金融项目第八天--课堂笔记

昨日回顾

（1）Jmeter脚本自动化测试执行

- 数据清理
  - 方式1：
    - 在Jmeter添加JDBC连接池
    - 再JDBC Request请求，有多少个（清除数据）SQL语句就添加多少个请求
  - 方式2：
    - 在Jmeter添加JDBC连接池，在数据库URL配置中增加一个参数（allowMultiQueries=true）
    - 再添加一个JDBC Request（Type：Callable statement），填写所有的清除数据的SQL数据

> 所有的SQL语句中的过滤条件都不用写成固定值，需要从用户定义的变量中引用，便于维护

- 自动化脚本执行：
  - 将测试计划中的”独立运行每个线程组“，保证所有的脚本顺序执行
  - 将清理测试数据的脚本放入到teardown线程组中，保证最后执行完成后清理数据
- 生成测试报告：
  - jmeter -n -t 测试计划路径 -l 测试结果数据 -e -o 测试报告的路径
- 将Jmeter脚本部署到持续集成平台
  - 添加一个持续集成项目
  - 配置持续集成项目
    - 配置构建定时器（每个开发合入代码就构建、1天、3天）
    - 配置构建时命令：执行Jmeter测试脚本的命令
      - 先删除 持续集成项目目录 下的Jmeter测试报告（上一次执行的报告）
      - 生成执行报告命令：jmeter -n -t 测试计划路径 -l 测试结果数据 -e -o 测试报告的路径
    - 配置构建后的邮件通知：
      - Publish HTML Report：发送邮件的报告的路径
      - Email配置：Email中的报告格式、Email的收件人、Email发送的触发条件

（2）Python代码自动化

- python代码自动化的目录结构
  - api
  - script
  - log
  - lib
  - report
  - data
  - app.py
  - utlis.py

    - runsuite.py
- 安装依赖包
    - requests、parameterized、pymysql： pip install 包名
    - HTMLTestRunner：下载HTMLTestRunner包，拷贝到lib目录下
- python代码自动化的框架
    - 定义日志初始化配置
    - 编写脚本
        - 定义接口
        - 编写脚本——调用接口，完成对应的业务测试
    - 编写测试套件，并生成测试报告

学习目标

- 能够使用Python+Request编写接口自动化测试脚本
- 清除测试数据

普通接口测试脚本：

1、定义接口脚本

- 定义发送接口请求所需的url和请求参数
    - API接口定义时，参数可以设置默认值，减少脚本调用时传参的工作量
- 调用request的get/post方法来发送请求
    - 如果是multi-part多消息体数据，传递参数的方法，需要传递多种数据格式：response = session.post(self.approve_url,data=data,files={'x':'y'})
- 返回response响应结果

2、按照测试用例编写测试脚本

- 按照接口API定义，准备需要传递的参数
- 传入参数，并调用接口API中对应的方法发送请求
    - 如果有多个参数，只有部分参数需要传递时，可以只传入前几个，或者写明要传入参数的参数名
- 接收返回的响应，并按照用例预期结果编写断言

3、编写测试脚本中部分常用的动作，可以定义到utils工具类中，方便后续调用

第三方接口测试脚本：

1、作用

BeautifulSoup包：python提供的专门用于解析HTML/XML文件并读取其中的数据内容

2、安装：

pip install beautifulsoup4（必须带上4，表示第4个版本，不带4会安装第3个版本）

3、基本使用

```
from bs4 import BeautifulSoup
                    如果index.html为文件名，参数为：open(文件名)
soup = BeautifulSoup(open("index.html"), "html.parser")
                                                  python中的html解析器

soup = BeautifulSoup("<html>data</html>", "html.parser")
                  如果参数为HTML字符串的内容，参数：html内容的字符串
```

4、常用的BeautifulSoup的方法：

```
soup = BeautifulSoup(html,'html.parser')            初始化soup对象，解析后的html
#获取title的对象
print(soup.title)          获取title标签的对象元素
#获取title的标签名称
print(soup.title.name)     获取title页面对象元素的标签名
#获取title的值
print(soup.title.string)   获取title页面对象元素的值

#获取p的对象
print(soup.p)
#获取所有的P的对象
print(soup.find_all('p'))  获取页面中所有p标签的对象元素
#获取第一个P标签对应的ID属性的值
print(soup.p['id'])        获取P标签的id属性的值
```

```
from bs4 import BeautifulSoup

html = """
<html>
<head><title>黑马程序员</title></head>
<body>
<p id="test01">软件测试</p>
<p id="test02">2020年</p>
<a href="/api.html">接口测试</a>
<a href="/web.html">Web自动化测试</a>
<a href="/app.html">APP自动化测试</a>
</body>
</html>
"""
soup = BeautifulSoup(html,'html.parser')
#获取title的对象
print(soup.title)
#获取title的标签名称
print(soup.title.name)
#获取title的值
print(soup.title.string)

#获取p的对象
print(soup.p)
#获取所有的P的对象
print(soup.find_all('p'))
#获取第一个P标签对应的ID属性的值
print(soup.p['id'])
# 依次打印出所有A标签的href属性的值和A标签的值
for s in soup.find_all('a'):
    print("href={} text={}".format(s['href'],s.string))
```

发送第三方开户请求的脚本：

```
    def test01_trust_register(self):
        #先登录
        response = self.login_api.login(self.session)
        logging.info('login response = {}'.format(response.json()))
        assert_util(self,response,200,200,'登录成功')

        #发送开户请求
        response = self.trust_api.trust_register(self.session)
        logging.info('trust response = {}'.format(response.json()))
        self.assertEqual(200,response.status_code)
        self.assertEqual(200,response.json().get("status"))

        #提取开户请求的响应中的HTML内容
        html_data = response.json().get("description").get("form")
        #初始化Bs4的对象
        soup = BeautifulSoup(html_data,'html.parser')
        #提取第三方请求的URL
        third_request_url = soup.form['action']
        #提取第三方请求的参数名和参数值
        json_para = {}
        for params in soup.find_all('input'):
            json_para.setdefault(params.get("name"),params.get("value"))
        logging.info('params = {}'.format(json_para))
        #发送第三方请求
        response = self.session.post(third_request_url,data=json_para)
        logging.info('third request reposne = {}'.format(response.text))
        self.assertEqual(200,response.status_code)
        self.assertEqual('UserRegister OK',response.text)
```

团队测试工作进展

- 每个人能够说出自己的工作进度
- 每个人能够说出自己接下来的要做的工作
- 组长告知小组成员目前小组的整体工作进度（单独发送）
- 组长能够根据小组的整体工作进度及时调整工作安排