0320课堂笔记

1.针对RequestsClient.py做扩展

对于所有接口来说，都有可能会从响应中提取数据，这个时候我们可以封装一个通过jsonpath来提取数据的方法

```
    # 通过jsonpath去提取响应中的数据
    # index表示要得到匹配结果中的第几个
    def extract_resp(self,jsonpath_express,index=0):
        """

        :param jsonpath_express: jsonpath表达式
        :param index: 大于0时返回匹配到的第几个，小于0时返回匹配到所有
        :return:
        """
        text = self.resp.text #获取响应结果的文本形式
        if text != '':
            resp_dict = json.loads(text) #将json格式的字符串转成字典形式
            if index>=0:
                try:
                    res = jsonpath.jsonpath(resp_dict,jsonpath_express)[index]
                    self.logger.info(f'通过{jsonpath_express}在{text}提取成功:
{res}')
                except Exception as e:
                    self.logger.exception(f'通过{jsonpath_express}在{text}提取失
败')
                    raise e
            else:
                res = jsonpath.jsonpath(resp_dict, jsonpath_express)
                self.logger.info(f'通过{jsonpath_express}在{text}提取成功:{res}')
        else:
            self.logger.exception('响应为空无法提取数据')
            raise Exception('响应为空无法提取数据')
        return res
    def get_status_code(self):
        return self.resp.status_code
    def get_resp_text(self):
        return self.resp.text
    def get_resp_json(self):
        return self.resp.json()
```

2.取消订单异常业务测试

异常场景主要是和订单状态有关，比如已发货不能取消，已收货不能取消，已确认收款不能取消

```
# !/usr/bin python3
# encoding: utf-8 -*-
# @author: 沙陌 微信：Matongxue_2
# @Time: 2022-03-20 9:50
# @Copyright：北京码同学
from api.buyer.cart import BuyNowApi
from api.buyer.create_trade import CreateTradeApi
from api.buyer.orders import CancelOrderApi, ConfirmOrgApi
```

```python
from api.seller.order import OrderDeliveryApi, OrderPayApi


class TestCancelOrder:


    def test_alreadysend_cancel(self,get_goods):
        goods_id,sku_id = get_goods
        # 先得造已发货的订单数据
        BuyNowApi(sku_id).send()
        create_trade_api = CreateTradeApi()
        create_trade_api.send()
        order_sn = create_trade_api.extract_resp('$..sn')
        pay_price = create_trade_api.extract_resp('$..total_price')
        OrderDeliveryApi(order_sn=order_sn).send()
        # 取消
        cancel_order_api = CancelOrderApi(order_sn=order_sn)
        cancel_order_api.send()
        resp_status_code = cancel_order_api.get_status_code()
        assert resp_status_code == 400 #注意这个接口有bug，不管啥都是取消成功
    def test_alreadyshouhuo_cancel(self,get_goods):
        goods_id,sku_id = get_goods
        # 先得造已收货的订单数据
        BuyNowApi(sku_id).send()
        create_trade_api = CreateTradeApi()
        create_trade_api.send()
        order_sn = create_trade_api.extract_resp('$..sn')
        pay_price = create_trade_api.extract_resp('$..total_price')
        OrderDeliveryApi(order_sn=order_sn).send()
        ConfirmOrgApi(order_sn=order_sn).send()
        # 取消
        cancel_order_api = CancelOrderApi(order_sn=order_sn)
        cancel_order_api.send()
        resp_status_code = cancel_order_api.get_status_code()
        assert resp_status_code == 400 #注意这个接口有bug，不管啥都是取消成功
    def test_alreadyshouhuo_cancel(self,get_goods):
        goods_id,sku_id = get_goods
        # 先得造已收款的订单数据
        BuyNowApi(sku_id).send()
        create_trade_api = CreateTradeApi()
        create_trade_api.send()
        order_sn = create_trade_api.extract_resp('$..sn')
        pay_price = create_trade_api.extract_resp('$..total_price')
        OrderDeliveryApi(order_sn=order_sn).send()
        ConfirmOrgApi(order_sn=order_sn).send()
        OrderPayApi(order_sn=order_sn, pay_price=pay_price).send()
        # 取消
        cancel_order_api = CancelOrderApi(order_sn=order_sn)
        cancel_order_api.send()
        resp_status_code = cancel_order_api.get_status_code()
        assert resp_status_code == 400 #注意这个接口有bug，不管啥都是取消成功
```

3.如何提升执行效率

接口自动化本身执行是比较快的，但是不排除我有成千上万条接口用例需要执行，按照1秒钟执行完3个条测试用例的话，那么1万条用例执行时间是10000/3=3333秒，差不多一个小时，时间有点久，我们希望能快点执行完

