## 框架背景

前面我们已经学习了Appium各种元素定位，手势操作、数据配置、Pageobject设计模式等等。但是前面的功能都是比较零散的，没有整体融合起来，实际项目实践过程中我们需要综合运用,那么本章节我们将结合之前所学的内容，从0到1搭建一个完整的自动化测试框架。

## 框架功能

*  业务功能的封装
*  测试用例封装
*  测试包管理
*  截图处理
*  断言处理
*  日志获取
*  测试报告生成
*  数据驱动
*  数据配置

## 测试案例

#### 测试环境

* Win10 64Bit
* Appium 1.7.2
* 考研帮App Android版3.1.0
* 夜神模拟器 Android 5.1.1

#### 覆盖用例

1.登录场景

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

2.注册场景

注册一个新的账号(账户和密码可以随机生成)，完善院校和专业信息 （如：院校：上海-同济大学 专业：经济学类-统计学-经济统计学）

## 框架设计图



## 代码实现

#### driver配置封装

kyb\_caps.yaml 配置表

**platformName**: **Android**

#模拟器

**platformVersion**: 5.1.1

**deviceName**: 127.0.0.1:62025

**#mx4**真机

**#platformVersion**: 5.1

**#udid**: 750**BBKL22GDN**

**#deviceName**: **MX4**

**appname**: **kaoyan3**.1.0.apk

**noReset**: **False**

**unicodeKeyboard**: **True**

**resetKeyboard**: **True**

**appPackage**: **com**.tal.kaoyan

**appActivity**: **com**.tal.kaoyan.ui.activity.SplashActivity

**ip**: 127.0.0.1

**port**: 4723

desired\_caps.py

**import** yaml

**import** logging.config

from appium **import** webdriver

**import** os

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

logging.config.fileConfig(CON\_LOG)

logging = logging.getLogger()

def appium\_desired():

**with** open('../config/kyb\_caps.yaml','r',encoding='utf-8') **as** file:

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

desired\_caps={}

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

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

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

base\_dir = os.path.dirname(os.path.dirname(\_\_file\_\_))

app\_path = os.path.join(base\_dir, 'app', **data**['appname'])

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

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(5)

return driver

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

appium\_desired()

# **with** open('../config/kyb\_caps.yaml','r',encoding='utf-8') **as** file:

# **data** = yaml.load(file)

#base\_dir = os.path.dirname(os.path.dirname(\_\_file\_\_))

#app\_path = os.path.join(base\_dir, 'app', **data**['appname'])

#print(app\_path)

###### 相对路径符号含义

1. “.”表示当前目录
2. “..” 表示当前目录的上一级目录。
3. “./”表示当前目录下的某个文件或文件夹，视后面跟着的名字而定
4. “../”表示当前目录上一级目录的文件或文件夹，视后面跟着的名字而定。

#### 基类封装

[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)

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

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

**def** **get\_window\_size**(**self**):

**return** **self**.driver.get\_window\_size()

**def** **swipe**(**self**,start\_x, start\_y, end\_x, end\_y, duration):

**return** **self**.driver.swipe(start\_x, start\_y, end\_x, end\_y, duration)

#### common公共模块封装

公共方法封装 : common\_fun.py

**from** baseView.baseView **import** BaseView

**from** common.desired\_caps **import** appium\_desired

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

**import** logging.config

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

**import** os

**import** time

**import** csv

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

*#取消升级和跳过引导按钮*

cancel\_upgradeBtn=(By.ID,'android:id/button2')

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

*# 登录后浮窗广告取消按钮*

wemedia\_cacel=(By.ID, 'com.tal.kaoyan:id/view\_wemedia\_cacel')

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

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

**try**:

element = self.driver.find\_element(\*self.cancel\_upgradeBtn)

**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()

**def** **get\_screenSize**(self):

'''

获取屏幕尺寸

:return:

'''

x = self.get\_window\_size()['width']

y = self.get\_window\_size()['height']

**return** (x, y)

**def** **swipeLeft**(self):

logging.info('swipeLeft')

l = self.get\_screenSize()

y1 = int(l[1] \* 0.5)

x1 = int(l[0] \* 0.95)

x2 = int(l[0] \* 0.25)

self.swipe(x1, y1, x2, y1, 1000)

**def** **getTime**(self):

self.now = time.strftime("%Y-%m-%d %H\_%M\_%S")

**return** self.now

**def** **getScreenShot**(self, module):

time = self.getTime()

image\_file= os.path.dirname(os.path.dirname(\_\_file\_\_)) + '/screenshots/%s\_%s.png' % (module, time)

logging.info('get %s screenshot' % module)

self.driver.get\_screenshot\_as\_file(image\_file)

