金融项目第六天--课堂笔记

昨日回顾

（1）接口（自动化）过程中测试数据的构造

- 手工构造数据
  - 优点：简单
  - 缺点：不能用于频繁构造测试数据
- 接口请求方式构造数据
  - 优点：可以用脚本方式重复构造测试数据；比数据库方式相对简单
  - 缺点：API接口之前的依赖比较强，其中任何一个接口出问题，可能会导致数据构造失败
- 数据库方式构造数据
  - 优点：灵活、效率高
  - 缺点：难度高（一条测试数据涉及到多张表时）

自动化测试过程中选择测试数据的构造方式：

- 手工方式：数据不需要频繁构造时
- 接口方式：数据需要频繁构造时，复杂度高
- 数据库方式：数据需要频繁构造，但复杂度不高（一条测试数据涉及到表不超过两张）

（2）接口用例手工执行

- 工具：
  - Jmeter（手工 + 自动化测试） —— 70%
  - Postman（手工） + Python Request（自动化） —— 30%
- Jmeter编写接口测试脚本
  - 关键组件：
    - HTTP请求取样器
    - HTTP消息头管理器
    - HTTP请求默认值
    - JSON/正则表达式提取器
    - JSON/响应断言
    - 查看结果树
  - 手工执行的过程：
    - HTTP请求默认值 —— 设置URL里的IP+port等，方便后续修改
    - HTTP消息头管理器 —— 对所有的请求消息头进行统一设置
    - HTTP Cookie管理器 —— 同一个业务操作中多个接口请求可以通过cookie管理来自动管理令牌
    - 线程组 —— 一个线程组可以对应一个用例；（多个用例组合成一个完整的业务过程，可以将多个用例放在同一个线程组）
    - 当前HTTP请求中有多消息体时，需要在HTTP请求取样器勾选“multi-part”设置
    - 在开户等第三方业务时，需要提取返回中“HTML代码”里部分字段，通过正则表达式提取器进行数据提取，并赋值后后续请求

学习目标

- 能够使用Jmeter对接口测试脚本进行自动化调试
- 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

接口自动化脚本应用场景：

- 测试目的：防止开发引入新的问题；保障项目迭代的质量
- 编写自动化脚本的时机：
    - 开发完成后端代码转测，前端代码未转测，接口用例手工执行已经结束，就可以进行接口自动化脚本的编写
    - 开发完成前后端代码的同时转测，需要先进行测试系统用例的手工执行，然后再进行接口自动化脚本的编写
    - 接口自动化脚本的编写 与 系统测试用例的手工执行时间 在实际工作中并没有固定的先后顺序，只是系统测试用例的手工执行优先级更高，但可以进行系统测试执行时，优先进行。
- 执行接口自动化脚本的时机：
    - 在接口自动化脚本调试通过后，就立即添加到持续集成环境中，定时、自动的运行这些自动化脚本，一直运行下去，直到项目结束
- 测试依据
    - 接口自动化测试的脚本，可以根据接口文档来进行编写
    - 接口自动化测试的脚本，也可以根据环境的抓包来进行编写（接口手工测试的脚本不可能根据抓包来编写）

