金融项目第七天--课堂笔记

昨日回顾

（1）接口测试自动化测试的场景

- 目的：保证软件版本的迭代质量，防止开发修改代码时引入新的问题
- 编写自动化脚本的时机：
    - 如果开发后端代码先转测，前端代码未转测，在接口测试手工执行结束后，就可以进行自动化脚本编写
    - 如果开发前后端代码同时转测，先进行系统测试用例执行，再进行接口自动化脚本的编写
    - 接口自动化脚本编写和系统系统执行在实际过程没有固定的先后顺序，系统测试执行的优先级更高。如果具备系统测试执行的条件，优先进行系统测试执行。
- 执行自动化脚本的时机：
    - 当自动化脚本编写完成后，会将脚本添加到持续集成中，定时、自动的一直运行下去，直到项目结束
- 接口自动化脚本的依据：
    - 参照接口文档进行编写
    - 参考实际环境的抓包进行编写

（2）将接口的Jmeter手工脚本转化为自动化脚本

- 请求参数化
- 响应断言
- 测试报告

（3）清除测试数据

- 原因：
    - 保证脚本可以多次重复执行
    - 防止脚本的数据对于其他脚本的运行产生影响

学习目标

- 能够运行Jmeter自动化脚本并生成测试报告
- 能够配置持续集成任务，自动运行Jmeter脚本
- 能够使用Python+Request编写接口自动化测试脚本

- 能够配置持续集成任务，自动运行接口自动化的代码脚本

测试数据的清理：

熟悉用户注册对应的数据库的表，理清表之前的依赖关系：

- mb_member：用户主表

- mb_member_info：用户信息表
- mb_member_login_log：用户登录日志表
- mb_member_register_log：用户注册日志表

整理清除数据所需要的sql语句：

1、需要先编写测试SQL，查询出所有的测试数据（确保SQL完全正确，不能造成误删）

```
#查询用户信息表中的所有测试数据
select i.* from mb_member_info i INNER JOIN mb_member m ON i.member_id = m.id
where m.phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
#查询用户的登录日志表中的所有测试数据
SELECT l.* from mb_member_login_log l INNER JOIN mb_member m ON l.member_id =
m.id where  m.phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
# 查询注册日志表中的所有测试数据
select * from mb_member_register_log where phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
# 查询出用户主表中的所有测试数据
SELECT * from mb_member  where phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
```