**def** **check\_market\_ad**(self):

'''检测登录或者注册之后的界面浮窗广告'''

logging.info('=======check\_market\_ad=============')

**try**:

element=self.driver.find\_element(\*self.wemedia\_cacel)

**except** NoSuchElementException:

**pass**

**else**:

logging.info('close market ad')

element.click()

**def** **get\_csv\_data**(self,csv\_file,line):

'''

获取csv文件指定行的数据

:param csv\_file: csv文件路径

:param line: 数据行数

:return:

'''

**with** open(csv\_file, 'r', encoding='utf-8-sig') **as** file:

reader=csv.reader(file)

**for** index, row **in** enumerate(reader,1):

**if** index == line:

**return** row

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

driver=appium\_desired()

*# c=Common(driver)*

*# c.check\_updateBtn()*

*# # c.check\_skipBtn()*

*# c.swipeLef()*

*# c.swipeLef()*

*# c.getScreenShot("startApp")*

### 业务模块封装

#### 1.登录页面业务逻辑模块

[loginView.py](http://loginView.py)

**import** logging

**from** common.desired\_caps **import** appium\_desired

**from** common.common\_fun **import** Common,By

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

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

#登录界面元素

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'**)

#个人中心元素

username=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**activity\_usercenter\_username'**)

button\_myself=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**mainactivity\_button\_mysefl'**)

# 个人中心下线警告提醒确定按钮

commitBtn = (**By**.**ID**, '**com**.**tal**.**kaoyan**:**id**/**tip\_commit'**)

#退出操作相关元素

settingBtn = (**By**.**ID**, '**com**.**tal**.**kaoyan**:**id**/**myapptitle\_RightButtonWraper'**)

logoutBtn=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**setting\_logout\_text'**)

tip\_commit=(**By**.**ID**,'**com**.**tal**.**kaoyan**:**id**/**tip\_commit'**)

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

self.check\_updateBtn()

self.check\_skipBtn()

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

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

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

logging.info('**password** **is**:%**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**!')

def check\_account\_alert(**self**):

'''检测账户登录后是否有账户下线提示'''

logging.info('====**check\_account\_alert**======')

try:

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

except **NoSuchElementException**:

pass

else:

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

element.click()

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

logging.info('==========**check\_loginStatus**===========')

self.check\_market\_ad()

self.check\_account\_alert()

try:

self.driver.find\_element(\***self**.**button\_myself**).click()

self.driver.find\_element(\***self**.**username**)

except **NoSuchElementException**:

logging.error('**login** **Fail**!')

self.getScreenShot('**login** **Fail**')

return **False**

else:

logging.info('**login** **success**!')

l.logout\_action()

return **True**

def logout\_action(**self**):

logging.info('=========**logout\_action**==========')

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

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

self.driver.find\_element(\***self**.**tip\_commit**).click()

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

driver=appium\_desired()

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

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

l.check\_loginStatus()

#### 注册页面业务逻辑封装

[registerView.py](http://registerView.py)

import logging

from common.desired\_caps import appium\_desired

from common.common\_fun import Common,By,NoSuchElementException

import random

**class** **RegisterView**(**Common**):

*#登录界面注册按钮*

register\_text=(By.ID,'com.tal.kaoyan:id/login\_register\_text')

*#头像设置相关元素*

userheader=(By.ID,'com.tal.kaoyan:id/activity\_register\_userheader')

item\_image=(By.ID,'com.tal.kaoyan:id/item\_image')

saveBtn=(By.ID,'com.tal.kaoyan:id/save')

*# 注册-个人信息界面元素*

register\_username=(By.ID,'com.tal.kaoyan:id/activity\_register\_username\_edittext')

register\_password=(By.ID,'com.tal.kaoyan:id/activity\_register\_password\_edittext')

register\_email=(By.ID,'com.tal.kaoyan:id/activity\_register\_email\_edittext')

register\_btn=(By.ID,'com.tal.kaoyan:id/activity\_register\_register\_btn')

*#完善信息列表元素*

perfectinfomation\_school=(By.ID,'com.tal.kaoyan:id/perfectinfomation\_edit\_school\_name')

perfectinfomation\_major=(By.ID,'com.tal.kaoyan:id/activity\_perfectinfomation\_major')

perfectinfomation\_goBtn=(By.ID,'com.tal.kaoyan:id/activity\_perfectinfomation\_goBtn')

