Python+unittest+requests接口测试二

# 模块或方法的引用

## 引入内置模块

引入模块：

import os

import re

import sys

import base64

import datetime

import time

使用模块：

print(sys.path)

path为sys模块下的方法，可以通过 模块.方法 的方式进行使用

直接引入模块下的方法

from time import time, sleep

from os import path

使用方法

sleep(5)

## 引入开源模块

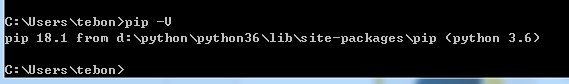
引入的目的：Python内置中没有这个模块，或者内置的模块不能满足需求

**1、安装要引入的模块**

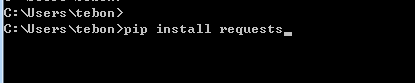
推荐pip在线安装：pip为Python内置的安装工具，在安装Python时，顺便将pip加入到环境变量







安装开源模块requests



pip install requests

**2、安装完成后，引入及使用的方式与内置模块一样**

import requests

## 引入自定义模块

* 导入一个py文件，解释器解释该py文件
* 导入一个包，解释器解释该包下的 \_\_init\_\_.py 文件

**1、自定义py文件或者模块**

如果定义一个模块，要在模块下创建 \_\_init\_\_.py文件

**2、确定包的路径**

print(sys.path)

如果sys.path路径列表没有你想要的路径，可以通过 sys.path.append('路径') 添加

fapath = os.path.dirname(os.path.dirname(\_\_file\_\_))

sys.path.append(fapath)

3、引入包（方法）、使用包（方法）

**示例演示（不推荐使用 from A import \* 的引入方式）**

参考：https://www.cnblogs.com/liu-yao/p/5186322.html

# 数据的引入

定义数据（json格式数据）

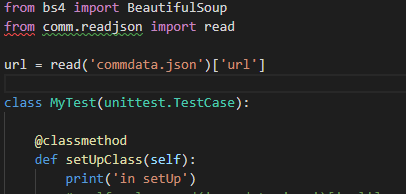


## 导入数据读取的方法

from comm.readjson import read

## 构造数据

1、在文件头构造



此时的url为全局变量

从commdata.json文件中读取url节点的值

def read(filename):

'读取json文件'

# 数据文件路径

path = os.path.join(os.path.dirname(os.path.dirname(\_\_file\_\_)), "datas\\"+filename)

datapath = os.path.abspath(path)

# 解析json文件

try:

with open(datapath, encoding='utf-8') as f:

datas = json.load(f)

# print(datas)

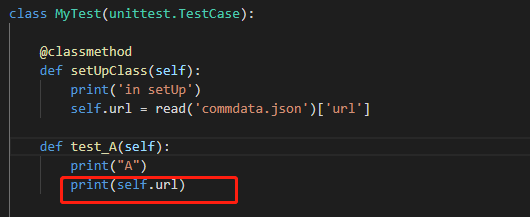
return datas

except FileNotFoundError:

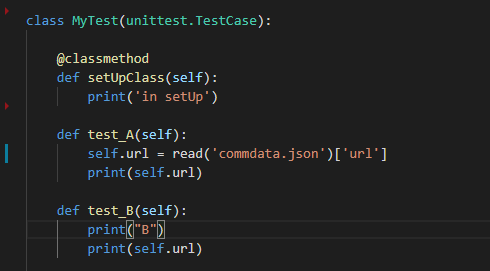
print('%s不存在，请检查'%filename)

2、在setUp()或者setUpClass()中构造

此时self.url为全局变量



3、在单个测试案例中构造



此时self.url为test\_A()的局部变量，如果在test\_B()中使用则会报错

# 发送请求

## get请求

r = requests.get(url=url, params=data, cookies=cookie)

url为接口地址

params为参数类型，值为data

## post请求

**消息体为表单**

def test\_b\_insert(self):

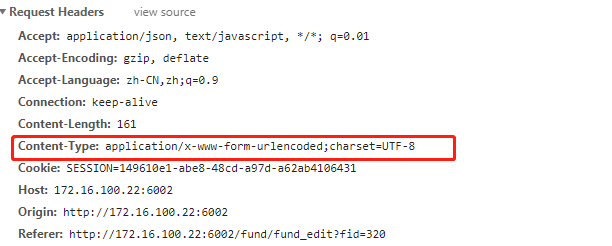
'新增主产品信息'

r = requests.post(url=url+'productinfo/insertOrUpdate', data=productinfo, headers=header\_www, cookies=cookie)

data消息体数据接收格式，一般默认都是这种样式

此时对应的header中

"Content-Type": "application/x-www-form-urlencoded;charset=UTF-8"



**消息体为json**

r = requests.post(url=url+'transferinstructflow/task/complete', json=trancomplete, headers=header\_json, cookies=cookie)

消息体数据接收格式为json

此时对应的header中

"Content-Type": "application/json"