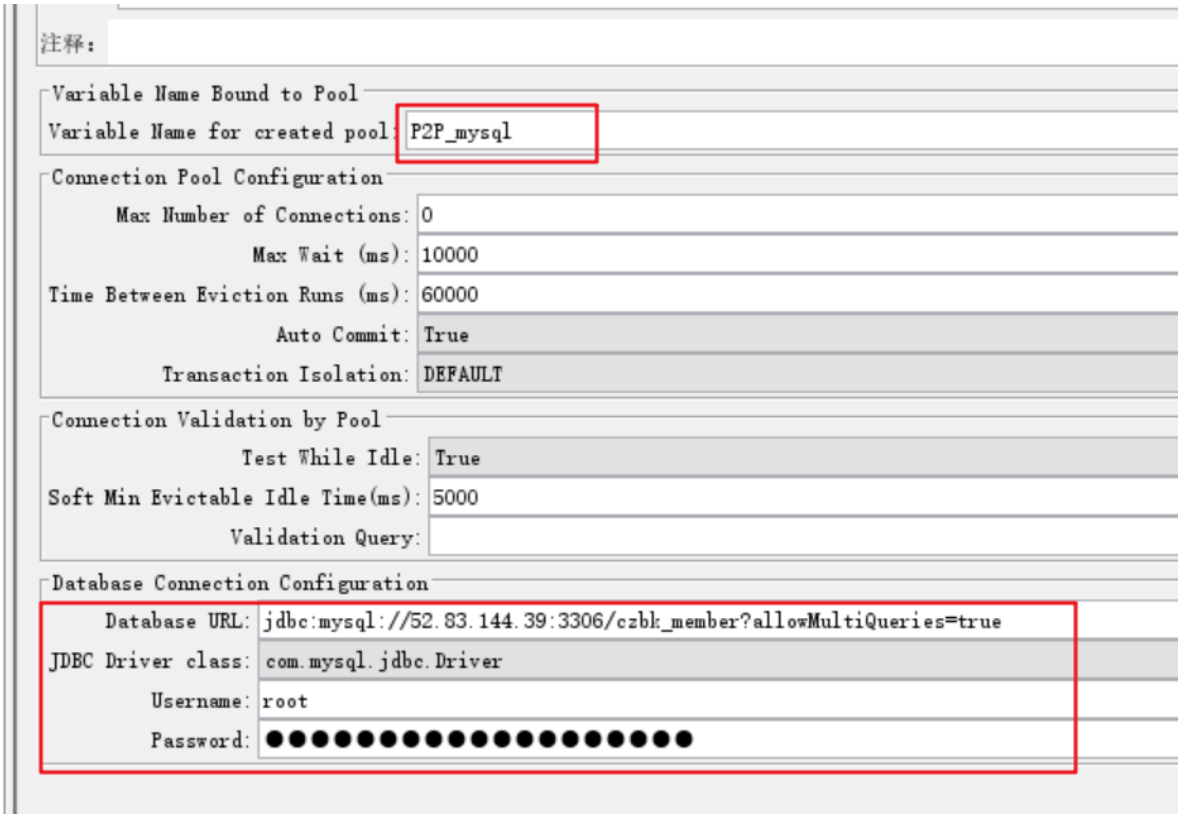
2、将上面的查询sql修改为删除的sql语句，执行删除操作（只修改前面的关键字即可）

```
#删除用户信息表中的所有测试数据
DELETE i.* from mb_member_info i INNER JOIN mb_member m ON i.member_id = m.id
where m.phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
#删除用户的登录日志表中的所有测试数据
DELETE l.* from mb_member_login_log l INNER JOIN mb_member m ON l.member_id =
m.id where  m.phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
# 删除注册日志表中的所有测试数据
DELETE from mb_member_register_log where phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
# 删除出用户主表中的所有测试数据
DELETE from mb_member  where phone in
('13011334522','13011334523','13011334524','13011334525','13011334526');
```

jmeter自动化执行清除数据

第一种方式（多个SQL语句同时执行）：

- 1、配置JDBC连接池
  - 配置连接池名称
  - 配置连接池相关的连接参数
    - allowMultiQueries=true：添加该参数后，在后面的JDBC Request请求中可以同时执行多个SQL语句