接口自动化常用的工具：

- 基于工具自动化：Jmeter（要求每个同学都掌握）
- 基本代码自动化：Python + Request（可选）

接口自动化脚本的调优：

- 请求参数化
- 响应断言
- 输出整体测试报告

接口自动化脚本的调试：

1、获取图片验证码：

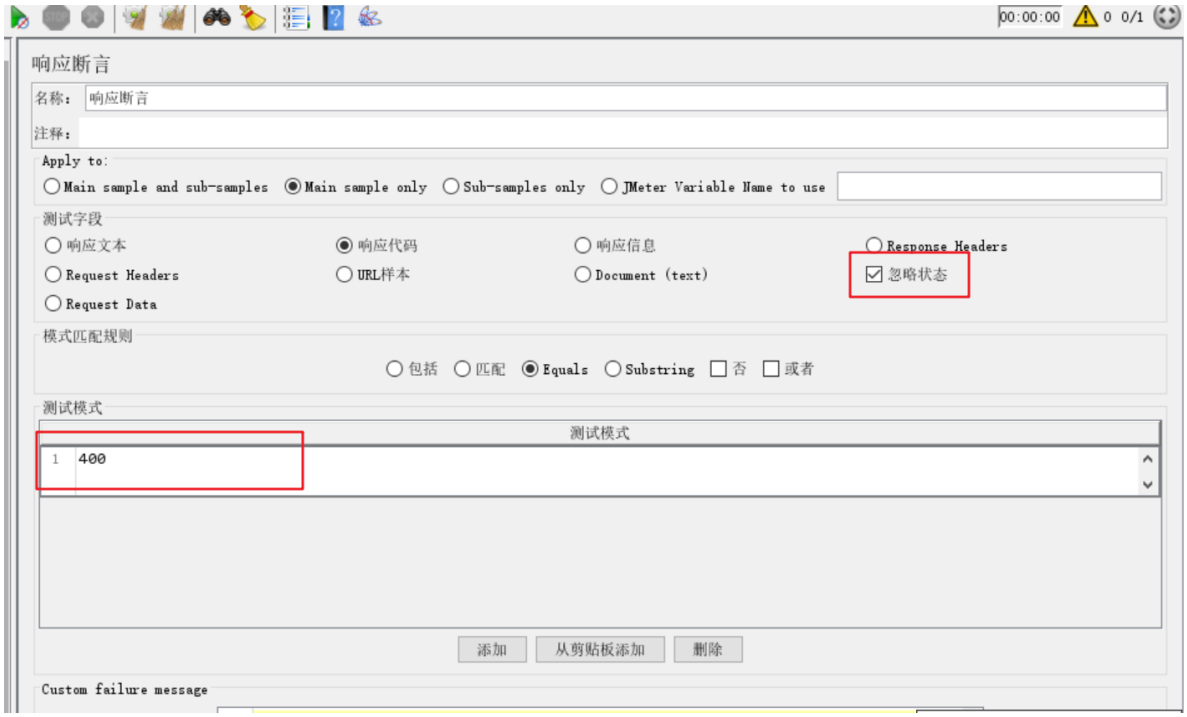
成功获取的脚本：

- 随机整数参数化：${__Random(10000000,99999999,)}
    - 这个函数只能生成随机整数，不能生成随机小数
- 断言：响应断言 —— 响应码：200

失败获取的脚本：

- 随机字母参数化：${__RandomString(10,abcdefghijiklmnopq,)}
    - 10表示随机字母的个数，”abcdefghijiklmnopq“从这段字母中随机挑选

- 断言：响应断言 —— 响应非200，需要勾选上响应断言中的忽略状态



2、获取短信验证码

成功获取的脚本：

- 参数：使用用户定义的变量来定义 手机号和图片验证码
- 断言：
  - 响应断言：200
  - Json断言：{"status":200,"description":"短信发送成功"}

失败获取的脚本：

- 参数：
  - 使用用户定义的变量来定义 手机号和图片验证码，
  - 失败/错误的测试数据，由于只用一次，可以提前定义变量，也可以不定义
- 断言：
  - 响应断言：200
  - json断言：{"status":100,"description":"图片验证码错误"}

3、注册：

成功注册的脚本：

- 参数化：使用用户定义的变量来定义 手机号和图片验证码、密码、短信验证码
- 断言：
  - 响应断言：200
  - json断言：{"status":200,"description":"注册成功"}

输入已存在的手机号，失败注册的脚本：

- 参数化：
  - 图片验证码、密码、短信验证码都是使用用户定义的变量
  - 已存在的手机号：
    - 去数据库里查已存在的手机号，使用查询的手机号进行测试 —— 可以不用控制，所有脚本可以并行执行

