下测试场景该怎么办？）
* 元素定位属性和代码混杂在一起

以上这些都是需要优化的地方。

| **测试场景** | **操作步骤** | **预期结果** |
| --- | --- | --- |
| 多账号登录 | 不同的用户名密码来进行登录 | 能够正常登录 |
| 异常登录 | 用户名或者密码错误、或者为空进行登录， | 登录失败，同时界面要给出相应的提示 |
| 注册 | 点击注册，然后进行注册信息填写 | 能够注册成功 |

#### 重构优化思路

* 将一些公共的内容（如：check\_updateBtn，check\_skipBtn,capability）抽离出来。
* 元素定位方法和元素属性值与业务代码分离
* 登录功能模块封装为一个独立的模块
* 使用unittest进行用例综合管理

#### Page Object

Page Object是Selenium自动化测试项目开发实践的最佳设计模式之一，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。

## 脚本实现

封装App启动配置信息 desired\_caps.py

import yaml

import logging.config

from appium import webdriver

CON\_LOG = '../log/log.conf'

logging.config.fileConfig(CON\_LOG)

logging = logging.getLogger()

def appium\_desired():

stream = open('../yaml/desired\_caps.yaml', 'r')

data = yaml.load(stream)

desired\_caps={}

desired\_caps['platformName']=data['platformName']

desired\_caps['platformVersion']=data['platformVersion']

desired\_caps['deviceName']=data['deviceName']

desired\_caps['app']=data['app']

desired\_caps['noReset']=data['noReset']

desired\_caps['unicodeKeyboard']=data['unicodeKeyboard']

desired\_caps['resetKeyboard']=data['resetKeyboard']

desired\_caps['appPackage']=data['appPackage']

desired\_caps['appActivity']=data['appActivity']

logging.info('start run app...')

driver = webdriver.Remote('http://'+str(data['ip'])+':'+str(data['port'])+'/wd/hub', desired\_caps)

driver.implicitly\_wait(8)

return driver

if \_\_name\_\_ == '\_\_main\_\_':

appium\_desired()

记得在原来的yaml配置表desired\_caps.yaml补充如下内容：

unicodeKeyboard: True

resetKeyboard: True

封装基类： [baseView.py](http://baseView.py)

**class** **BaseView**(**object**):

**def** **\_\_init\_\_**(**self**,driver):

**self**.driver=driver

**def** **find\_element**(**self**,\*loc):

**return** **self**.driver.find\_element(\*loc)

补充资料：

[Python类与对象视频教程](http://www.51zxw.net/show.aspx?id=60317&cid=615)

[Selenium PageObject 视频讲解](http://www.51zxw.net/show.aspx?id=61331&cid=615)

[Python可变数量参数视频讲解](http://www.51zxw.net/show.aspx?id=61330&cid=615)

封装通用公共类 common\_fun.py

**from** appium\_advance.page\_object.baseView **import** BaseView

**from** selenium.common.exceptions **import** NoSuchElementException

**import** logging

**from** selenium.webdriver.common.by **import** By

**from** appium\_advance.page\_object.desired\_caps **import** appium\_desired

**class** **Common**(BaseView):

cancelBtn=(By.ID,'android:id/button2')

skipBtn=(By.ID,'com.tal.kaoyan:id/tv\_skip')

**def** **check\_cancelBtn**(self):

logging.info("============check\_cancelBtn===============")

**try**:

element = self.driver.find\_element(\*self.cancelBtn)

**except** NoSuchElementException:

logging.info('update element is not found!')

**else**:

logging.info('click cancelBtn')

element.click()

**def** **check\_skipBtn**(self):

logging.info("==========check\_skipBtn===========")

**try**:

element = self.driver.find\_element(\*self.skipBtn)

**except** NoSuchElementException:

logging.info('skipBtn element is not found!')