*#院校列表元素*

forum\_title=(By.ID,'com.tal.kaoyan:id/more\_forum\_title')

university=(By.ID,'com.tal.kaoyan:id/university\_search\_item\_name')

*#专业列表元素*

major\_subject\_title= (By.ID, 'com.tal.kaoyan:id/major\_subject\_title')

major\_group\_title= (By.ID, 'com.tal.kaoyan:id/major\_group\_title')

major\_search\_item\_name= (By.ID, 'com.tal.kaoyan:id/major\_search\_item\_name')

*# 个人中心元素*

username = (By.ID, 'com.tal.kaoyan:id/activity\_usercenter\_username')

button\_myself = (By.ID, 'com.tal.kaoyan:id/mainactivity\_button\_mysefl')

**def** **register\_action**(**self**,register\_username,register\_password,register\_email):

**self**.check\_cancelBtn()

**self**.check\_skipBtn()

logging.info('=========register\_action===========')

**self**.driver.find\_element(\***self**.register\_text).click()

*#头像设置*

logging.info('set userheader')

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

**self**.driver.find\_elements(\***self**.item\_image)[10].click()

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

*#用户名密码填写*

logging.info('register username is %s' %register\_username)

**self**.driver.find\_element(\***self**.register\_username).send\_keys(register\_username)

logging.info('register\_password is %s' %register\_password)

**self**.driver.find\_element(\***self**.register\_password).send\_keys(register\_password)

logging.info('register\_email is %s' %register\_email)

**self**.driver.find\_element(\***self**.register\_email).send\_keys(register\_email)

logging.info('click register button')

**self**.driver.find\_element(\***self**.register\_btn).click()

*# 判断是否进入到完善信息界面--注册太频繁会被限制无法进入该界面*

try:

**self**.driver.find\_element(\***self**.perfectinfomation\_school)

except NoSuchElementException:

logging.error('register Fail!')

**self**.getScreenShot('register Fail')

**return** False

else:

**self**.add\_register\_info()

*#注册结果判断*

**if** **self**.check\_registerStatus():

**return** True

else:

**return** False

**def** **add\_register\_info**(**self**):

logging.info('===========add\_register\_info===========')

*# 院校选择:上海——同济大学*

logging.info("select school...")

**self**.driver.find\_element(\***self**.perfectinfomation\_school).click()

**self**.driver.find\_elements(\***self**.forum\_title)[1].click()

**self**.driver.find\_elements(\***self**.university)[1].click()

*#专业选择：经济学类-统计学-经济统计学*

logging.info("select major...")

**self**.driver.find\_element(\***self**.perfectinfomation\_major).click()

**self**.driver.find\_elements(\***self**.major\_subject\_title)[1].click()

**self**.driver.find\_elements(\***self**.major\_group\_title)[2].click()

**self**.driver.find\_elements(\***self**.major\_search\_item\_name)[1].click()

**self**.driver.find\_element(\***self**.perfectinfomation\_goBtn).click()

**def** **check\_register\_status**(**self**):

**self**.check\_market\_ad()

logging.info('==========check\_registerStatus===========')

try:

**self**.driver.find\_element(\***self**.button\_myself).click()

**self**.driver.find\_element(\***self**.username)

except NoSuchElementException:

logging.error('register Fail!')

**self**.getScreenShot('register\_Fail')

**return** False

else:

logging.info('register success!')

**self**.getScreenShot('register\_success')

**return** True

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

driver=appium\_desired()

register=RegisterView(driver)

username='zxw2018'+'FLY'+str(random.randint(1000,9000))

password='zxw'+str(random.randint(1000,9000))

email='51zxw'+str(random.randint(1000,9000))+'@163.com'

register.register\_action(username,password,email)

#### data数据封装

##### 使用背景

在实际项目过程中，我们的数据可能是存储在一个数据文件中，如txt,excel、csv文件类型。我们可以封装一些方法来读取文件中的数据来实现数据驱动。

##### 案例

将测试账号存储在account.csv文件，内容如下：

| **自学网2017** | **zxw2017** |
| --- | --- |
| 自学网2018 | zxw2018 |
| 666 | 222 |

##### enumerate()简介

enumerate()是python的内置函数

* enumerate在字典上是枚举、列举的意思
* 对于一个可迭代的（iterable）/可遍历的对象（如列表、字符串），enumerate将其组成一个索引序列，利用它可以同时获得索引和值
* enumerate多用于在for循环中得到计数。

