0313课堂笔记

1.数据驱动

通常情况下我们不会直接将多组测试数据存储代码中，所以我们可以采用数据驱动的方式，将测试数据存储在存储介质上(excel/yaml/xml/数据库)，咱们在这里选择excel作为我们的数据载体

- 操作excel的第三方库安装

```
# windows
pip install openpyxl
pip install xlrd
pip install pandas
# 如果是苹果电脑
python3 -m pip install openpyxl
python3 -m pip install xlrd
python3 -m pip install pandas
```

windows电脑在安装时可能会报错缺失vc++组件，这个时候先尝试升级pip之后再装

- 数据存储结构

测试用例名称	sku_id	num	期望响应状态码	期望业务码	期望提示语
产品已下架	5875	1	500	"004"	产品已下架
产品id不存在	6636655	1	500	"451"	商品已失效，请刷新购物车
num超过库存	5173	99999999	500	"451"	商品库存已不足，不能购买。
num为0	5173	0	400	"004"	加入购物车数量必须大于0
num为负数	5173	-1	400	"004"	加入购物车数量必须大于0
产品id为空		1	400	"004"	产品id不能为空
num为空	5173		400	"004"	购买数量不能为空

- 封装excel数据读取

在common包下创建一个file_load.py文件，在这里封装各种文件的操作方法

```
# !/usr/bin python3
# encoding: utf-8 -*-
# @author: 沙陌 微信：Matongxue_2
# @Time: 2022-03-13 9:45
# @Copyright：北京码同学
import pandas
```

```
from setting import DIR_NAME


def read_excel(filepath,sheet_name):

    # keep_default_na 为false表示当你的单元格为空的时候，返回的是空字符串
    # engine表示选择的底层引擎是什么
    res =
pandas.read_excel(DIR_NAME+filepath,sheet_name=sheet_name,keep_default_na=Fa
lse,engine='openpyxl')
    # print(res)
    # 需要将拿到的数据转换成列表套列表的形式
    data = [] # 总的数据存储
    lines_count = res.shape[0] # 表示从行数,不包括表头
    cols_count = res.shape[1] #表示总列数
    # print(lines_count,cols_count)
    for l in range(lines_count):
        lines = []
        for c in range(cols_count):
            cell_data = res.iloc[l,c] #获取单元格数据
            lines.append(cell_data)#将单元格数据追加lines这个列表里
        # 遍历列完成后，当前行的数据都存在lines中了
        data.append(lines)
    return data
if __name__ == '__main__':
    print(read_excel('/data/mtxshop_testdata.xlsx', '添加购物车'))
```

执行时失败，修改测试用例中的处理

```
# !/usr/bin python3
# encoding: utf-8 -*-
# @author: 沙陌 微信：Matongxue_2
# @Time: 2022-03-06 15:57
# @Copyright：北京码同学
import pytest

from api.buyer.cart import AddCartApi
from common.file_load import read_excel
from setting import DIR_NAME


class TestAddCart:

    # test_data = [
    #     ['必填参数均正确没有非必填字段',5173,   1,       200,'',''],
    #     ['产品已下架',                5875,   1,      500,'004','产品已下架'],
    #     ['产品不存在',              456665,  1,      500,'451','商品已失效，请
刷新购物车'],
    #     ['购买数量超过库存',          5173, 99999999, 500,'451','商品库存已不
足，不能购买。'],
    #     ['购买数量为0',               5173,  0,        400,'004','加入购物车
数量必须大于0'],
    #     ['购买数量为负数',            5173,   -1,     400,'004','加入购物车数
量必须大于0'],
    #     ['购买数量为空',              5173,  None,    400,'004','购买数量不能
为空'],
```

```
    #     ['产品id为空',                None,      1,    400,'004','产品id不能为空']
    # ]
    test_data = read_excel('/data/mtxshop_testdata.xlsx', '添加购物车')


 @pytest.mark.parametrize('casename,sku_id,num,expect_status_code,expect_busi_code,expect_message', test_data)
    def test_buy_now(self, casename, sku_id, num, expect_status_code, expect_busi_code, expect_message):
        # 调用接口，传入测试数据
        add_cart = AddCartApi(sku_id=sku_id,num=num)
        resp = add_cart.send()
        status_code = resp.status_code
        assert status_code == expect_status_code
        print(resp.text)  # 由于响应信息可能为空，所以我们使用resp.text打印结果，因为空的字符串是无法使用resp.json()去获取的
        # 注意这个接口如果业务正常成功，那么响应信息是空的
        if status_code != 200:
            # 状态码不是200时，才进行响应信息的校验
            try:
                resp_json = resp.json()
            except:
                pass
            code = resp_json['code']
            # assert code == expect_busi_code

            pytest.assume(code == expect_busi_code.replace('"',''))
            # 判断message的值是 商品已失效，请刷新购物车
            message = resp_json['message']
            # assert message == expect_message
            pytest.assume(message == expect_message)
```