- 使用前面注册成功的脚本中使用的手机号进行测试。 —— 必须控制脚本按照指定的顺序执行

```
第一种方式（查数据库）：
  方式：
  - 不能通过手工方式查询一次数据库中的已存在手机号，写死在脚本中，因为在公司的测试环境是由多人使用，账号信息可能会被删除。
  - 需要在脚本中添加JDBC Request，查询出当前存在的手机号，存为变量；在后续重复注册的请求中，引用该变量，构造用户重复注册的请求
  优点： 当前脚本与之前脚本没有依赖关系，可以不用控制顺序
  缺点： 第一种方式，脚本编写更复杂（如上：先查询数据库，提取结果再发请求）

第二种方式（使用之前注册成功的脚本中的手机号） -- 类似接口构造测试数据
  优点： 快，直接使用之前注册成功的手机号发请求即可
  缺点： 依赖之前的脚本，如果之前脚本失败/执行顺序出问题，都会导致脚本失败
```

- 断言：
  - 响应断言：200
  - json断言：{"status":100,"description":"手机已存在!"}

注册失败的其他脚本：

- 参数化：
  - 图片验证码、密码、短信验证码、手机号都是使用用户定义的变量
  - 在测试过程由于出现bug，导致手机号被注册成功（预期不成功），为了不影响后续的脚本执行，需要使用新的手机号
- 断言：
  - 响应断言：200
  - json断言：status":100,"description":每种错误对应的错误描述

```
- 注意： 测试脚本如果在调试过程中出现bug，确定是代码bug，保持脚本中的断言为用例中的预期结果；不要为了让脚本通过将断言中的结果改为实际的响应。
```

4、登录脚本：

- 参数化：
  - 登录的账号就使用之前已经注册成功的手机号
  - 密码使用用户定义的变量
- 断言：
  - 响应断言：200
  - json断言：
    - 成功：status": 200,"description":"登录成功"
    - 失败："status": 100,"description":每种错误对应的错误描述

5、认证脚本：

认证成功：

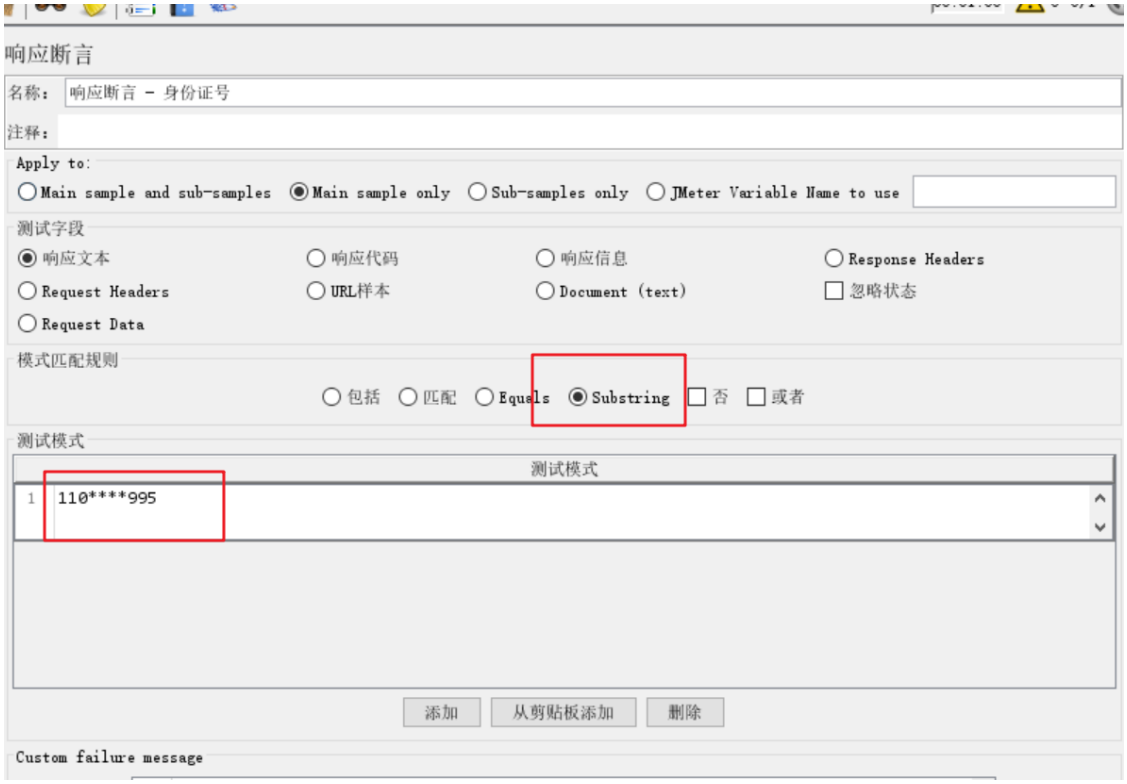
- 参数化：登录的账号就使用之前已经注册成功的手机号
- 断言：
  - 响应断言：200
  - json断言：{"status":200,"data":{"card_id":"xxx","realname":"李**"},"description":"提交成功!"}

