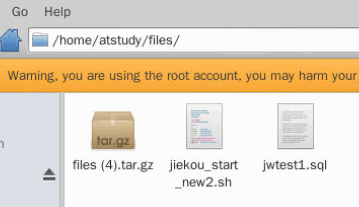
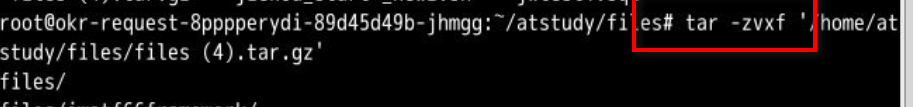
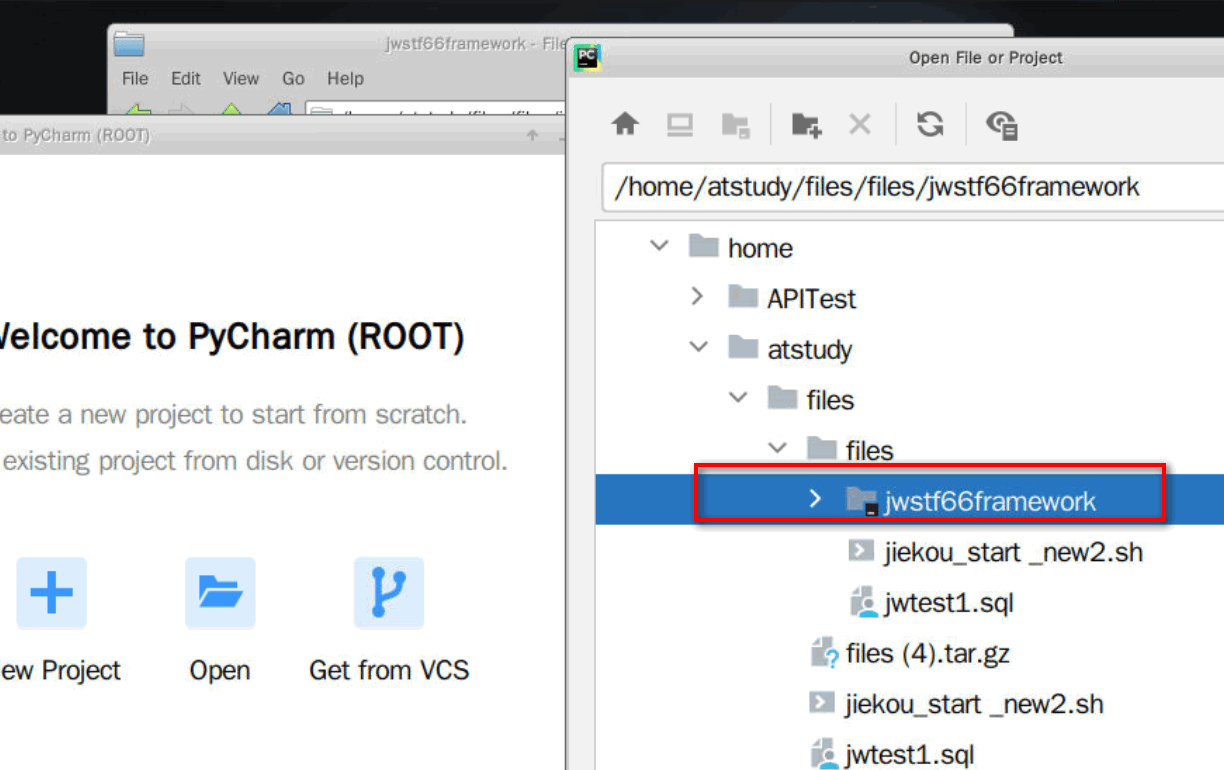
全程66期接口自动化第七次