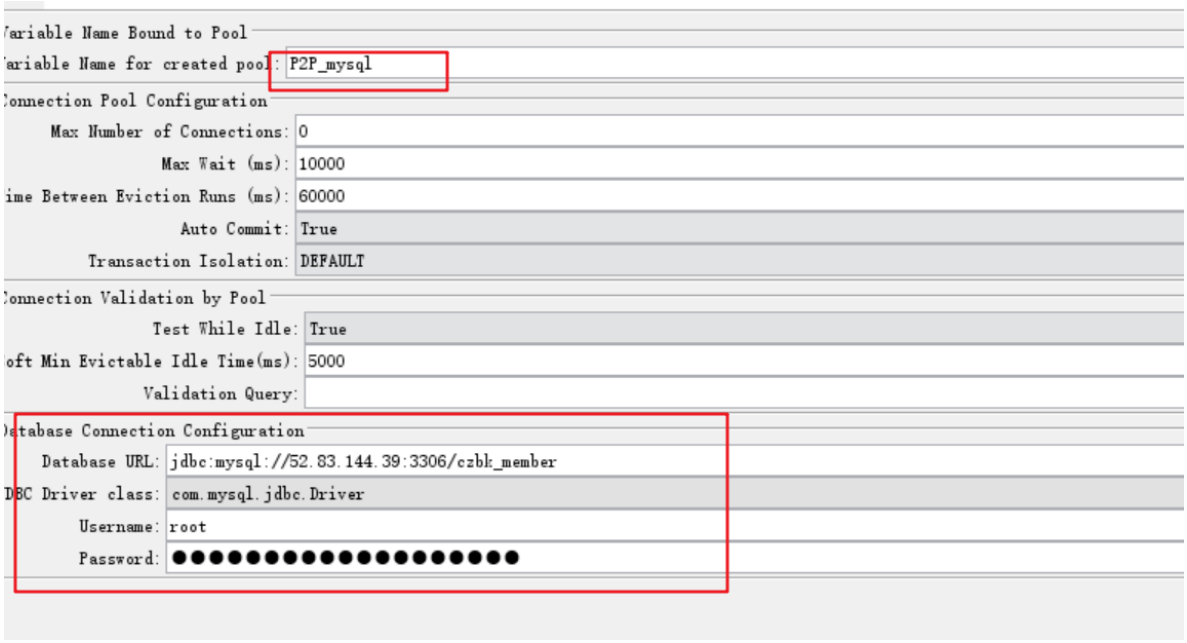


- 2、添加JDBC Request请求
  - 配置连接池的名称
  - 添加需要执行的多条SQL语句
    - SQL语句想批量执行：必须在JDBC连接池配置中的URL参数中加入：allowMultiQueries=true
    - 删除的测试数据（手机号）需要引用用户定义的变量（不要写死）

