全程66互联网接口自动化第五次

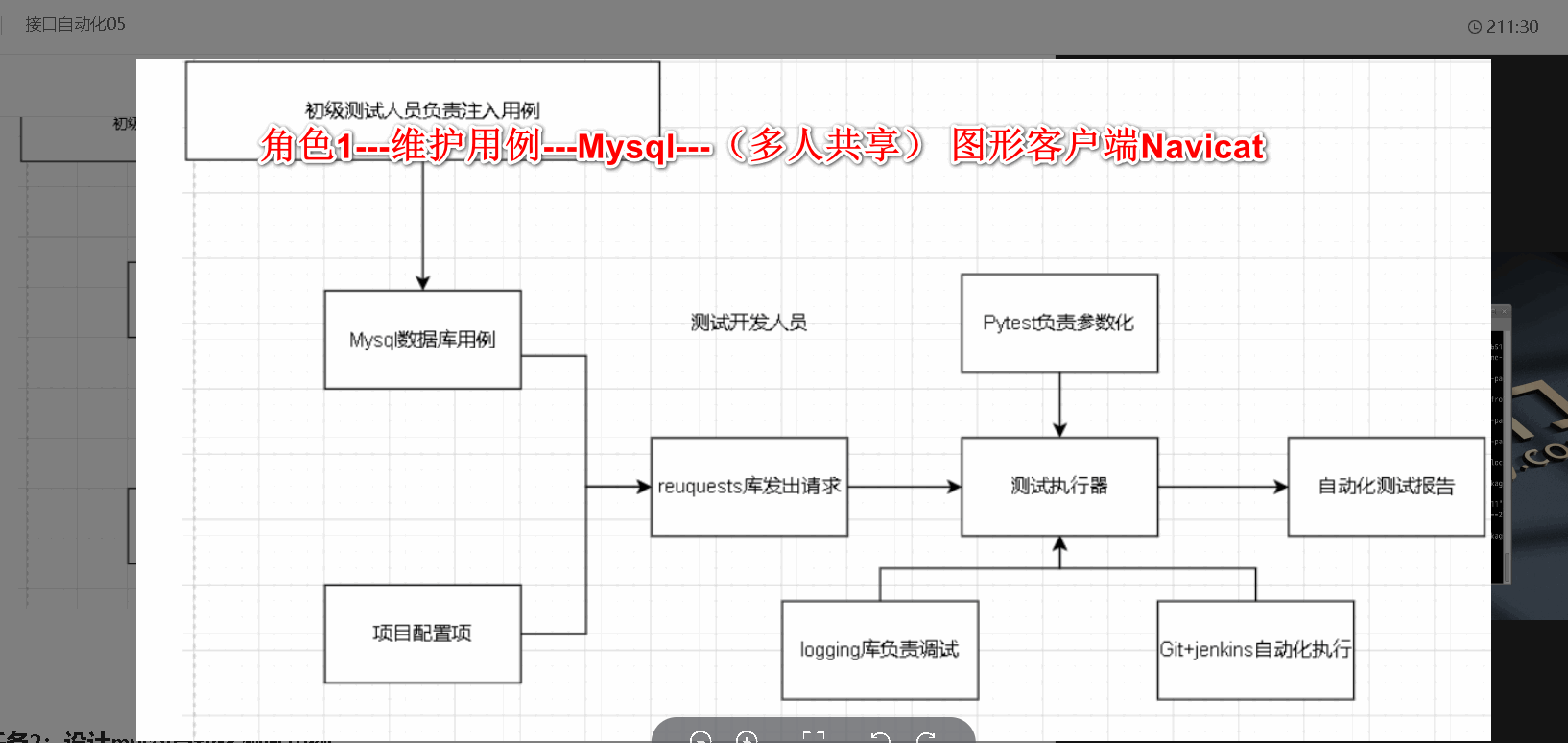
回顾：

1 request API + pytest.mark.paramterize (DDT) +logging (日志)+Pymysql（API）

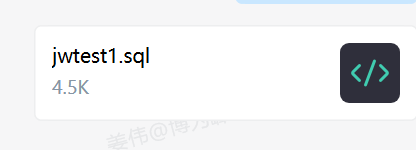
2 重要： **面向工具类编程+ 熟悉企业二次封装的代码**

**3 Python 语法 3.1 降低代码风险 get 和try 3.2 字符串 split Template 正则查找 eval**

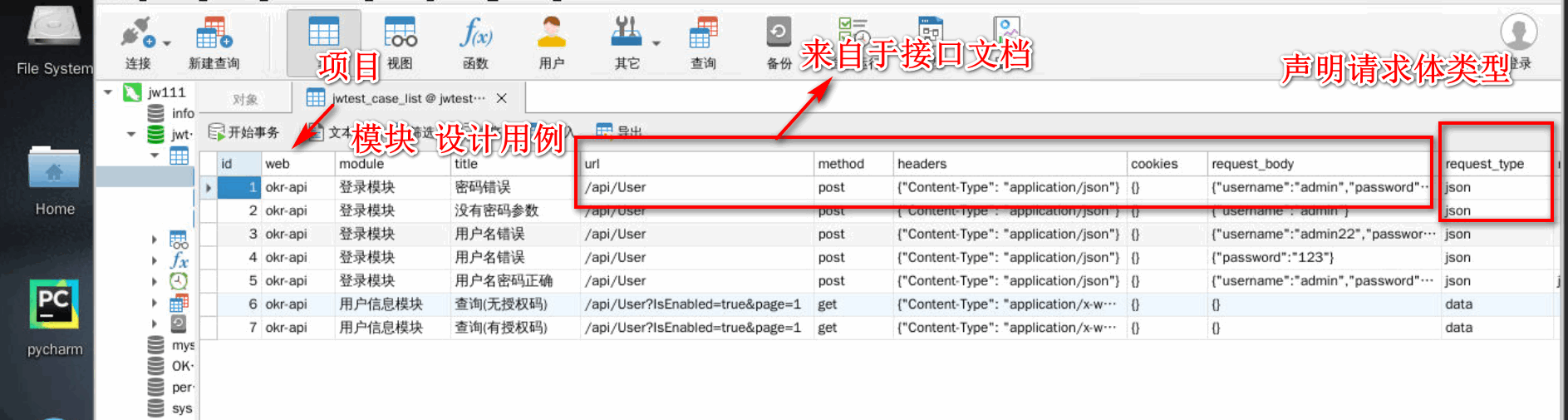
### 内容1： 接口自动化----架构图：

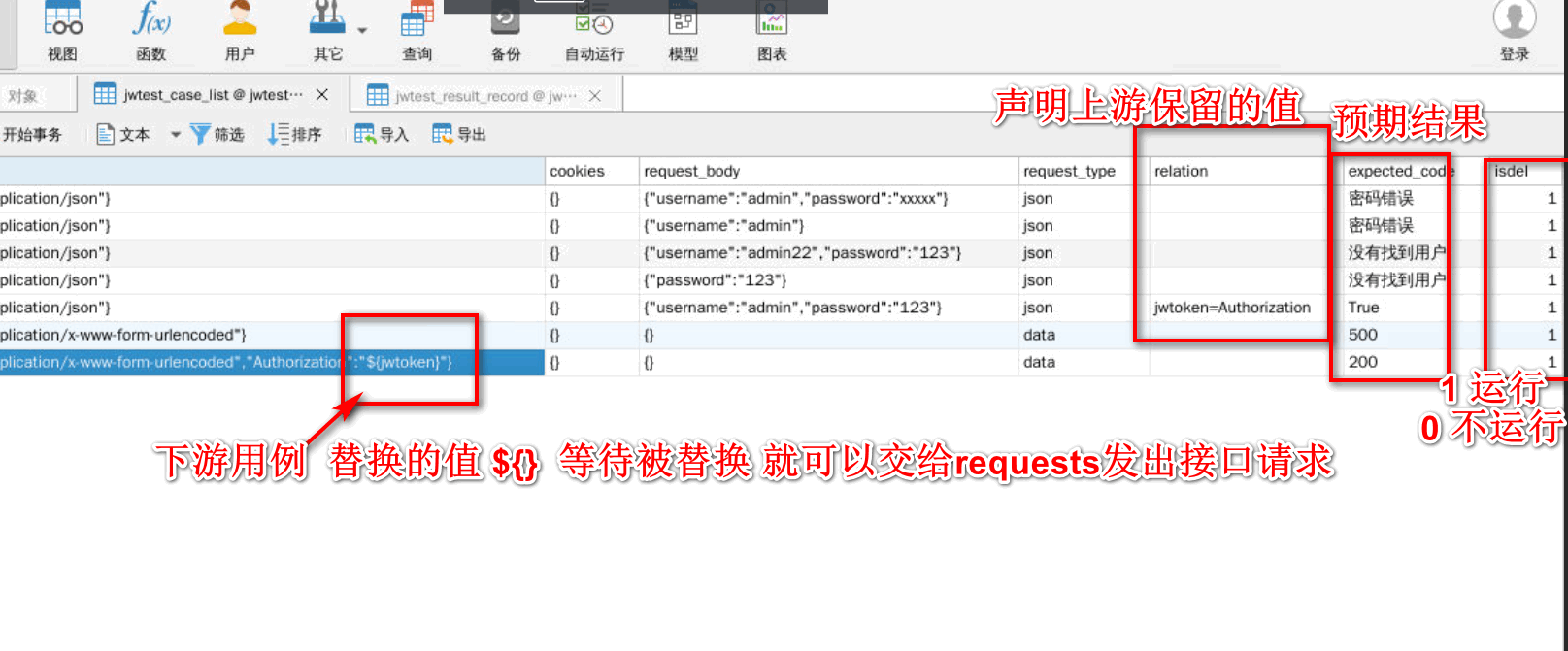


**熟悉: 接口测试用例**



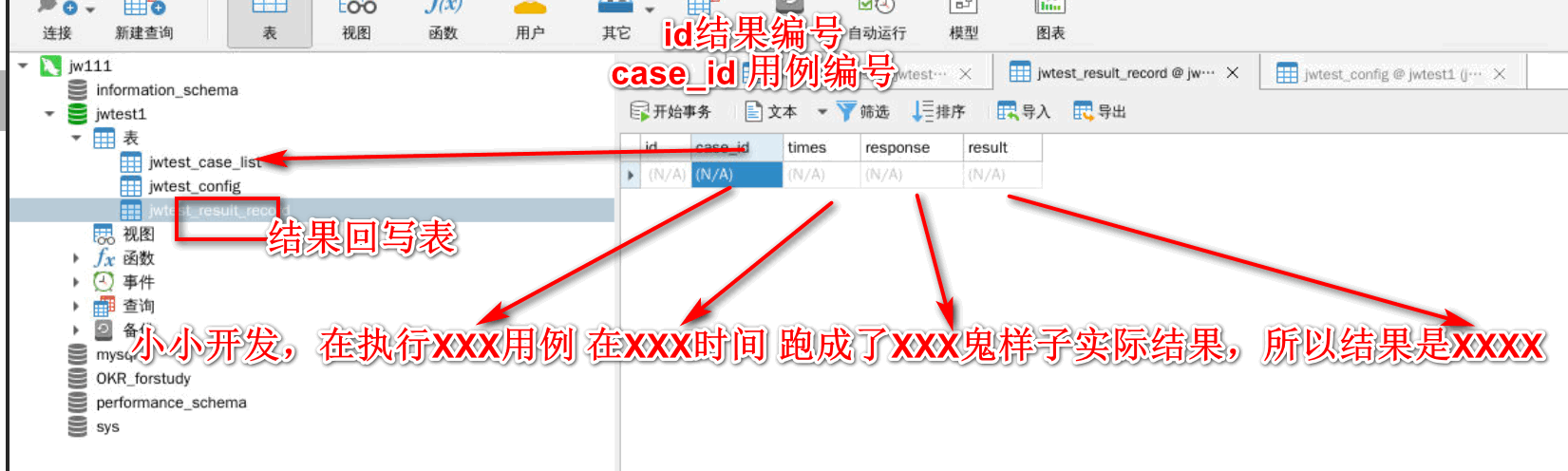




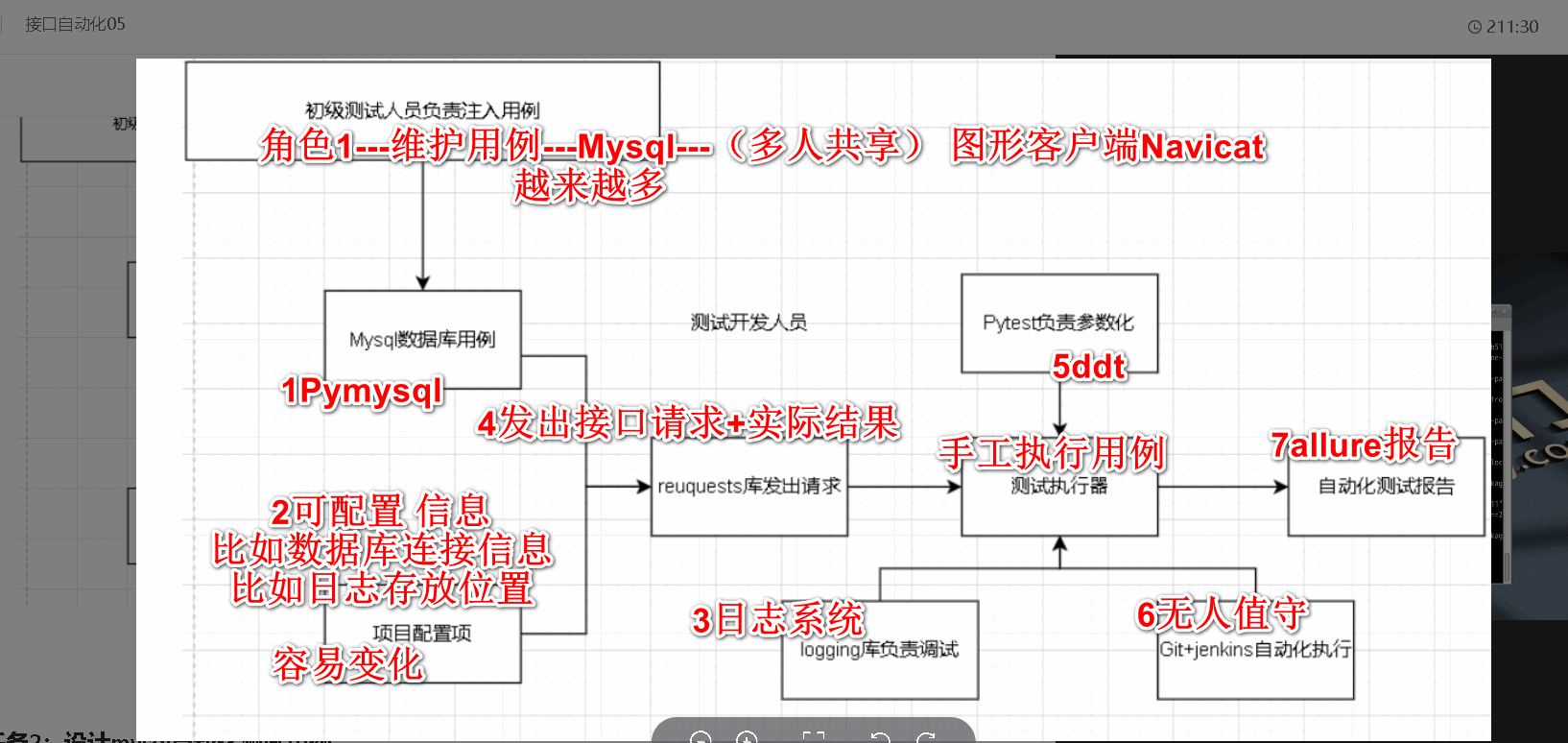




结果 存放： 只存 一个 测试是否通过？----》不可以 描述不详细



**架构： 背~~~指导的剧情~**



小结：

1测试用例 不断维护

2测试开发 v1--》v2--》v3 一次编写到处运行

3 sh脚本