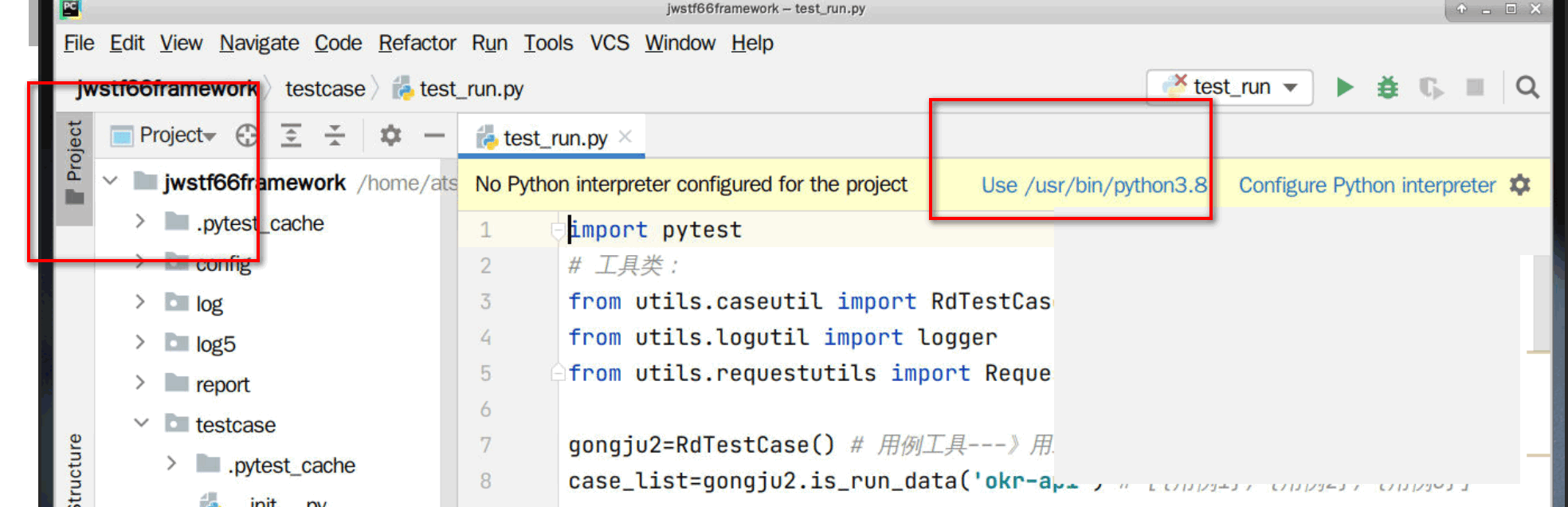
回顾：

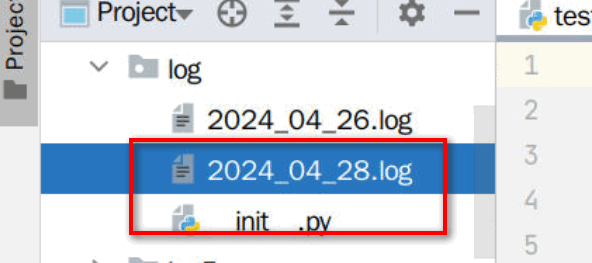
1 环境： -----









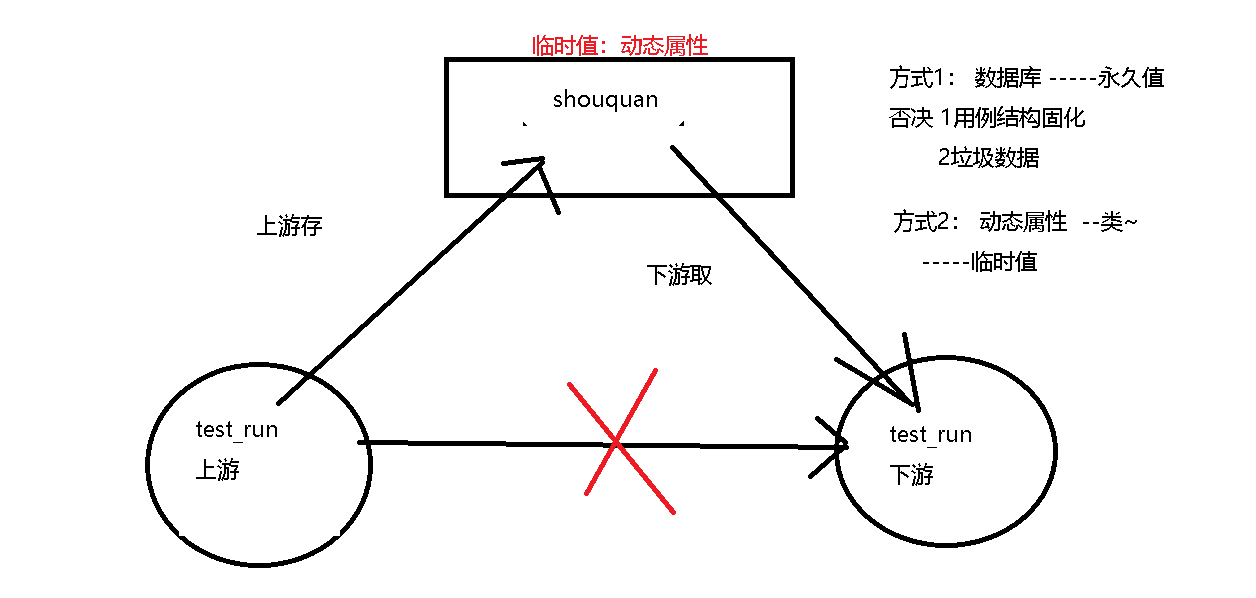


### 内容1： 上下游接口：

回顾：

1获取 上游接口关联值----------------变量（1requests 2 postman 3jmeter）------》下游引用~

方案失败：



**例子1： 动态属性语法：**

|  |
| --- |
| # 需求：  # 对象---专有属性～  # 冠希----》打工-----》gongzi工资属性  # 阿娇 ---》上学 ----》daxue学历属性  # 解决：  # 1setattr（类对象，“属性名字”,“属性值”） 设置属性～  # 2 getattr（类对象，“属性名字”，“默认值”） 获取属性～  # 属性  from time import sleep  class Person:  def \_\_init\_\_(self,name,age):  self.name=name # 普通属性～ 所有对象都有  self.age=age  p1=Person("冠希",28)  p2=Person("阿娇",18)  if \_\_name\_\_ == '\_\_main\_\_':  print(p1.name,p1.age)  print(p2.name,p2.age)  print("随着系统运行～大千世界不断变化～～～")  sleep(3)  setattr(p1,"gongzi",20000)  # print(p1.name,p1.age,p1.gongzi) # 对象独有～  # print(p2.name, p2.age, p2.gongzi) # 获取  print(p1.name,p1.age,getattr(p1,"gongzi","无工资"))  print(p2.name, p2.age, getattr(p2, "gongzi", "无工资")) # 降低风险～～  print("嘿嘿～ 有钱了~") |