第二种方式（每个SQL语句单独执行）：

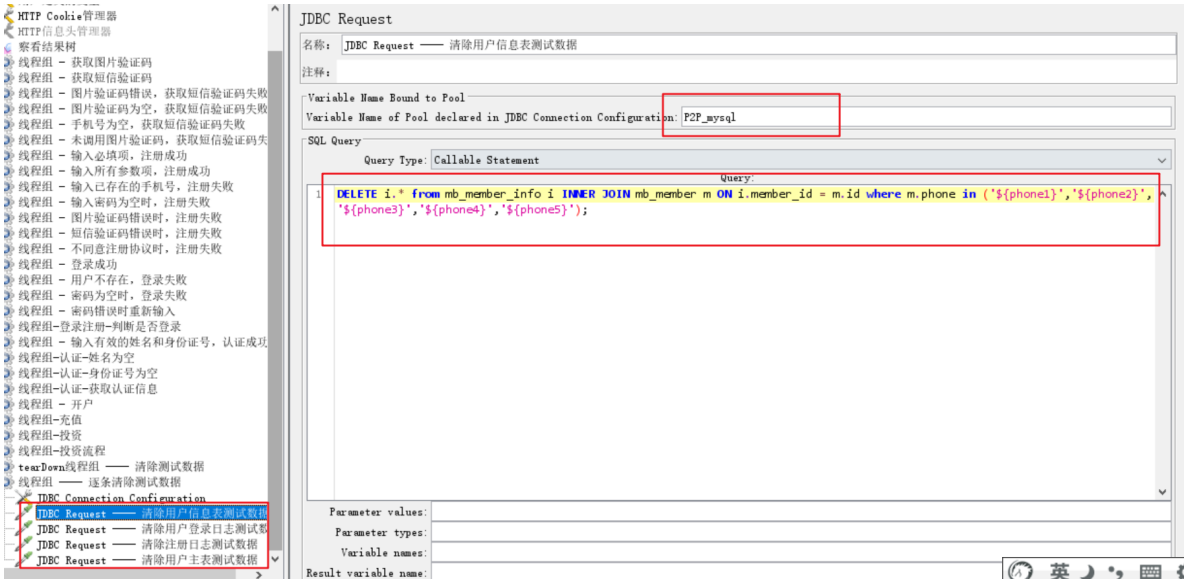
1、配置JDBC连接池

- 配置连接池名称
- 配置连接池相关的连接参数



2、添加多个JDBC Request请求（有多少个删除的SQL语句就配置多少个JDBC Request）

- 配置连接池名称
- 每个请求中添加一个SQL语句
  - 删除的测试数据（手机号）需要引用用户定义的变量（不要写死）

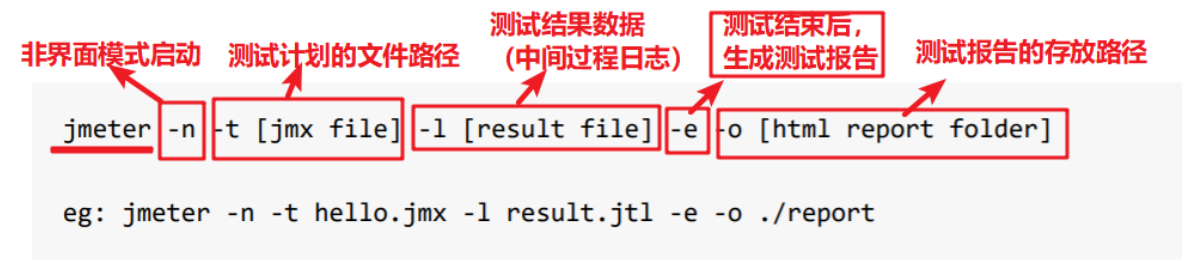


自动化脚本的最终运行：

1、在测试计划中将"独立运行每个线程组"勾选上，保证所有Jmeter脚本顺序执行

2、可以将清除测试数据的脚本，放入到teardown线程组中，保证该脚本始终在最后执行

Jmeter执行并生成测试报告




```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.18362.476]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\tools\apache-jmeter-5.0\bin>
C:\tools\apache-jmeter-5.0\bin>jmeter -n -t "D:\传智播客\02 授课\北京22期-金融项目\day7\jmeter\P2P_autotest.jmx" -l resu
lt.jtl -e -o ./report
Creating summariser <summary>
Created the tree successfully using D:\传智播客\02 授课\北京22期-金融项目\day7\jmeter\P2P_autotest.jmx
Starting the test @ Wed Aug 26 10:46:00 CST 2020 (1598409960820)
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary +      1 in 00:00:00 =    2.9/s Avg:   162 Min:   162 Max:   162 Err:     0 (0.00%) Active: 1 Started: 1 Finishe
d: 0
summary +     41 in 00:01:03 =    0.7/s Avg:    67 Min:    40 Max:   102 Err:     2 (4.88%) Active: 1 Started: 17 Finish
ed: 16
summary =     42 in 00:01:03 =    0.7/s Avg:    69 Min:    40 Max:   162 Err:     2 (4.76%)
summary +     38 in 00:00:04 =    9.4/s Avg:   103 Min:     2 Max:  1075 Err:     2 (5.26%) Active: 0 Started: 1 Finishe
d: 1
summary =     80 in 00:01:07 =    1.2/s Avg:    85 Min:     2 Max:  1075 Err:     4 (5.00%)
Tidying up ...    @ Wed Aug 26 10:47:08 CST 2020 (1598410028831)
... end of run
C:\tools\apache-jmeter-5.0\bin>
```

查看report报告：

1、通过error来检查哪些脚本执行失败

Statistics

Requests	Executions			Response Times (ms)						
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput
Total	80	4	5.00%	85.55	2	1075	135.30	160.95	1075.00	1.19
Debug Sampler	1	0	0.00%	2.00	2	2	2.00	2.00	2.00	500.00
HTTP请求 - 不同意注册协议时，注册失败	1	1	100.00%	60.00	60	60	60.00	60.00	60.00	16.67
HTTP请求 - 参数正确，获取短信验证码失败	1	0	0.00%	84.00	84	84	84.00	84.00	84.00	11.90
HTTP请求 - 参数正确，获取短信验证码成功	8	0	0.00%	55.00	51	63	63.00	63.00	63.00	3.98
HTTP请	1	0	0.00%	53.00	53	53	53.00	53.00	53.00	18.87