##### enumerate()使用

如果对一个列表，既要遍历索引又要遍历元素时，首先可以这样写：

**list** = ["这", "是", "一个", "测试","数据"]

**for** i **in** **range**(len(**list**)):

**print**(i,**list**[i])

>>>

0 这

1 是

2 一个

3 测试

4 数据

上述方法有些累赘，利用enumerate()会更加直接和优美：

list1 = ["这", "是", "一个", "测试","数据"]

**for** index, item **in** enumerate(list1):

print(index,item)

>>>

0 这

1 是

2 一个

3 测试

4 数据

##### 数据读取方法封装

import csv

def get\_csv\_data(csv\_file,line):

with **open**(csv\_file, 'r', encoding='utf-8-sig') **as** **file**:

reader=csv.reader(**file**)

**for** index, row in enumerate(reader,1):

**if** index == line:

**return** row

csv\_file='../data/account.csv'

data=get\_csv\_data(csv\_file,3)

**print**(data)

###### utf-8与utf-8-sig两种编码格式的区别

UTF-8以字节为编码单元，它的字节顺序在所有系统中都是一样的，没有字节序的问题，也因此它实际上并不需要BOM(“ByteOrder Mark”)。但是UTF-8 with BOM即utf-8-sig需要提供BOM。

#### config文件配置

日志文件配置 log.config

**[loggers]**

keys=root,infoLogger

**[logger\_root]**

level=DEBUG

handlers=consoleHandler,fileHandler

**[logger\_infoLogger]**

handlers=consoleHandler,fileHandler

qualname=infoLogger

propagate=0

**[handlers]**

keys=consoleHandler,fileHandler

**[handler\_consoleHandler]**

class=StreamHandler

level=INFO

formatter=form02

args=(sys.stderr,)

**[handler\_fileHandler]**

class=FileHandler

level=INFO

formatter=form01

args=('../logs/runlog.log', 'a')

**[formatters]**

keys=form01,form02

**[formatter\_form01]**

format=%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s

**[formatter\_form02]**

format=%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s

#### 测试用例封装

1.测试用例执行开始结束操作封装 [myunit.py](http://myunit.py)

**import** unittest

**from** common.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.注册用例：test\_register.py

**from** common.myunit **import** StartEnd

**from** businessView.registerView **import** RegisterView

**import** logging

**import** random

**import** unittest

**class** **RegisterTest**(**StartEnd**):

def test\_user\_register(**self**):

logging.info('=========**test\_user\_register**======')

r=**RegisterView**(**self**.**driver**)

username = 'zxw2018' + '**FLY'** + str(**random**.**randint**(1000, 9000))

password = 'zxw' + str(**random**.**randint**(1000, 9000))

email = '51zxw' + str(**random**.**randint**(1000, 9000)) + '@163.com'

self.assertTrue(**r**.**register\_action**(**username**, **password**, **email**))

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

unittest.main()

3.登录用例：test\_login.py

from common.myunit import StartEnd

from businessView.loginView import LoginView

import unittest

import logging

**class** **LoginTest**(**StartEnd**):

csv\_file = '../data/account.csv'

*# @unittest.skip("test\_login\_zxw2017")*

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

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

l=LoginView(**self**.driver)

data = l.get\_csv\_data(**self**.csv\_file,1)

l.login\_action(data[0],data[1])

**self**.assertTrue(l.check\_loginStatus())

*# @unittest.skip('skip test\_login\_zxw2018')*

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

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

l=LoginView(**self**.driver)

data = l.get\_csv\_data(**self**.csv\_file,2)

l.login\_action(data[0],data[1])

**self**.assertTrue(l.check\_loginStatus())

*# @unittest.skip("test\_login\_erro")*

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

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

l=LoginView(**self**.driver)

data = l.get\_csv\_data(**self**.csv\_file, 3)

l.login\_action(data[0], data[1])

**self**.assertTrue(l.check\_loginStatus(),msg='login fail!')

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

unittest.main()

#### 执行测试用例&报告生成

[BSTestRunner下载地址](https://github.com/easonhan007/HTMLTestRunner)

[run.py](http://run.py)

**import** unittest

from BSTestRunner **import** BSTestRunner

**import** time

**import** logging

#指定测试用例和测试报告的路径

test\_dir = '../test\_case'

report\_dir = '../reports'

#加载测试用例

discover = unittest.defaultTestLoader.discover(test\_dir, pattern='test\_login.py')

#定义报告的文件格式

now = time.strftime("%Y-%m-%d %H\_%M\_%S")

report\_name = report\_dir + '/' + now + ' test\_report.html'

#运行用例并生成测试报告

**with** open(report\_name, 'wb') **as** f:

runner = BSTestRunner(stream=f, title="Kyb Test Report", description="kyb Andriod app Test Report")

logging.info("start run testcase...")