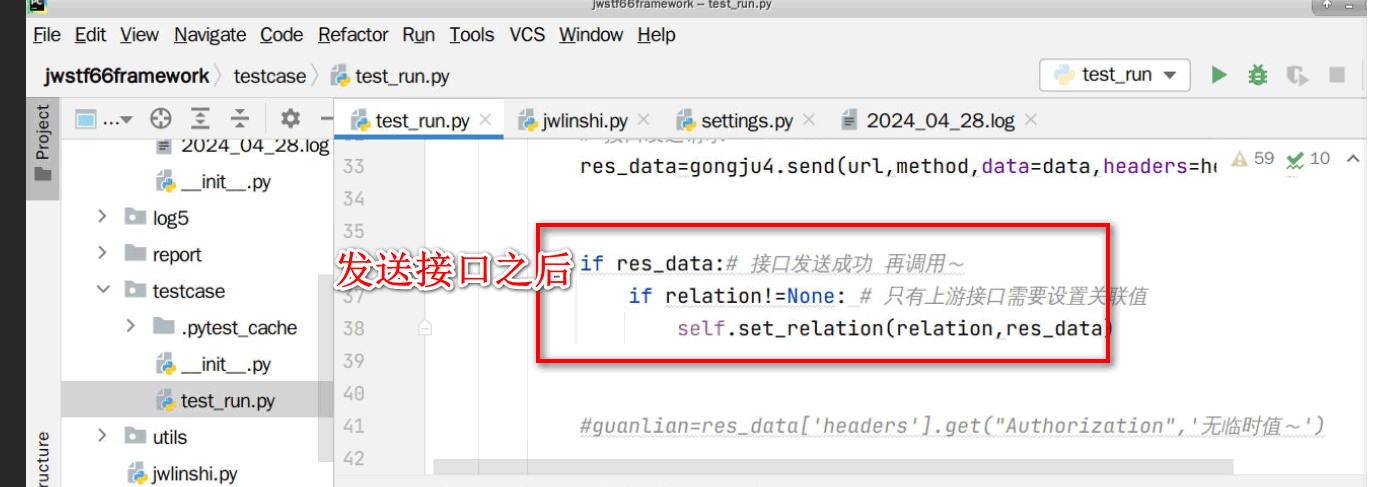
**例子2： 应用动态属性（完成临时值存储使用~ 方案：----上下游接口处理逻辑）**

|  |
| --- |
| **思路：**  **1 准备文件 类 等待存放 容易变化内容（临时）**  **descript**  **2 （函数1） 判断上游用例 根据relation 值 将需要存放值 提取 存放setattr到 文件类**    **代码实现思路：**  **1def set\_relation(形参relation，实际结果res\_data):**  **1.1判断relation 是否有值=非空=》上游用例**  **1.2 切割 relation**  **1.2.1 =左边 ====》动态属性名字**  **1.2.2 根据=右边到实际结果res\_data中查找===》动态属性值**  **1.3 setattr(类，动态属性名字，动态属性值)==》存放结束**  **3 （函数2） 判断下游用例 根据 ${} 值 到 文件类 进行提取getattr，替换${} 交给requests工具类 发出接口请求**  **3.1 def correlation(data):**  **3.1.1判断data 是否有${}====》字符串正则查找**  **3.1.2 用getattr（Dynamic，${}的值，“默认值”）==》关联值**  **3.1.3 关联值 字符串替换${}===>Template**  **3.1.4 替换完值字符串转成字典方便给requests使用 ====>json.loads** |

**上游接口的定义函数：**

|  |
| --- |
| **# 1def set\_relation(形参relation，实际结果res\_data):**  **def set\_relation(self,relation,res\_data):**  **# 健壮性 1验证 2 try方便定位报错代码～**  **try:**  **# 1.1判断relation 是否有值=非空=》上游用例**  **if relation:**  **# 1.2 切割 relation**  **jwlist=relation.split("=") #[jwtoken,Authorization]**  **# 1.2.1 =左边 ====》动态属性名字**  **zuobian=jwlist[0]**  **# 1.2.2 根据=右边到实际结果res\_data中查找===》动态属性值**  **youbian=jwlist[1]**  **guanlian=res\_data.get("headers").get(youbian,"无关联值")**  **# 1.3 setattr(类，动态属性名字，动态属性值)==》存放结束**  **setattr(DynamicParam,zuobian,guanlian)**  **#验证：**  **print("志玲验证是否存放成功:",getattr(DynamicParam,zuobian,"存放临时值失败～"))**  **except:**  **print("47~57行写的可能有问题～～")** |