2、查看每个接口失败的断言结果

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	**80**	**4**	**Value expected to match regexp '100', but it did not match: '200'**	**2**	**Assertion failed**	**2**	**null**	**null**	**null**	**null**
HTTP请求 - 不同意注册协议时，注册失败	1	1	Value expected to match regexp '100', but it did not match: '200'	1	null	null	null	null	null	null
HTTP请求 - 输入密码为空时，注册失败	1	1	Value expected to match regexp '100', but it did not match:	1	null	null	null	null	null	null

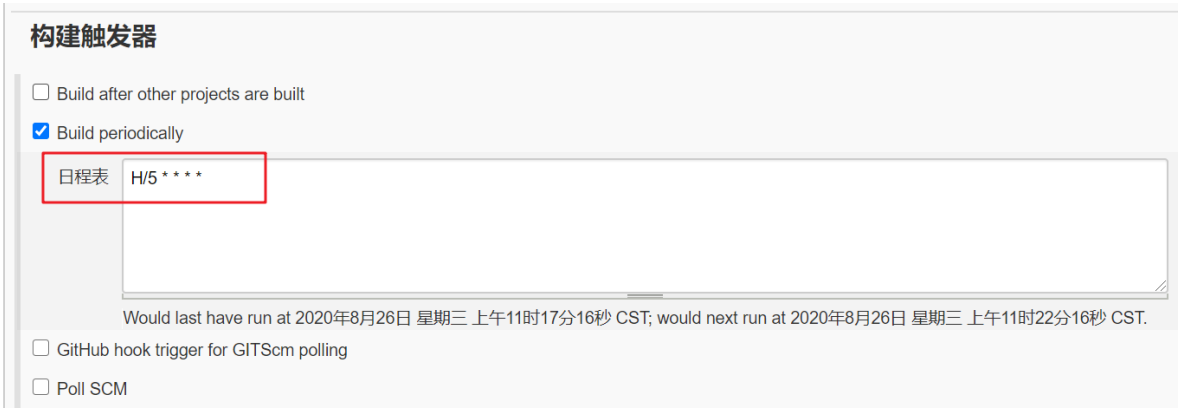
Jenkis配置：

配置步骤：

1、添加Jenkins定时项目

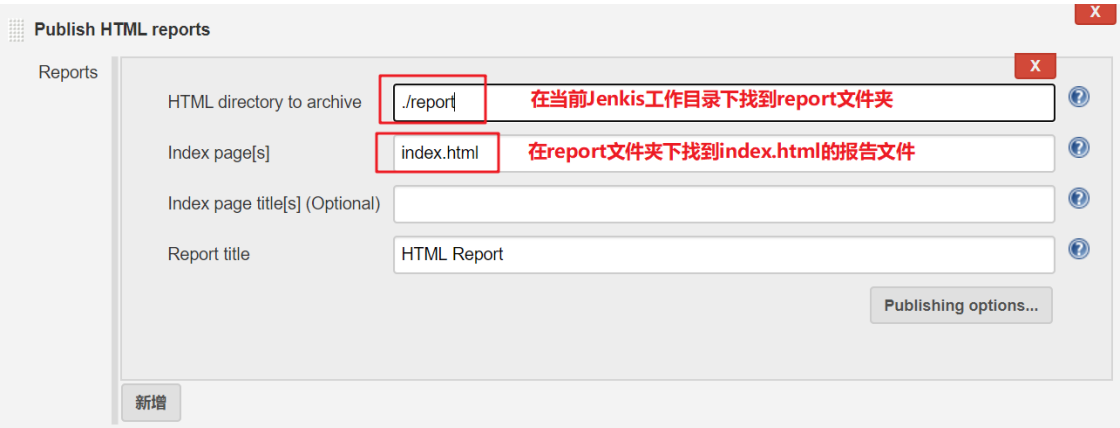
2、配置Jenkins项目

- 构建触发器 （多久运行一次该项目脚本）

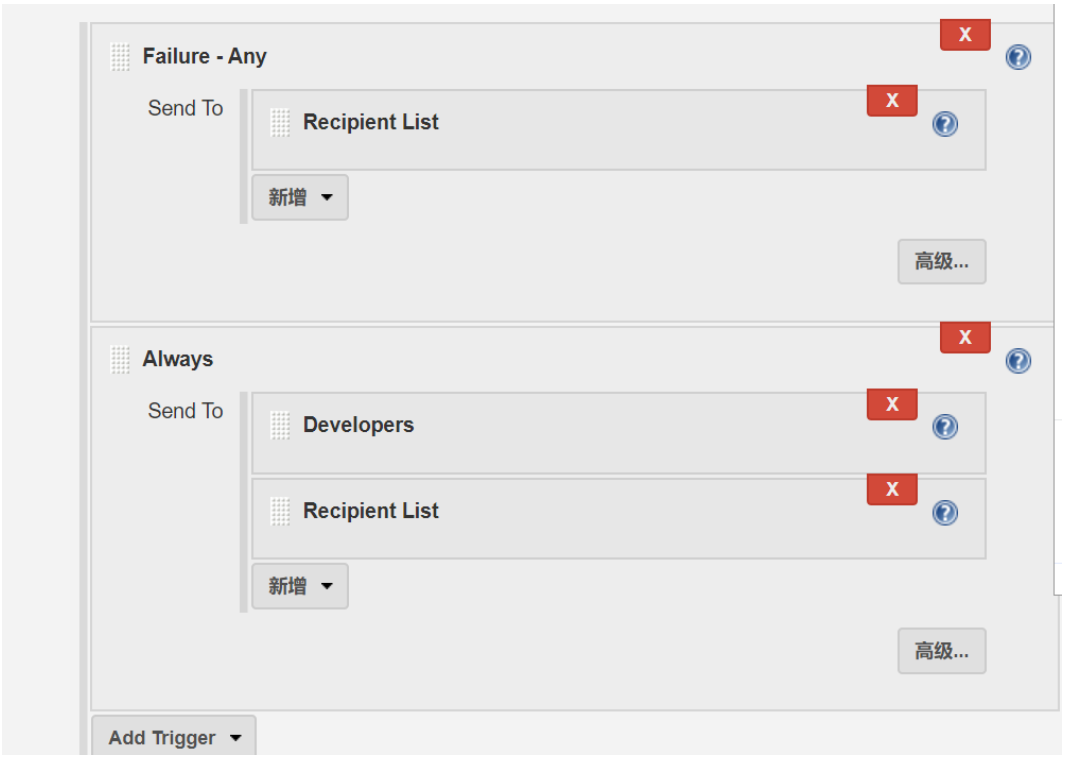


- 构建中命令
    - Execute Windows batch command

- 构建后操作
  - Publish HTML reports： 构造后的日志结果路径



  - Editable Email Notification：构建后的日志结果发送的邮件收件人和对应的邮件格式
    - 选择右下角Advanced Settings，添加发送邮件的时机和对象



第一次执行时成功的日志打印：

控制台输出

```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p
No emails were triggered.