接口文件中参考



## 文件处理

### 上传文件

**multipart/form-data: 表单数据编码为一条消息，每个控件对应消息的一部分**

def test\_k0\_upload(self):

'上传文件-pdf格式'

filedata = {"pid":FileTree.nodepid,"file":"1"}

namepath = os.path.join(os.path.dirname(os.path.dirname(\_\_file\_\_)), 'downfile\\'+'ApacheFlink.pdf')

filedata['file'] = ('ApacheFlink.pdf', open(namepath,'rb').read())

encode\_data = encode\_multipart\_formdata(filedata)

filedata = encode\_data[0]

header = {'Content-Type':'1'}

header['Content-Type'] = encode\_data[1]

r = requests.post(url=url+'document/upload', data=filedata, headers=header, cookies=cookie)

print(r.text) # 如果添加成功，r.text = '1'

self.assertEqual("操作成功", r.json()['message'])

filedata：定义此次请求要发送的数据；

namepaht：要上传的文件路径+文件名称

filedata['file'] = ('ApacheFlink.pdf', open(namepath,'rb').read())

将文件以二进制流数据形式打开，ApacheFlink.pdf：文件名称

encode\_data = encode\_multipart\_formdata(filedata)

将要上传的数据进行编码

filedata = encode\_data[0]

要上传的数据部分

header['Content-Type'] = encode\_data[1]

数据头格式

最后发送数据请求

### 文件下载

def test\_k2\_download(self):

'下载文件-pdf'

r = requests.post(url=url+'document/download', data={'id':(FileTree.docxid)}, headers=header\_www, cookies=cookie)

print(r.text)

self.assertEqual(200, r.status\_code)

与普通post请求一样

### 文件删除

def test\_m1\_delete(self):

'删除目录节点'

r = requests.post(url=url+'document/delete', data={'id':(FileTree.nodepid)}, headers=header\_www, cookies=cookie)

print(r.text)

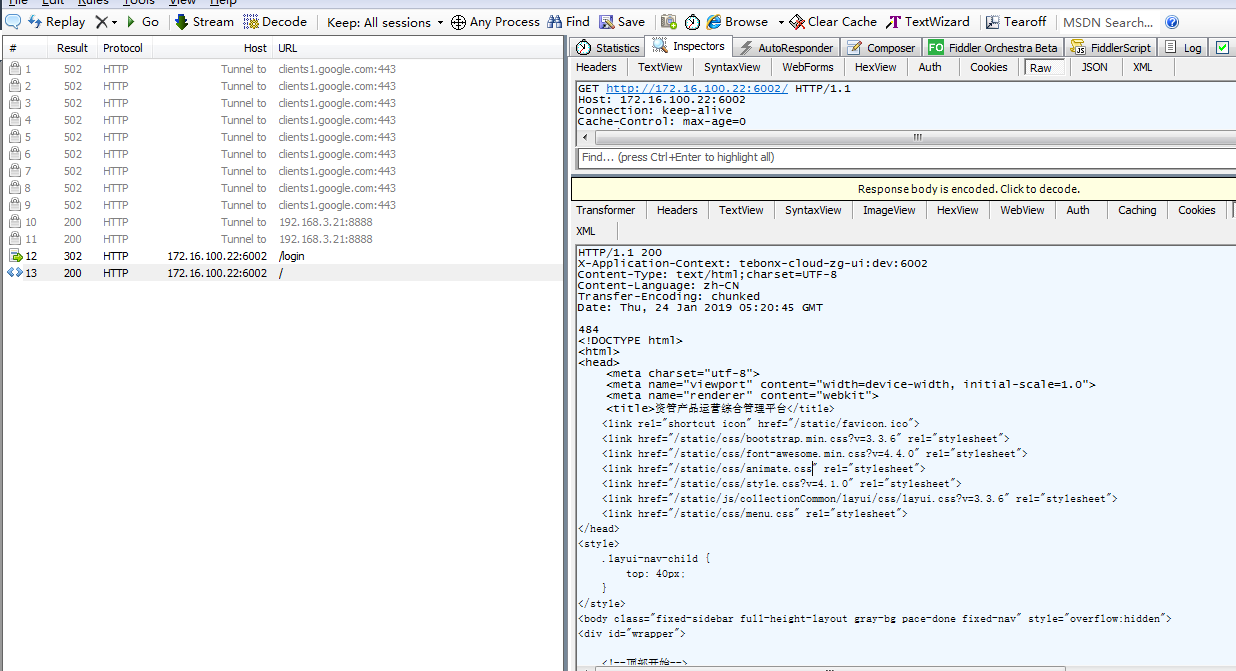
self.assertEqual("操作成功", r.json()['message'])