可以利用并发的方式去执行，pytest有一个插件pytest-xdist可以实现多进程并发

- 安装插件

```
# windows
pip install pytest-xdist
# mac
python3 -m pip install pytest-xdist
```

- 配置pytest.ini中的参数

  修改addopts参数，在最后增加-n 2，表示启动两个进程来执行测试，

  默认的分配进制：会将每个用例作为最小单位，分配到两个进程上去，但是用例之间可能存在依赖，分配到不同进程后必然导致失败，所以我们尽量不使用默认分配机制

  可以使用另外一种配置机制

  修改addopts参数，在最后增加-n 2 --dist=loadfile，也是两个进程，--dist=loafile表示以每一个测试文件作为最小单位进行分配

  进程数量自动生成，-n auto，auto就表示会根据你当前的电脑配置自动生成进程数量

- 并发执行的前提

  在用例层级尽量减少跨文件的依赖

4.如何提升自动化稳定性

造成不稳定的原因

- 数据问题

  有一些接口使用了死数据，所以我们尽量现用现造

- 网络波动

  突然间的网络波动导致接口请求失败

  可以采用失败重试的策略，需要借助pytest的插件pytest-rerunfailures

  安装

```
# windows
pip install pytest-rerunfailures
# mac
python3 -m pip install pytest-rerunfailures
```

  修改pytest.ini中的addpopts参数，在其最后增加-rerun 2， 2就代表失败重试的最大次数

5.关键字驱动或者说纯数据驱动设计

对于excel表格的设计，但是注意excel并不是唯一的数据存储介质，你可以存储到任意文件或者数据库

- 测试集合管理设计

  在excel中单独弄一个sheet工作表，起名字叫测试用例集合

测试集合名称	是否执行
立即购买测试集合	y
添加购物车测试集合	y
订单流程测试集合	y

通过是否执行这一列来控制某一个测试集合

测试集合名称对应的某一个sheet工作表的名称

- api设计

在excel中单独弄一个sheet工作表，起名字叫api，该sheet工作表是专门用来定义管理单独接口的

接口名称	url	method	headers	默认参数
买家登录	${buyerHost}/passport/login	post		{ "params":{ "username":"shamo", "password":"${{md5(123456)}}", "captcha":"1512", "uuid":"hjsdhdfhhf" } }
卖家登录	${sellerHost}/seller/login	get		{ "params":{ "username":"shamoseller", "password":"${{md5(123456)}}", "captcha":"1512", "uuid":"hjsdhdfhhf" } }

接口名称：接口的名字

url：接口地址，注意域名要进行统一管理，否则修改时会比较麻烦，将公共域名定义成变量

method：接口的请求方式

headers：接口的头信息，必须以json格式字符串的形式进行填写，比如下面

```
{
 "Authorization":"${buyerToken}",
 "Content-Type":"application/json"
}
```

默认参数：指的是该接口在作为依赖接口传递，不需要传什么数据时，都会采用默认数据进行发起，由于参数的类型有多种，比如查询参数、json参数、表单参数、文件参数，设计是这样的，也是json格式字符串

```
{
"params":{
            "ship_no":"sddffdhjfhhf",
            "logi_id":12,
            "logi_name":"中通"
        },
 "data":{
            "aaa":"xxx"
        },
 "json":{
            "aaa":"xxxx"
         }
}
```

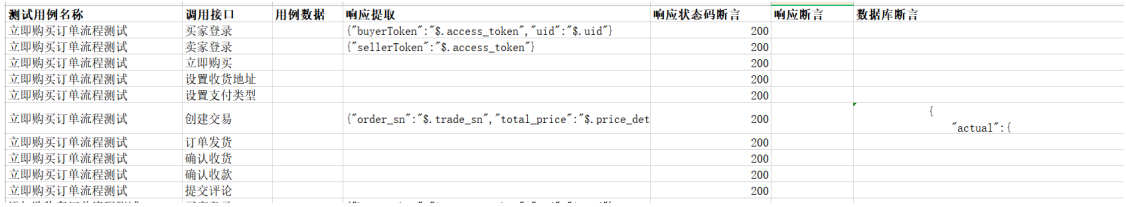
- 全局变量

  将公共型的数据进行配置，在需要的地方通过调用变量的方式进行引用，调用变量的方式是${xxxxx}这个xxxxx就是变量名称

变量名称	变量值
buyerHost	http://www.mtxshop.com:7002
sellerHost	http://www.mtxshop.com:7003