认证失败：

- 参数化：登录的账号就使用之前已经注册成功的手机号（不能使用前面认证成功的手机号，否则认证失败无法确定原因）
- 断言：
  - 响应断言：200
  - JSON：{"status":100,"description":每种错误对应的错误描述}

获取认证信息：

- 参数化：登录的账号就使用之前已经认证成功的手机号
- 断言：
  - 响应断言：200
  - 对于结果：如果要校验 身份证号 或者 姓名时，需要使用响应断言。



6、第三方接口 —— 开户、充值、投资

参数化：

- 用于投资的账户需要提前准备（投资需要注册账号、认证、风险评估）
- 用于投资的标的需要提前准备（每个人测试时使用自己准备的测试数据）
  - 设置投资标的为200000，一次投资100，可以运行2000次，因此这个投资标的的测试数据可以通过手工方式来创建
- 其他参数，都可以定义在用户定义的变量

断言：

- 响应码：200
- 其他内容：
  - 按照用例来描述来断言：status和description
  - 对于第三方接口的断言：在mock代码中return的字符串

7、业务流程的接口的自动化

- 只需要在前面的单个接口自动化调试时，将脚本调通（完成参数化和断言）
- 在业务流程自动化时，每个接口完全使用前面的参数化和断言的方法即可。

清理测试数据：

为什么要清理测试数据：

- 还原到测试的环境
- 实现测试用例的多次执行
- 减少对其他脚本的影响

清除测试数据的方式：

- 一定是通过自动化的方式来清理
    - 调用删除接口进行数据删除（取决于开发有没有提供删除接口）
    - 直接从数据库中删除对应的数据
        - 要求测试人员对于测试数据涉及到的数据库表结构非常熟悉

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：

```
delete i.* from mb_member_info i INNER JOIN mb_member m ON i.member_id = m.id
where m.phone in ('13099981111','13099981112','13099981113','13099981114')

delete l.* from mb_member_login_log l INNER JOIN mb_member m ON l.member_id =
m.id where m.phone in ('13099981111','13099981112','13099981113','13099981114')

delete from mb_member where phone in
('13099981111','13099981112','13099981113','13099981114')

delete from mb_member_register_log where phone in
('13099981111','13099981112','13099981113','13099981114')
```

jmeter自动化执行清除数据

-

Jenkis配置：

- 添加jenkis任务，
    - 配置定时任务，拷贝Jmeter代码到工作目录下（C:\Users\ThinkPad.jenkins\workspace），定时执行
    - Jmeter报告必须要在report文件不存在，因此需要提前删除该报文文件

```
del result.jtl
rmdir report /S /Q
```

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