[bj22_jmeter_p2p] $ cmd /c call C:\Users\ThinkPad\AppData\Local\Temp\jenkins9064707759100925693.bat
```

1、进入jenkins任务的工作目录下，准备启动任务

```
C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p>jmeter -n -t "D:\传智播客\02 授课\北京22期-金融项目\day7\jmeter\P2P_autotest.jmx"
-l result.jtl -e -o ./report
Creating summariser <summary>
Created the tree successfully using D:\浼狗櫛鎰□□\02 鎚堝□\鍖櫛含22鏈€-閱戠瀹椤圭洰\day7\jmeter\P2P_autotest.jmx
Starting the test @ Wed Aug 26 11:29:48 CST 2020 (1598412588638)
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary =     80 in 00:01:08 =    1.2/s Avg:    93 Min:     2 Max:  1218 Err:     4 (5.00%)
Tidying up ...    @ Wed Aug 26 11:30:57 CST 2020 (1598412657598)
... end of run
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
```

2、执行配置中构造部分的命令行，并打印执行的过程

```
[htmlpublisher] Archiving HTML reports...
[htmlpublisher] Archiving at PROJECT level C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p to
C:\Users\ThinkPad\.jenkins\jobs\bj22_jmeter_p2p\htmlreports\HTML_20Report
No emails were triggered.
Finished: SUCCESS
```