- 测试集合设计

  一个测试集合就是一个sheet工作表，一个测试集合中可以有多条用例



测试用例名称	调用接口	用例数据	响应提取	响应状态码断言	响应断言	数据库断言
立即购买订单流程测试	买家登录		{"buyerToken":"$.access_token","uid":"$.uid"}	200		
立即购买订单流程测试	卖家登录		{"sellerToken":"$.access_token"}	200		
立即购买订单流程测试	立即购买			200		
立即购买订单流程测试	设置收货地址			200		
立即购买订单流程测试	设置支付类型			200		
立即购买订单流程测试	创建交易		{"order_sn":"$.trade_sn","total_price":"$.price_det	200		{ "actual":{
立即购买订单流程测试	订单发货			200		
立即购买订单流程测试	确认收货			200		
立即购买订单流程测试	确认收款			200		
立即购买订单流程测试	提交评论			200		

  - 测试用例名称：测试用例名称相同的代表的是一条用例
  - 调用接口：指的该用例这一步需要调用的接口名称，必须和api中定义的接口名称相同
  - 用例数据：该用例在调用接口时，需要传递的测试数据，如果不需要测试数据就为空，会使用某个接口的默认数据发起调用
  - 响应提取：一个用例中可能存在接口的关联，所以我们需要在这里填写你要提取的数据，填写规则设计如下，以json格式字符串进行编写，key是你提取后要保存的变量名称，value是jsonpath表达式

```
{
    "buyerToken":"$.access_token",
    "uid":"$.uid"
}
```

  - 响应状态码断言：响应状态码的期望值
  - 响应断言：响应信息中的某些字段的期望值，响应信息中可能存在很多字段，如何填写，也是json格式字符串的方式进行填写，最外层是一个列表形式，里边套的一个一个的json串

```
[
    {
        "actual":"$.message",
        "expect":"商品已失效，请刷新购物车"
    },
    {
        "actual":"$.code",
        "expect":"004"
    }
]
```

  - 数据库断言：判断响应信息中和数据库中的某些字段是否一致，设计如下

```
{
    "actual":{
        "trade_sn":"$.trade_sn",
        "order_price":"$.price_detail.total_price"

    },
    "expect":{
        "sql":"SELECT trade_sn,order_price FROM mtxshop_trade.es_order
WHERE member_id=${uid} ORDER BY order_id DESC LIMIT 1;"
    }
}
```

6.实现底层代码解析

7.git代码版本管理

- 远程仓库

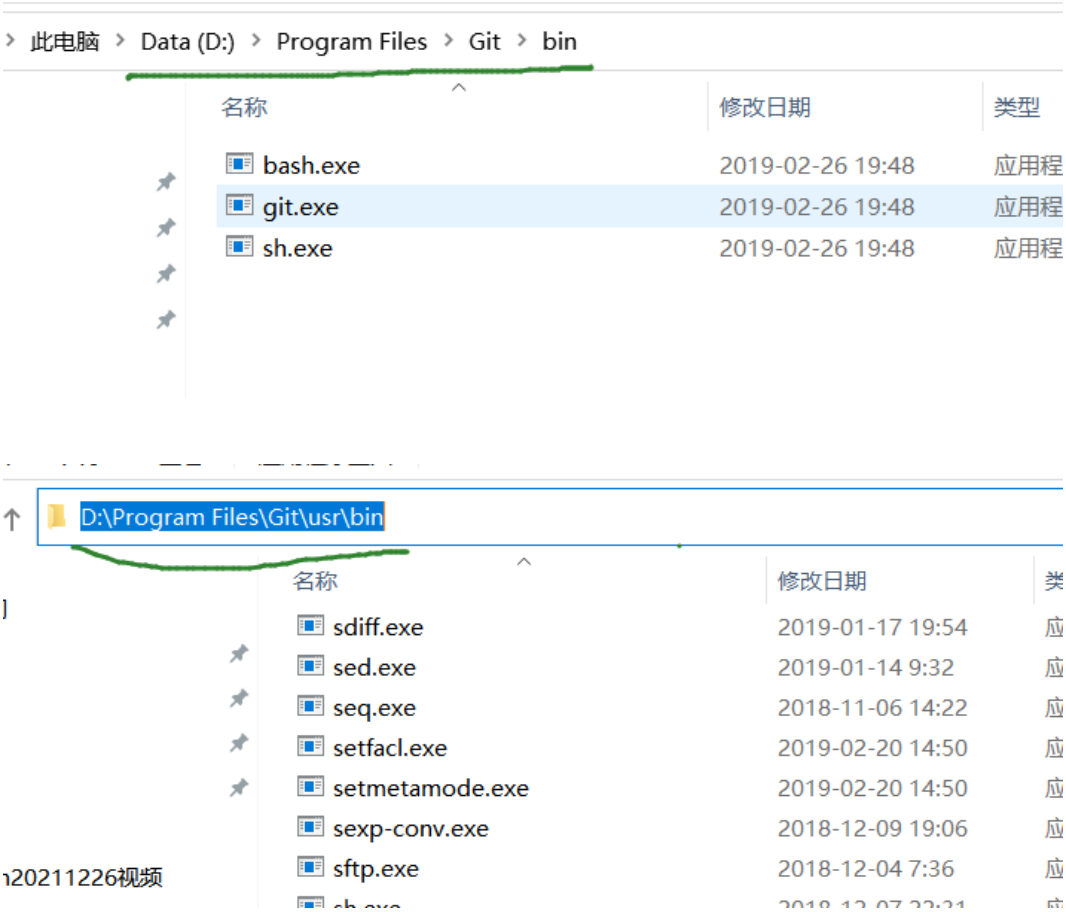
  在公司里一般会有公司自己搭建的一个仓库，在咱们课上使用 码云 这个平台来作为远程代码管理仓库

