**python+unittest+requests接口自动化测试**

# 应用

HTTP接口测试

# 优劣

优势：工具开源免费、可扩展性强、可定制报告样式、案例耦合度低

劣势：熟悉python语言和unittest模块

# 安装

工具准备：python(3.6)、Visual Studio Code(vscode)

## python安装

Python官网下载地址：https://www.python.org/

安装教程：http://www.runoob.com/python3/python3-install.html

## Vscode安装

微软免费代码编辑工具

下载地址：<https://code.visualstudio.com/>

使用教程：https://www.cnblogs.com/k0204/p/7189472.html

## 第三方模块安装

由于在使用过程中会使用第三方的python包，将使用到的定义在requirements.txt

requests-toolbelt==0.8.0

selenium==3.14.1

requests==2.21.0

PyMysql==0.9.2

bs4==0.0.1

requirements.txt安装方法：

pip install -r requirements.txt

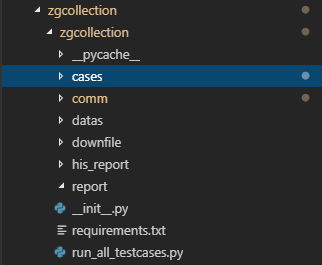
## 集合项目代码

GitLab： <http://192.168.6.32/interface/zgcollection.git>

SVN：https://192.168.130.29:8443/svn/testgroup/AutoTest/InterfaceTest/zgcollection

# 开始code

## 代码目录结构



\_\_pycache\_\_：Python的编译文件，可忽略

cases：测试案例

comm：通用方法（读取数据，发邮件，记录日志等）

datas：案例预埋数据

downfile：上传下载的文件

his\_report：历史报告

report：最新执行报告

requirements.txt：第三方包列表

run\_all\_testcases.py：项目的主入口（main函数）

## 编写testcase.py

**文件名格式test\_项目名称\_接口名.py**

一般一个接口定义一个testcase.py，流程性的也可以一个模块定义一个testcase.py

1. 指定文件编码类型（utf-8）
2. 文档注释(三个双引号，两行)
3. 引入要用的包或方法（注意引入顺序）

引入python自带的包

Import os

引入第三方的包

Import requests

引入自定义的方法

from comm.readjson import read

解释：从comm下的readjson中引入自定义的read()方法

read()方法说明：

读取datas目录中的json文件，并返回json格式的数据

1. 引入数据

user = read('userinfo.json')['userinfo']

cookie = read('cookies.json')

此处引入的变量为全局变量

## 定义测试类

类名一般用接口或者模块的名称

class Login(unittest.TestCase):

'登录接口'

1、继承unittest.TestCase，后续会用到TestCase中的方法

2、添加类的注释，在报告文件中会用到

3、定义案例执行前和执行后的行为

@classmethod

def setUpClass(cls):

pass

@classmethod

def tearDownClass(cls):

pass

setUpClass()/tearDownClass():每个用例类执行一次

SetUp()/teardown():每个用例函数执行一次

备注：步骤4中的数据也可以在setUpClass()中进行引入，代码更整洁

## 编写用例

def test\_a1\_sendsms(self):

1、定义一个方法（案例名称），格式为（test\_执行顺序\_接口名称）

2、添加案例的注释

3、准备此次案例需要的数据（此处定义的变量为局部变量）

4、构造请求

Post请求

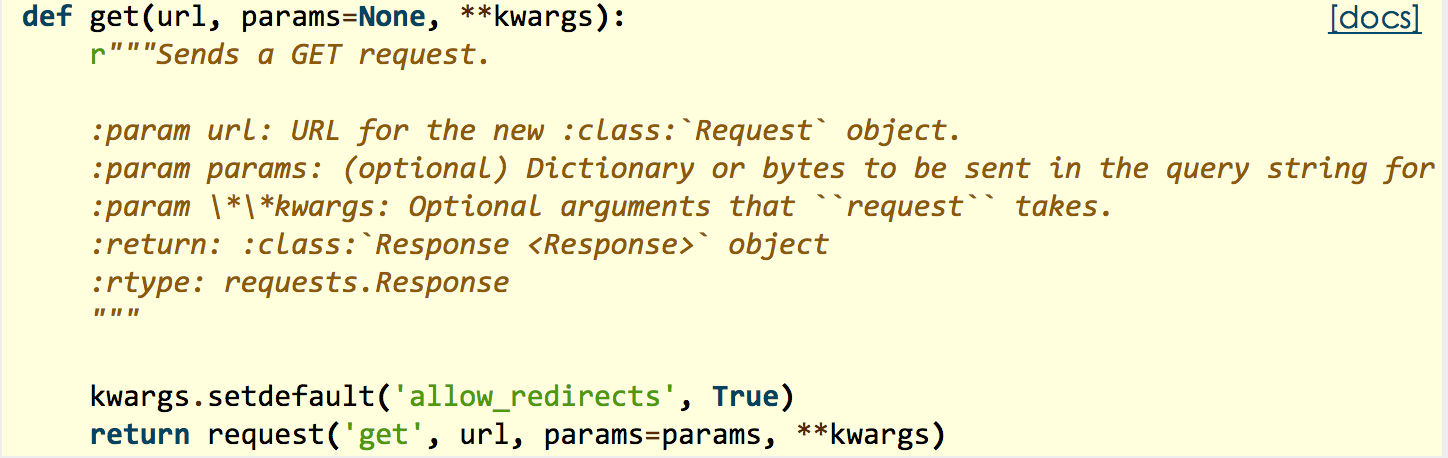
r = requests.post(url=url+'login',data=user, headers=header\_www, cookies=Login.mycookie)