### 内容2 Pymysql 加载用例：

**需求： 忘了+ 根本没学过？**

**解决： 工具类----使用？**

**-----静态角度： ---以函数单位注释**

|  |
| --- |
| **工具名字 类名 MysqlUtil**  **功能1 get\_fetchone 获取单条数据**  **功能2 get\_fetchall 获取多条数据**  **功能3 sql\_execute 更新类sql** |

**-----动态角度： ------运行一下满足注释**

|  |
| --- |
| # 导入PyMySQL库  import pymysql  # 导入数据库的配置信息  # from config.settings import DB\_CONFIG  # 设置数据库工具类的名称  class MysqlUtil:  def \_\_init\_\_(self):  self.db = pymysql.connect(host="127.0.0.1", user="root", password="123456", database="jwtest1", port=3306,  charset="utf8")  # 读取配置文件，初始化pymysql数据库连接  #self.db = pymysql.connect(\*\*DB\_CONFIG)  # 创建数据库游标 返回字典类型的数据  self.cursor = self.db.cursor(cursor=pymysql.cursors.DictCursor)  # 获取单条数据  def get\_fetchone(self, sql):  # 执行sql  self.cursor.execute(sql)  # 查询单条数据，结果返回  return self.cursor.fetchone()  # 获取多条数据  def get\_fetchall(self, sql):  # 执行sql  self.cursor.execute(sql)  # 查询多条数据，结果返回  return self.cursor.fetchall()  # 执行更新类sql  def sql\_execute(self, sql):  try:  # db对象和指针对象同时存在  if self.db and self.cursor:  # 执行sql  print("sql是",sql)  self.cursor.execute(sql)  # 提交执行sql到数据库，完成insert或者update相关命令操作，非查询时使用  self.db.commit()  print("sql执行成功～！")  except Exception as e:  # 出现异常时，数据库回滚  self.db.rollback()  # 返回结果为失败  return False  # 关闭对象，staticmethod静态方法，可以直接使用类名.静态方法。  @staticmethod  def close(self):  # 判断游标对象是否存在  if self.cursor is not None:  # 存在则关闭指针  self.cursor.close()  # 判断数据库对象是否存在  if self.db is not None:  # 存在则关闭数据库对象  self.db.close()  # 测试代码  if \_\_name\_\_ == '\_\_main\_\_':  # 验证编写的方法  gongju1 = MysqlUtil()  res1=gongju1.get\_fetchone("select \* from jwtest\_case\_list")  print(res1)  res2 = gongju1.get\_fetchall("select \* from jwtest\_case\_list")  print(res2)  res3=gongju1.sql\_execute("insert into jwtest\_result\_record (case\_id,result) values ('9999','测试通过');")  print(res3) |