3、按照配置中构造后的日志路径配置，拷贝日志文件并邮件发送

日志路径

4、结果：成功

第二次执行失败日志打印：

```
Started by timer
Running as SYSTEM
Building in workspace C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p
No emails were triggered.
[bj22_jmeter_p2p] $ cmd /c call C:\Users\ThinkPad\AppData\Local\Temp\jenkins8843214018926159252.bat
```

1、准备启动执行

```
C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p>jmeter -n -t "D:\传智播客\02 授课\北京22期-金融项目\day7\jmeter\P2P_autotest.jmx"
-l result.jtl -e -o ./report
An error occurred: Cannot write to 'C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p\.\report' as folder is not empty
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
errorlevel=1
请按任意键继续. . .
Build step 'Execute Windows batch command' marked build as failure
```

report目录必须为空（不存在），才能通过命令行方式来执行

2、执行过程

```
[htmlpublisher] Archiving HTML reports...
[htmlpublisher] Archiving at PROJECT level C:\Users\ThinkPad\.jenkins\workspace\bj22_jmeter_p2p\report to
C:\Users\ThinkPad\.jenkins\jobs\bj22_jmeter_p2p\htmlreports\HTML_20Report
Email was triggered for: Failure - Any
Sending email for trigger: Failure - Any
An attempt to send an e-mail to empty list of recipients, ignored.
Finished: FAILURE
```

3、拷贝报告并发送

4、结果：失败

代码方式完成接口自动化：

创建接口自动化代码目录结构

```
apiTestP2P
├── api          → 定义的接口
├── script       → 所有的测试脚本
├── data         → 测试数据（参数化数据）
├── report       → 测试报告
├── lib          → 第三方库
├── log          → 日志文件
├── app.py       → 全局变量
├── utils.py     → 定义的工具类
└── run_suite.py → 测试套件，执行测试的主程序
```

安装依赖包

- 安装 `requests` 包 → 发请求
- 安装 `parameterized` 包 → 参数化
- 安装 `pymysql` 包 → 连接数据库
- 添加 `HTMLTestRunner` → 测试报告

安装方法：

- requests、parameterized、pymysql包安装：pip install 包名
- HTMLTestRunner包安装：将下载好的文件，拷贝到项目的lib目录下

初始化日志脚本：

```
# app.py
# 初始化日志配置
def init_log_config():
    # 创建日志器
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)
    # 创建控制台处理器
    sh = logging.StreamHandler()
    # 创建文件处理器
    log_path = BASE_DIR + "/log/p2p.log"
    fh = logging.handlers.TimedRotatingFileHandler(log_path, when="midnight", interval=1
,
    backupCount=7, encoding="UTF-8")
    # 创建格式化器
    f='%(asctime)s %(levelname)s [%(name)s] [%(filename)s(%(funcName)s:%(lineno)d)] - %
(message)s'
    formatter = logging.Formatter(f)
    # 把格式化器添加到处理器中
    sh.setFormatter(formatter)
    fh.setFormatter(formatter)
    # 把处理器添加到日志器中
    logger.addHandler(sh)
    logger.addHandler(fh)
```

初始化日志器对象

设置日志级别：表示比INFO级别高（包括info）的日志信息才记录

负责将日志信息输入到Pycharm下方的console窗口中

负责将日志信息输入到log目录下的日志文件中

在记录日志时的详情格式

把格式化器绑定到控制台和文件处理器中

app.py：

```
#初始化日志配置
def init_logger():
    #1、初始化日志对象
    logger = logging.getLogger()
    #2、设置日志级别
    logger.setLevel(logging.INFO)
    #3、创建控制台日志处理器和文件日志处理器
    sh = logging.StreamHandler()
    filename = BASE_DIR + "/log/p2p.log"
    fh = logging.handlers.TimedRotatingFileHandler(filename, when='M',
interval=3, backupCount=5, encoding='utf-8')
    #4、设置日志格式，创建格式化器
```

```
    fmt = '%(asctime)s %(levelname)s [%(name)s] [%(filename)s(%(funcName)s:%
(lineno)d)] - %(message)s'
    formatter = logging.Formatter(fmt)
    #5、将格式化器设置到日志器中
    sh.setFormatter(formatter)
    fh.setFormatter(formatter)
    #6、将日志处理器添加到日志对象
    logger.addHandler(sh)
    logger.addHandler(fh)
```

api/__init__.py

```
from app import init_log_config
import logging

init_log_config()
logging.info("info")
logging.error("error")
logging.debug("debug")
```