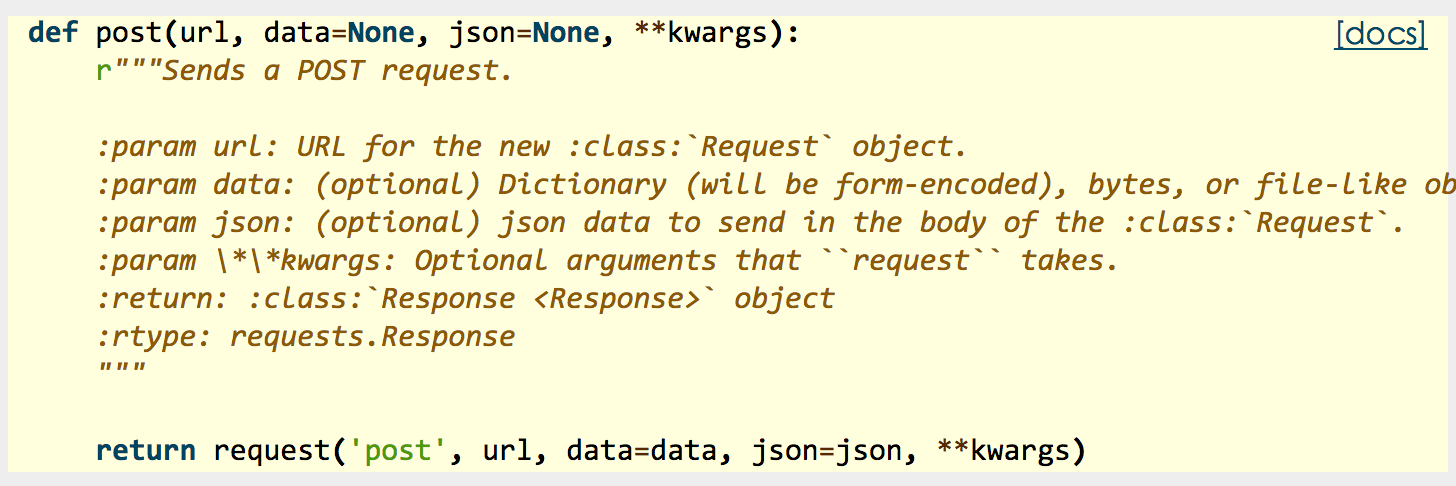
url:此次请求的接口地址

data:需要发送的数据

headers:请求头

cookies:这需携带cookie



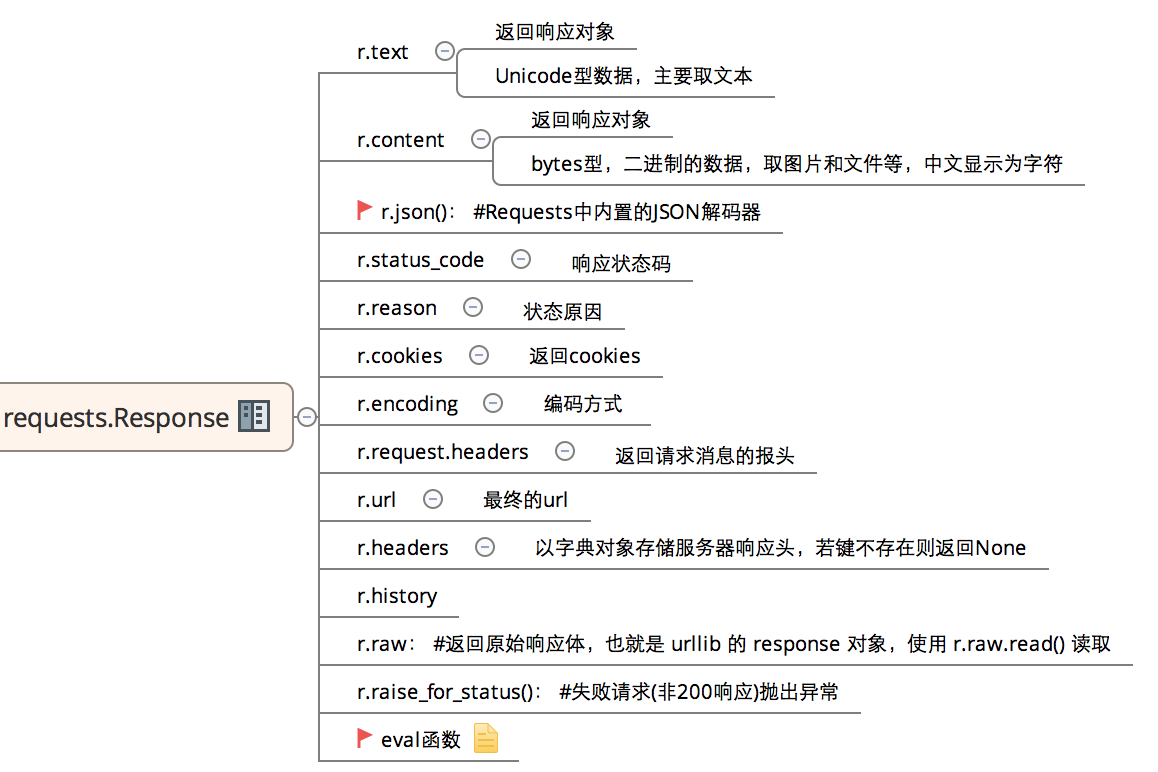


1. 查看请求成功后返回的数据

print(r.text)

print(r.json())

下图为请求返回数据的解析方式



6、添加断言

self.assertEqual('发送正常', r.json()['message'])



一条测试案例完成了

## 执行单个案例文件

1、手动添加案例

案例执行顺序为添加案例的顺序

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

suit = unittest.TestSuite()

suit.addTest(Login('test\_a1\_sendsms'))

suit.addTest(Login('test\_b1\_sendsms'))

suit.addTest(Login('test\_b2\_sendsms'))

suit.addTest(Login('test\_b3\_sendsms'))

runner = unittest.TextTestRunner()

runner.run(suit)

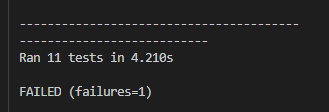
2、自动执行全部案例

按用例名ascii码先后顺序执行

unittest.main()

数字与字母的顺序为：0~9，A~Z,a~z

3、查看执行结果



执行11条案例，失败一条

## 执行整个项目的案例

项目编写完成后，批量执行多个py文件，并生成可视化测试报告

1、定义run\_all\_testcases.py文件

构造执行全部案例的方法

def all\_case():

'运行所有test\_\*.py的案例'

discover = unittest.defaultTestLoader.discover(case\_path, pattern="test\_\*.py",top\_level\_dir=None)

print (discover)

return discover