  https://gitee.com/ 自行注册账号

- 客户端

  客户端指的就是自己本地安装的git工具

  windows的下载群文件exe文件进行安装，安装完成之后要配环境变量，下面的路径追加到环境变量path



#mac的话可以使用brew install git去安装

- 配置本地git账户

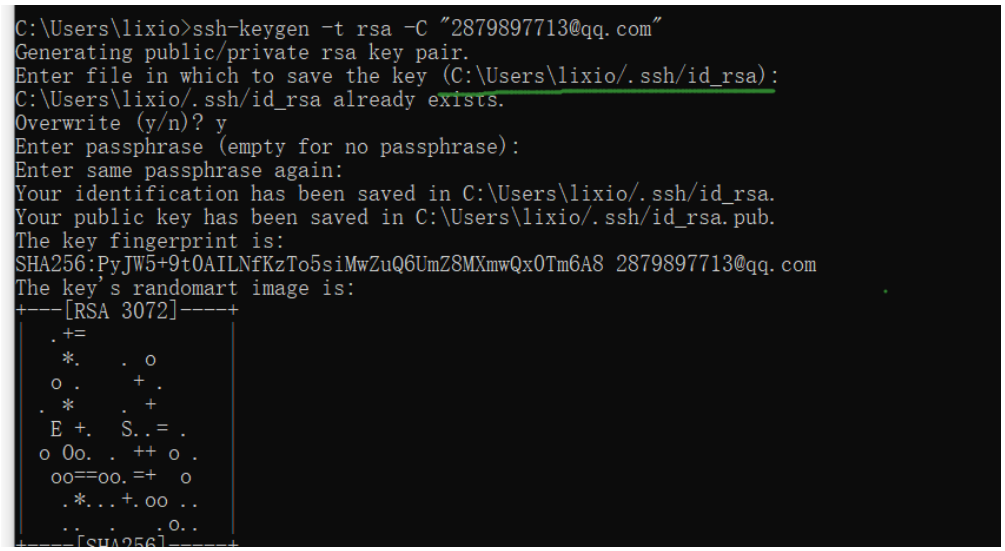
  账户是你在码云上注册的用户名和绑定的邮箱

```
git config --global user.name "sandroad"
git config --global user.email "2879897713@qq.com"
```

- 生成公钥和私钥

```
ssh-keygen -t rsa -C "2879897713@qq.com"
```

第一次时连敲三次回车



复制本地公钥到码云平台上进行配置，进入设置-->安全设置下的ssh公钥

然后尝试测试连接

```
ssh -T git@gitee.com
```

- 代码提交

  首先在远程仓库创建一个仓库

  pycharm命令行提交方式

  1. 进入pycharm命令行
  2. 初始化项目

```
git init
```

  3. 添加到git本地缓存区

```
git add *
```

  4. 提交到git本地仓库

```
git commit -m "第一次提交"
```

5. 添加远程仓库地址到本地

```
git remote add origin
https://gitee.com/sandroad/apiautotest0320.git
```

6. 提交代码到远程仓库

```
git push -u origin master
```

7. 修改一个文件提交步骤

```
# 添加你修改的文件到缓存区
git add keywords\keyword_run.py
# 提交到本地仓库
git commit -m "只是加个注释"
# 提交到远程仓库
git push -u origin master
```