2.更多的接口定义实战

3.测试用例编写

接口用例可以分为单接口的测试和基于业务流程的测试

单接口测试通常可以做该接口的一些异常用例，基于业务流程一般是比较正常的业务

- 立即购买单接口测试

4.数据准备和清除

数据准备的几种方式：

1. 使用对应接口，现用现造，以及调用接口清除数据
2. 使用sql来完成数据的构造和清除

   造数据或者清除数据时，涉及到的表结构比较复杂，造的数据可能本身不完整，清除时也清除不完整
3. 数据库备份和还原

   我提前做好一堆数据，把这个库备份成B，每次执行测试前先备份执行环境的数据库A，再还原我之前备份好的测试数据库B，执行完测试之后，再将其还原A

对于清除数据，其实也可以不清，可以清除

5.基于订单流程的测试用例

```
# !/usr/bin python3
# encoding: utf-8 -*-
# @author: 沙陌 微信：Matongxue_2
# @Time: 2022-03-13 14:31
# @Copyright：北京码同学
import time

import jsonpath
import pytest

from api.buyer.cart import BuyNowApi, DeleteCartApi, AddCartApi
from api.buyer.comment import CommentApi
from api.buyer.create_trade import CreateTradeApi
from api.buyer.orders import ConfirmOrgApi
from api.seller.order import OrderDeliveryApi, OrderPayApi
from common.file_load import read_excel


class TestOrderProcess:
    order_sn = ''
    pay_price = 0
    sku_id = ''
    # 买家下单
    test_data = read_excel('/data/mtxshop_testdata.xlsx', '创建交易')
    @pytest.mark.parametrize('casename,client,way,expect_statuscode',test_data)
    def test_create_trade(self,casename,client,way,expect_statuscode,get_goods):
        goods_id = get_goods[0]
        TestOrderProcess.sku_id = get_goods[1]
        # 创建交易需要调用立即购买接口或者添加购物车接口来提供数据
        if way == 'BUY_NOW':
            # 需要sku_id，不能写死，数据从哪里来
            BuyNowApi(sku_id=TestOrderProcess.sku_id,num=1).send()
        elif way == 'CART':
            # 清空购物车
            DeleteCartApi().send()
            AddCartApi(sku_id=TestOrderProcess.sku_id,num=1).send()
        resp = CreateTradeApi(client=client,way=way).send()
        pytest.assume(resp.status_code == 200)
        resp_json = resp.json()
        TestOrderProcess.order_sn = jsonpath.jsonpath(resp_json,'$..sn')[0]
        TestOrderProcess.pay_price =
jsonpath.jsonpath(resp_json,'$..total_price')[0]

    # 卖家发货
    def test_delivery(self):
        time.sleep(1)
        resp = OrderDeliveryApi(order_sn=TestOrderProcess.order_sn).send()
        pytest.assume(resp.status_code == 200)
    # 买家收货
    def test_org(self):
        time.sleep(1)
        resp = ConfirmOrgApi(order_sn=TestOrderProcess.order_sn).send()
        pytest.assume(resp.status_code == 200)
    # 卖家收款
    def test_confirm_pay(self):
```

```
        time.sleep(1)
        resp =
OrderPayApi(order_sn=TestOrderProcess.order_sn,pay_price=TestOrderProcess.pay_pr
ice).send()
        pytest.assume(resp.status_code == 200)
    # 买家评论
    def test_comment(self):
        time.sleep(1)
        resp =
CommentApi(order_sn=TestOrderProcess.order_sn,sku_id=TestOrderProcess.sku_id).se
nd()
        pytest.assume(resp.status_code == 200)
```