case\_path:案例所在的目录，本示例中为cases目录

pattern=“test\_\*.py” 执行所有以test\_开头，并以.py结尾的文件

2、定义执行方法，并生成报告（html格式）

fp = open(report\_abspsth, 'wb+')

runner = HTMLTestRunner.HTMLTestRunner(stream=fp, title=u"集合接口测试报告", description=u"用例执行情况")

# 执行所有的case

runner.run(all\_case())

fp.close()

report\_abspsth:测试报告文件（路径+文件名称）

## 其他同样模块（方法）：

### 1、记录日志：

将运行过程中的信息在屏幕输出，并记录到本地文件

def loginit():

'''

#定义日志格式，并将日志同时向屏幕输出并写入文件

'''

logging.basicConfig(level=logging.DEBUG,

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

#datefmt='%a, %d %b %Y %H:%M:%S',

datefmt= '%Y-%m-%d %H:%M:%S',

filename=logpath+"\\"+'zgcollection.log',

filemode='w')

#定义一个StreamHandler，将INFO级别或更高的日志信息打印到标准错误，并将其添加到当前的日志处理对象#

console = logging.StreamHandler()

console.setLevel(logging.INFO)

formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')

console.setFormatter(formatter)

logging.getLogger('').addHandler(console)

loginit()

logger=logging.getLogger(\_\_name\_\_)

使用方法：

先导入这个方法

from comm.logset import logger

再定义日志等级并写入日志内容

logger.info('执行成功')