runner.run(discover)

注意：

pattern参数可以控制运行不同模块的用例，如下所示表示运行指定路径以test开头的模块

discover = unittest.defaultTestLoader.discover(test\_dir, pattern='test\*.py')

### Bat批处理执行测试

前面脚本开发阶段我们都是使用pycharm IDE工具来运行脚本，但是当我们的脚本开发完成后，还每次打开IDE来执行自动化测试就不合理了，因为不仅每次打开比较麻烦，而且pycharm内存资源占用比较“感人”！这样非常影响执行效率。 针对这种情况，我们可以使用cmd命令或者封装为bat批处理脚本来运行。

###### 启动appium服务

start\_appium.bat

@echo off

appium

pause

@echo off 为关闭“回显”，让命令行界面显得整洁一些。

###### 执行测试用例

run.bat

**@echo** off

d:

cd D:\kyb\_testProject\test\_run

C:\Python35\python.exe run.py

pause

##### 注意事项：

1.执行之前需要在run.py脚本添加如下内容：

**import** sys

path='D:\\kyb\_testProject\\'

sys.path.append(path)

项目在IDE（Pycharm）中运行和我们在cmd中运行的路径是不一样的，在pycharm中运行时， 会默认pycharm的目录+我们的工程所在目录为运行目录。

而在cmd中运行时，会以我们的工程目录所在目录来运行。在import包时会首先从pythonPATH的环境变量中来查看包，如果没有你的PYTHONPATH中所包含的目录没有工程目录的根目录，那么你在导入不是同一个目录下的其他工程中的包时会出现import错误。

2.以上脚本编码格式必须为utf-8

## 自动化测试平台

前面我们已经开发完测试脚本，也使用bat批处理来封装了启动Appium服务和运行测试用例。但是还是不够自动化，比如我想每天下班时自动跑一下用例，或者当研发打了新包后自动开始运行测试脚本测试新包，那么该如实现呢？

#### 持续集成（Continuous integration）

持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通过每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽早地发现集成错误。

#### Jenkins简介

Jenkins是一个开源软件项目，是基于Java开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。

#### 下载与安装

下载地址：<https://jenkins.io/download/>

下载后安装到指定的路径即可，默认启动页面为localhots:8080,如果8080端口被占用无法打开，可以进入到jenkins安装目录，找到jenkins.xml配置文件打开，修改如下代码的端口号即可。

<arguments>-Xrs -Xmx256m -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar "%BASE%\jenkins.war" --httpPort=8080 --webroot="%BASE%\war"</arguments>

#### 构建触发器

1. 触发远程构建：如果您想通过访问一个特殊的预定义URL来触发新构建，请启用此选项。
2. Build after other projects are built:在其他项目触发的时候触发，里面有分为三种情况，也就是其他项目构建成功、失败、或者不稳定的时候触发项目；
3. Build periodically 定时构建
4. GitHub hook trigger for GITScm polling，根源Git的源码更新来触发构建
5. Poll SCM：定时检查源码变更（根据SCM软件的版本号），如果有更新就checkout最新code下来，然后执行构建动作。如下图配置：

*\*/5 \** \* \* \* （每5分钟检查一次源码变化）

#### jenkins定时构建语法

\* \* \* \* \*

(五颗星，中间用空格隔开）

* 第一个\*表示分钟，取值0~59
* 第二个\*表示小时，取值0~23
* 第三个\*表示一个月的第几天，取值1~31
* 第四个\*表示第几月，取值1~12
* 第五个\*表示一周中的第几天，取值0~7，其中0和7代表的都是周日

##### 使用案例

每天下午下班前18点定时构建一次

0 18 \* \* \*

每天早上8点构建一次

0 8 \* \* \*

每30分钟构建一次：

H/30 \* \* \* \*

补充资料：[Python邮件发送](http://www.51zxw.net/show.aspx?id=61323&cid=615)

#### 参考资料

1. <https://blog.csdn.net/vernice/article/details/46873169>
2. <https://blog.csdn.net/churximi/article/details/51648388>
3. <https://www.cnblogs.com/robert-zhang/p/9060365.html>
4. <https://baike.baidu.com/item/Jenkins/10917210?fr=aladdin>
5. [https://baike.baidu.com/item/持续集成/6250744](https://baike.baidu.com/item/%E6%8C%81%E7%BB%AD%E9%9B%86%E6%88%90/6250744)
6. <https://www.cnblogs.com/caoj/p/7815820.html>