6.集成redis和数据库

把之前封装好的redis和数据库复制到common目录下，在conftest.py中增加redis和数据库的初始化

在订单流程用例里增加确认发货后的数据库订单状态断言

7.添加商品的数据驱动

由于json参数本身很多，所以在做数据驱动时不方便将每个参数当做一列数据，所以我们将整体数据当做一列参数

```
# !/usr/bin python3
# encoding: utf-8 -*-
# @author: 沙陌 微信：Matongxue_2
# @Time: 2022-03-13 15:29
# @Copyright：北京码同学
import pytest

from api.seller.goods import AddGoodsApi
from common.file_load import read_excel


class TestAddGoods:
    test_data = read_excel('/data/mtxshop_testdata.xlsx','添加商品')
    @pytest.mark.parametrize('casename,jsonparams,expect_stauscode',test_data)
    def test_add_goods(self,casename,jsonparams,expect_stauscode):
        jsonparams = eval(jsonparams) #由于从excel读出的json参数是一个字符串，所以我们
将其转换字典
        add_goods_api = AddGoodsApi()
        add_goods_api.json = jsonparams #设置接口的json属性为excel读出的数据
        resp = add_goods_api.send()
        assert resp.status_code == expect_stauscode
```

8.配置文件抽取

- 抽取各个服务

  在config目录下创建一个http.yml、db.yml、redis.yml、common.yml

- 读取yml文件

  1. 安装yml文件操作的库

     先导入试试，如果能导入无需执行下述安装

```
# windows
pip install PyYAML
# mac
python3 -m pip install PyYAML
```

2. 在common下的file_load.py中增加yaml文件读取的方法

```
def load_yaml_file(filepath):
    with open(DIR_NAME+filepath,mode='r',encoding='UTF-8') as f:
        yaml_content = yaml.load(f,Loader=yaml.FullLoader)
        return yaml_content
```

9.加解密处理

md5加密

aes加密：对称加解密，加密时和解密时都需要用到一个私钥，这个私钥加密和解密是相同的，私钥是要找开发去要的，私钥的长度是16的倍数

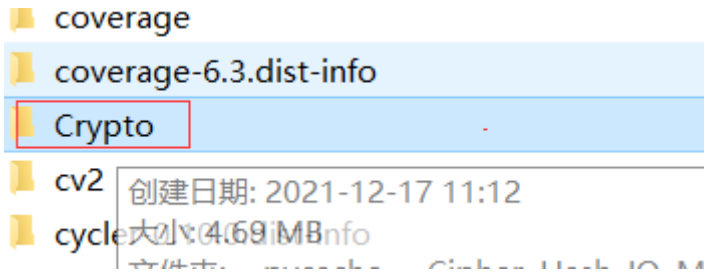
rsa加密：非对称加解密，加密和解密时用到的私钥不一样，rsa有两种钥匙，一种叫做公钥，一种叫做私钥，如果用公钥加密那么就用私钥解密，如果用私钥解密那么就用公钥解密，注意私钥都是掌握在自己手里，公钥是可以掌握在他人手里的

- 先安装第三方库

```
# windows
# windows 安装vc++14 可能需要,那就升级下pip再重新装
pip install pycryptodome -i https://pypi.douban.com/simple
# mac
python3 -m pip install pycryptodome -i https://pypi.douban.com/simple
```

装完之后要在本地的第三方库目录中，D:\Programs\Python\Python38\Lib\site-packages

默认装完后crypto目录的c小写的，我们要将其改成大写



注意导包：

```
import base64
import hashlib
from Crypto import Random

from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5 as PKCS1_signature
from Crypto.Cipher import PKCS1_v1_5 as PKCS1_cipher, AES
```

注意如果你们的接口是统一性的加解密方案，那么就在基类中完成加密和解密

如果只是单独的某些接口进行了需要加解密，那么就在单接口中定义完成

10.优化allure报告

给用例增加上一些分类标签

11.集成环境切换

设计思路：

将每个环境对应的各种信息，通过yaml进行配置，在执行run.py中增加接收外部参数的功能，参数就是环境的名称

比如在命令行执行

```
python run.py test
python run.py prepro
python run.py pro
```

代码实现