**上游接口调用部分：**

****

**目前 核心代码文件：**

|  |
| --- |
| import pytest  # 工具类：  from config.settings import DynamicParam  from utils.caseutil import RdTestCase  from utils.logutil import logger  from utils.requestutils import RequestSend  gongju2=RdTestCase() # 用例工具---》用工具1～  case\_list=gongju2.is\_run\_data('okr-api') # [{用例1}，{用例2}，{用例3}]  gongju4=RequestSend() # 发出接口工具  # 思想 跑用例： gongju2得到用例数据 交给 gongju4 使用  class TestApi:  @pytest.mark.parametrize('case',case\_list)  def test\_run(self,case): # 用例条数 决定了 test\_run 个数～  # 需求1：发接口时候 拼接接口服务器地址～  url=gongju2.loadConfKey('atstudy\_okr','url\_api')['value']+case['url']  method = case['method']  # 坑：来自于数据库 字符串---》字典才能给 工具4使用解决 eval（）  headers = eval(case['headers'])  #print("当前用例请求头类型",type(headers))  cookies = eval(case['cookies'])  data = eval(case['request\_body'])  relation =str( case['relation'])  case\_name=case['title']  try:  # 工具3  logger.info(f"主人，正在执行{case\_name}用例")  # 接口发送请求～  res\_data=gongju4.send(url,method,data=data,headers=headers,cookies=cookies)  if res\_data:# 接口发送成功 再调用～  if relation!=None: # 只有上游接口需要设置关联值  self.set\_relation(relation,res\_data)  #guanlian=res\_data['headers'].get("Authorization",'无临时值～')  logger.info(f"实际结果：{res\_data}")  #发送失败 断言失败 发送成功 不能说明断言成功 进一步比对预期和实际 ==》测试结果  except:  logger.error("用例执行失败，请进一步排查")  assert False  return res\_data  # 1def set\_relation(形参relation，实际结果res\_data):  def set\_relation(self,relation,res\_data):  # 健壮性 1验证 2 try方便定位报错代码～  try:  # 1.1判断relation 是否有值=非空=》上游用例  if relation:  # 1.2 切割 relation  jwlist=relation.split("=") #[jwtoken,Authorization]  # 1.2.1 =左边 ====》动态属性名字  zuobian=jwlist[0]  # 1.2.2 根据=右边到实际结果res\_data中查找===》动态属性值  youbian=jwlist[1]  guanlian=res\_data.get("headers").get(youbian,"无关联值")  # 1.3 setattr(类，动态属性名字，动态属性值)==》存放结束  setattr(DynamicParam,zuobian,guanlian)  #验证：  print("志玲验证是否存放成功:",getattr(DynamicParam,zuobian,"存放临时值失败～"))  except:  print("47~57行写的可能有问题～～")  if \_\_name\_\_ == '\_\_main\_\_':  pytest.main(["-s","-v","test\_run.py"]) |

下游准备代码:

|  |
| --- |
| # 我是公共的基础代码 为其他代码服务～  # 导入json库  import json  # 导入Template类  from string import Template  # 导入re库  import re  # 正则查找  def find(data):  # 判断data类型是否为字典  if isinstance(data, dict):  # 对象格式化为str  data = json.dumps(data)  # 定义正则匹配规则  pattern = "\\${(.\*?)}" # 提取  # 按匹配进行查询，把查询的结果放到列表中返回  return re.findall(pattern, data)  # 进行参数替换  # 要求 被替换数据 包含 ${}  # 替换 字典 字典中的键和 ${}一致才能替换～  def relace(ori\_data, replace\_data):  ori\_data = json.dumps(ori\_data)# 对象格式化为str  s = Template(ori\_data)  # 使用新的字符，替换  return s.safe\_substitute(replace\_data)  # 测试代码  if \_\_name\_\_ == '\_\_main\_\_':  # 验证第一个方法  # 查找id的数据  red = find({"id": "${id\_name}",  "title": "xiaojiang", "alias": "${heiheihei}", "sex": None})  print(red)  # 验证第二个方法  ori\_data = {"authorization": "${jwtoken}"}  replace\_data = {'jwtoken': '111111111'}  print(relace(ori\_data, replace\_data)) |

预告：

### 内容2： 断言+结果回写：

### 内容3： git+jenkins+allure （无人值守）