编写测试脚本：

- 方法1：先编写测试脚本，再从脚本中抽象出API接口类
    - 所有测试脚本中相同的那部分代码，可以抽象出来
- 方法2：先定义接口类，再编写测试脚本

1、定义接口

- api文件夹下先创建一个loginAPI的接口文件

```
import app
import requests

class loginAPI():
    def __init__(self):
        self.getImgCode_url = app.BASE_URL +'/common/public/verifycode1/'

    def getImgCode(self,session,r):
        url = self.getImgCode_url + r
        response = session.get(url)
        return response
```

2、编写测试脚本

- 新增注册登录脚本login.py，

```
import unittest
import random

import requests

from api.loginAPI import loginAPI
```

```
class login(unittest.TestCase):
    def setUp(self) -> None:
        self.login_api = loginAPI()
        self.session = requests.Session()

    def tearDown(self) -> None:
        self.session.close()


    def test01_get_img_code_random_float(self):
        #定义参数(随机小数)
        r = random.random()
        #调用接口类中的接口
        response = self.login_api.getImgCode(self.session,str(r))
        #接收接口的返回结果，进行断言
        self.assertEqual(200,response.status_code)
```

生成测试报告：

```
import unittest
import app,time
from lib.HTMLTestRunner_PY3 import HTMLTestRunner

from script.login import login

suite = unittest.TestSuite()
suite.addTest(unittest.makeSuite(login))

report_file = app.BASE_DIR +
"/report/report{}.html".format(time.strftime("%Y%m%d-%H%M%S"))
with open(report_file,'wb') as f:
    runner = HTMLTestRunner(f,title="P2P金融项目接口测试报告",description="test")
    runner.run(suite)
```

连接数据库封装基础类：

```
class DButils:
    @classmethod
    def get_conn(cls):
        conn =
pymysql.connect(app.DB_host,app.DB_user,app.DB_password,app.database_mem,autocom
mit=True)
        return conn

    @classmethod
    def close_conn(cls,cursor = None,conn = None):
        if cursor:
            cursor.close()
        if conn:
            conn.close()

    @classmethod
    def execute_sql(cls,sql):
```

```
        try:
            conn = cls.get_conn()
            cursor = conn.cursor()
            cursor.execute(sql)
        except Exception as e:
            conn.rollback()
        finally:
            cls.close_conn()
```

删除对应测试数据：

```
    @classmethod
    def tearDownClass(cls) -> None:
        sql1 = "delete from mb_member_register_log where phone in
('13099881111','13099881112','13099881113','13099881114');";
        DButils.execute_sql(sql1)
        logging.info('delete sql1 = {}'.format(sql1))
        sql2 = "delete i.* from mb_member_login_log i INNER JOIN mb_member m on
i.member_id = m.id WHERE m.phone in
('13099881111','13099881112','13099881113','13099881114');";
        DButils.execute_sql(sql2)
        logging.info('delete sql1 = {}'.format(sql2))
        sql3 = "delete i.* from mb_member_info i INNER JOIN mb_member m on
i.member_id = m.id WHERE m.phone in
('13099881111','13099881112','13099881113','13099881114');";
        DButils.execute_sql(sql3)
        logging.info('delete sql1 = {}'.format(sql3))
        sql4 = "delete from mb_member WHERE phone in
('13099881111','13099881112','13099881113','13099881114');";
        DButils.execute_sql(sql4)
        logging.info('delete sql1 = {}'.format(sql4))
```

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