### 2、发送邮件：

使用自带模块smtplib

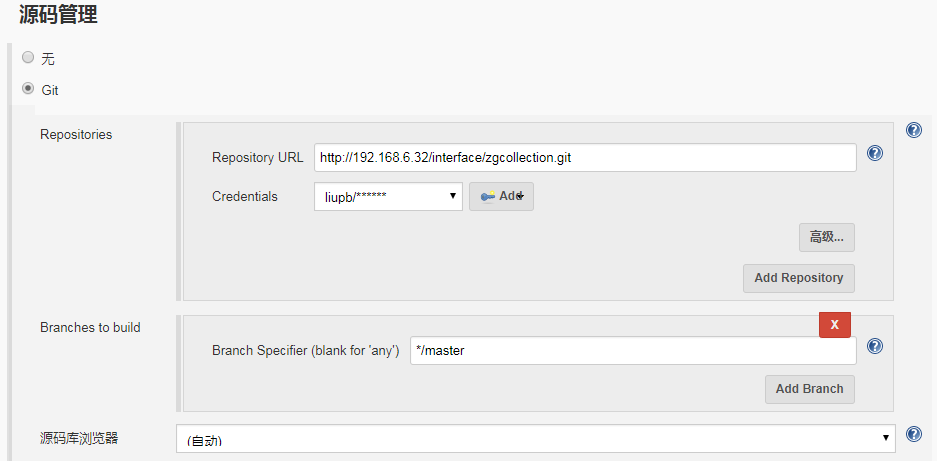
定义发送邮件的方法，并从配置文件中读取相应的参数

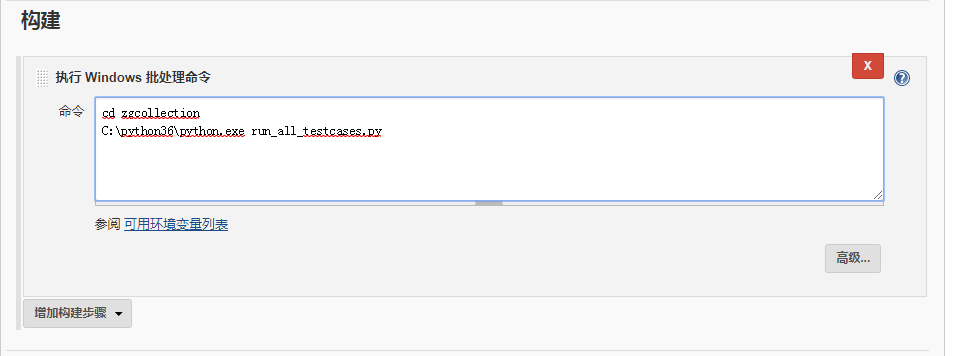
#执行完成 发送邮件

send\_email(report\_abspsth)

# 持续集成

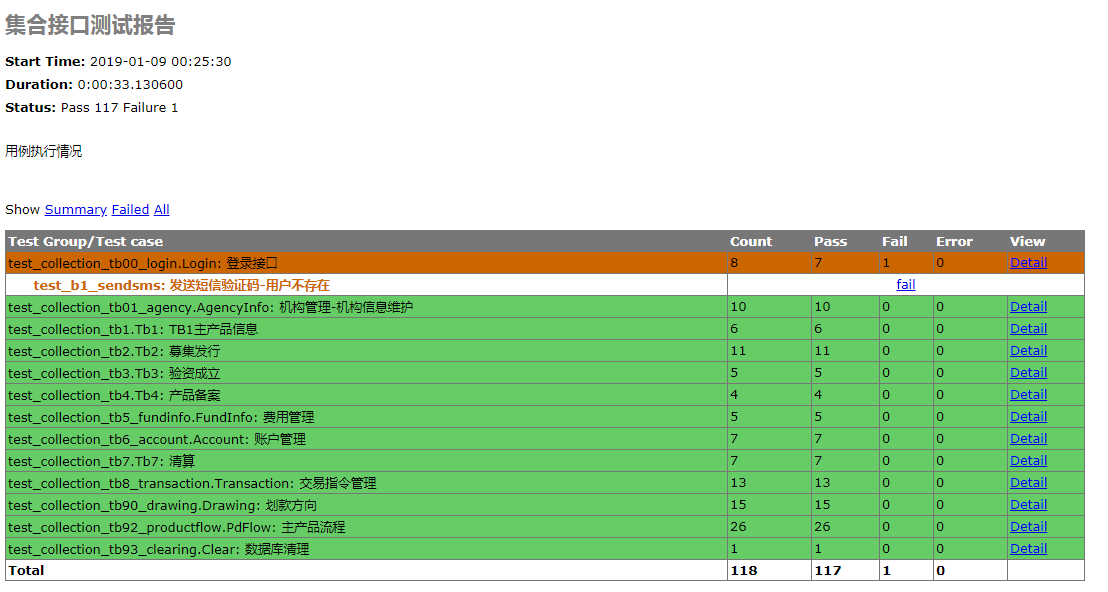
**模式：Jenkins+python+git**





邮件发送方式内置在项目中，无需在job中配置邮件发送

执行完成后，邮件收到如下报告



# 后续完善

1. 完善持续集成

Jenkins+python+git+Testlink

1. python接口性能测试

参考：https://www.jianshu.com/p/9d3f991c901a?\_wv=1031