小结：

1工具类价值----》不熟悉API+遗忘情况下~使用工具~

2库 还学不学？----》能学就学 不能学也能干活~ 2.1自己优化 2.2 自信 2.3 自定义工具类~~

3 封装了 Pymysql的API

说明 通用工具 -----》 Pymysql logging 提供（大量人员）

专有工具 -----》 用例工具类 （测试）

### 内容3： 自定义用例工具类 （CaseUtil）

**编程方式： 自定义用例 工具类思想 +目的： 小白 编程方式~~~ ？？**

|  |
| --- |
| **思路：**  **1细化需求**  **1.1 加载所有用例**  **1.2 加载需要执行用例**  **1.3 加载用例配置**  **1.4 回写存放用例结果**  **2 需求---》空函数+ 自测代码 （总比空白文件好）+其他工具类~**  **3 依次实现 函数 依次调试**  **价值： 方便交流 降低编程难度~~** |

|  |
| --- |
| #其他工具类  from utils.mysqlutil import MysqlUtil  gongju1=MysqlUtil() # gongju1 封装pymysql的API  #我是用例工具类  class RdTestCase:  # 1.1加载所有用例  def load\_all\_case(self):  print("我要写代码了～～～")  # 1.2加载需要执行用例  def is\_run\_data(self):  pass  # 1.3加载用例配置  def loadConfKey(self):  pass  # 1.4回写存放用例结果  def updateResults(self):  pass  if \_\_name\_\_ == '\_\_main\_\_':  gongju2=RdTestCase()  gongju2.load\_all\_case() # 自测第一个 |

练习： 完成gongju2 的第一个功能 ？

|  |
| --- |
| #其他工具类  from utils.mysqlutil import MysqlUtil  gongju1=MysqlUtil() # gongju1 封装pymysql的API  #我是用例工具类  # 考虑 通用尽可能复用  class RdTestCase:  # 1.1加载所有用例--->复用性更高～  def load\_all\_case(self,jwweb):  # 需求： 为了复用 应该在使用工具传递测试项目  # 解决： 形式参数  sql=f"select \* from jwtest\_case\_list where web='{jwweb}'"  return gongju1.get\_fetchall(sql)  # 1.2加载需要执行用例 标准 isdel字段 是否为1  # 方式1 sql实现 select XXX where web='{jwweb}' and isdel=‘1’ “  # 方式2 Python实现 已经能获取所有数据了～（上课 加强Python+语法：列表推导式）  # 根据字典的isdel 键进行筛选 存放到指定的 列表 即可  def is\_run\_data(self,jwweb):  #0 准备空列表 等待存放需要被执行的用例～  jw\_run\_list=[]  # 1 获取所有用例self.load\_all\_case()  all\_case=self.load\_all\_case(jwweb)  # 2 for 取出所有用例  for case in all\_case:  # 3 判断每条用例isdel字段  if case['isdel']==1:  # 4 isdel 为1用例存放到列表中即可～  jw\_run\_list.append(case)  return jw\_run\_list # 存放需要执行的用例～  # 1.3加载用例配置  def loadConfKey(self):  pass  # 1.4回写存放用例结果  def updateResults(self):  pass  if \_\_name\_\_ == '\_\_main\_\_':  gongju2=RdTestCase()  print("自测第一个",gongju2.load\_all\_case('okr-api')) # |

**下午：完成4个工具 ----》独立接口用例执行起来~**