与普通post请求一样

# 回报数据和断言

## 响应数据为表单

集合的登录接口



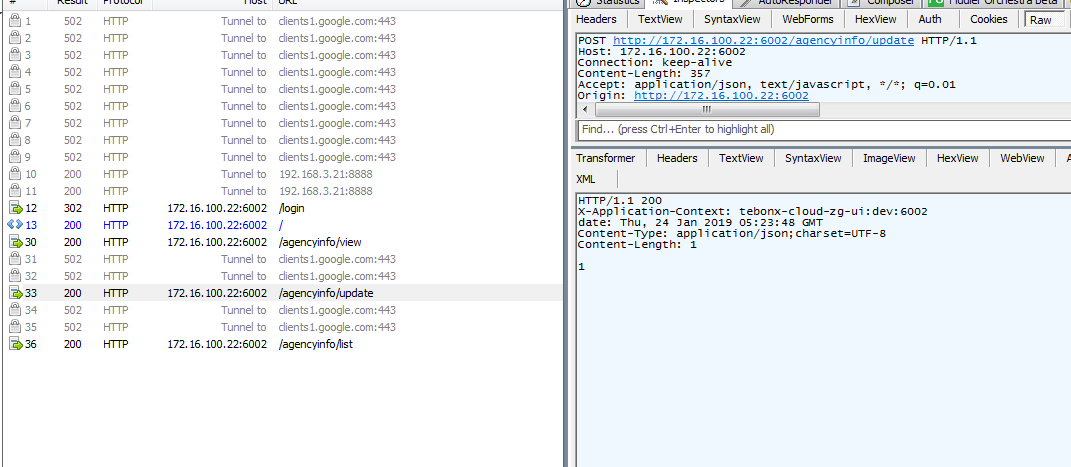
设置断言

self.assertIn('个人资料', r.text)

表示‘个人资料’这个字符串在返回的文本信息中

## 响应数据为文本

机构信息曾改

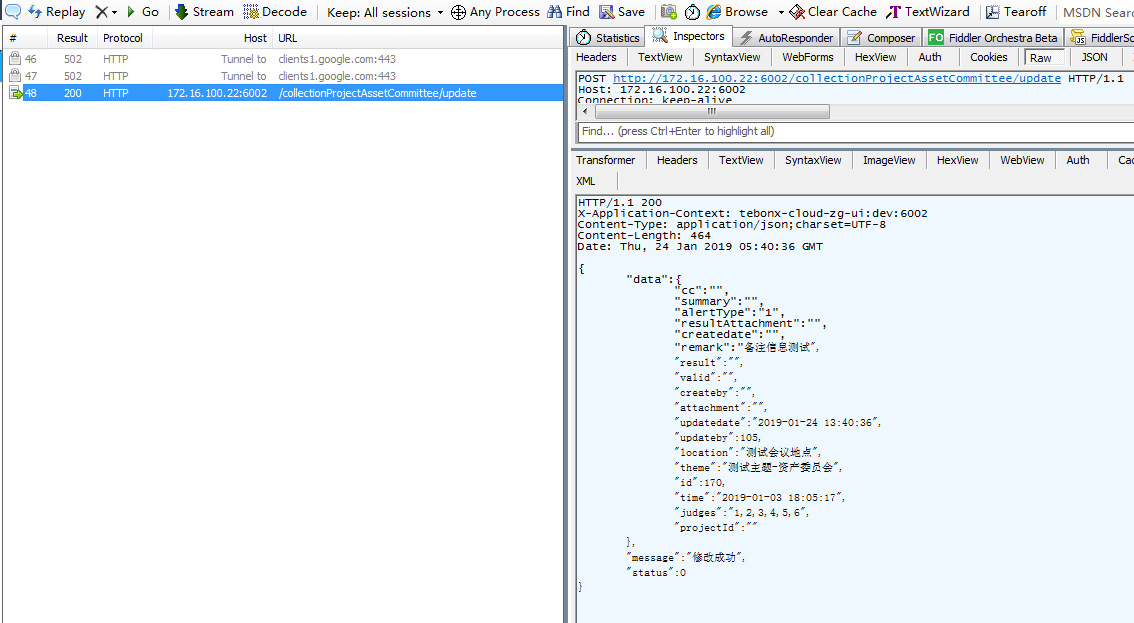


self.assertEqual('1',r.text)

返回的1表示成功，有的返回true或者false

## 响应数据为json

委员会信息修改



self.assertEqual("修改成功", r.json()['message'])

r.json()[‘message’]：获取message中的信息

## 响应数据json含有list

文档管理



响应数据中有list，list中有多个集合

断言两个list是否相等

def test\_a\_table(self):

'获取左侧目录列表'

r = requests.post(url=url+'document/table', data={"projectId":cdatas.projectid}, headers=header\_www, cookies=cookie)

print(r.text)

name\_list = []

for names in r.json():

name\_list.append(names['name'])

print(name\_list)

name\_list.sort()

directoryname.sort()

self.assertListEqual(name\_list, directoryname)

name\_list = [] #定义一个空的list

for names in r.json():

name\_list.append(names['name']) #在返回的数据中将数据添加到name\_list

print(name\_list)

name\_list.sort() # 将name\_list 进行排序（升序）

directoryname.sort() # 将目标数据也进行升序排序

self.assertListEqual(name\_list, directoryname)

比较两个list是否相同