**else**:

logging.info('click skipBtn')

element.click()

**if** \_\_name\_\_ == '\_\_main\_\_':

driver=appium\_desired()

com=Common(driver)

com.check\_updateBtn()

com.check\_skipBtn()

补充资料：[By方式元素定位视频讲解](http://www.51zxw.net/show.aspx?id=61329&cid=615)

封装登录操作 [loginView.py](http://loginView.py)

**import** logging

**from** appium\_advance.page\_object.common\_fun **import** Common

**from** appium\_advance.page\_object.desired\_caps **import** appium\_desired

**from** selenium.webdriver.common.by **import** By

**class** **LoginView**(**Common**):

username\_type=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**login\_email\_edittext'**)

password\_type=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**login\_password\_edittext'**)

loginBtn=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**login\_login\_btn'**)

def login\_action(**self**,**username**,**password**):

self.check\_cancelBtn()

self.check\_skipBtn()

logging.info('===============**login**===============')

logging.info('**input** **username**:%**s'**%**username**)

self.driver.find\_element(\***self**.**username\_type**).send\_keys(**username**)

logging.info('**input** **password**:%**s'**%**password**)

self.driver.find\_element(\***self**.**password\_type**).send\_keys(**password**)

logging.info('**click** **loginBtn**.')

self.driver.find\_element(\***self**.**loginBtn**).click()

logging.info('**login** **finished** ')

if \_\_name\_\_ == '\_\_main\_\_':

driver=appium\_desired()

l=**LoginView**(**driver**)

l.login\_action('自学网2018','**zxw2018'**)

## unittest用例封装

#### 测试场景

使用如下账号进行分别登录测试

| **用户名** | **密码** |
| --- | --- |
| 自学网2018 | zxw2018 |
| 自学网2017 | zxw2017 |
| 666 | 222 |

Tips必备基础知识：[Selenium自动化第六章-unittest单元测试框架](http://www.51zxw.net/list.aspx?page=4&cid=615)

1.封装用例启动结束时的配置： [myunit.py](http://myunit.py)

**import** unittest

**from** appium\_advance.page\_object.desired\_caps **import** appium\_desired

**import** logging

**from** time **import** sleep

**class** **StartEnd**(**unittest**.**TestCase**):

def setUp(**self**):

logging.info('======**setUp**=========')

self.driver=appium\_desired()

def tearDown(**self**):

logging.info('======**tearDown**=====')

sleep(5)

self.driver.close\_app()

2.用例封装 **test\_login.py**

**from** appium\_advance.unittest.myunit **import** StartEnd

**from** appium\_advance.page\_object.loginView **import** LoginView

**import** unittest

**import** logging

**class** **TestLogin**(StartEnd):

**def** **test\_login\_zxw2018**(self):

logging.info('=========test\_login\_zxw2018============')

l=LoginView(self.driver)

l.login\_action('自学网2018','zxw2018')

**def** **test\_login\_zxw2017**(self):

logging.info('==========test\_login\_zxw2017========')

l=LoginView(self.driver)

l.login\_action('自学网2017','zxw2017')

**def** **test\_login\_error**(self):

logging.info('=======test\_login\_error=========')

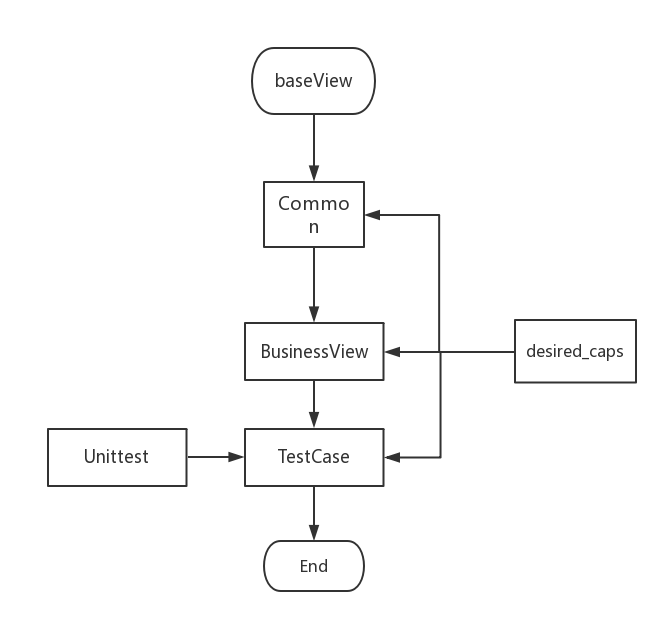
l=LoginView(self.driver)

l.login\_action('666','222')

**if** \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

## 小结



代码写死一时爽，框架重构火葬场。此处功能将来必改，不要写死！

