全程66期接口自动化第四次

回顾：

1工具类 -----来源+价值 **(基于工具类编程** 码农？ 二次封装？)

2 价值 不需要 API 源码 直接复用~

### 内容1： 优化工具类

**需求1： 日志存放位置 存放在指定包下（比如log包）**

解决：

1 绝对路径？----



解决  **需要一段代码 可以识别当前路径路径** 然后使用即可

Python 提供OS库

API:

|  |  |
| --- | --- |
| 对象名字 | 对象的API |
| OS | \_ \_file\_ \_ 魔法变量 得到当前文件的名字路径 |
|  | 1 os.path.dirname(\_ \_file\_ \_ ) 当前目录  多写一层  os.path.dirname(os.path.dirname(\_ \_file\_ \_ )） 上一层目录  2 os.sep 路径分割符号  linux /  windows \  拼接目录  3 os.path.join(目录，文件名)  在目录上拼接文件 |
| time | sleep 强制等待  localtime() 打印出当前时间信息  strftime("%Y\_%m\_%d %H:%M:%S",时间) |

|  |
| --- |
| project\_path=os.path.dirname(os.path.dirname(\_\_file\_\_))  log\_path=project\_path+os.sep+"log"  log\_path\_name=os.path.join(log\_path,"jw2.log") |

**需求2： 日志名字 包含当天时间 合理**

**思路 ： 只要取出年月日 格式化字符串 作为文件一部分即可**

**语法文件:**

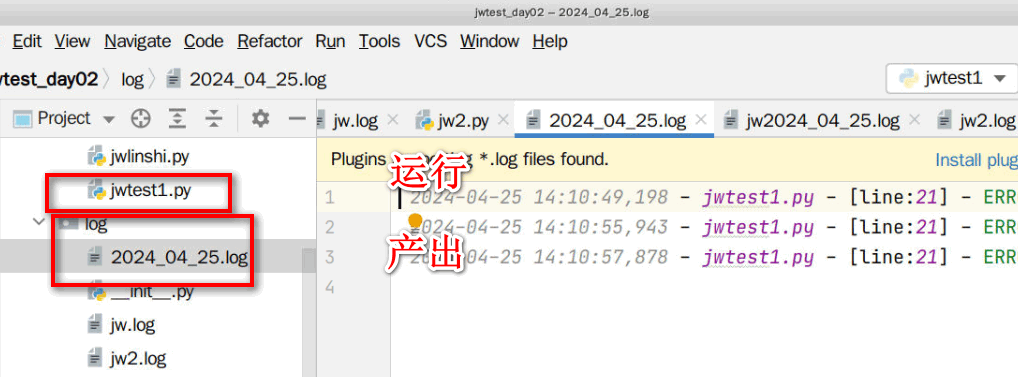
|  |
| --- |
| **import os**  **import time**  **print(\_\_file\_\_)**  **print(os.path.dirname(\_\_file\_\_))**  **project\_path=os.path.dirname(os.path.dirname(\_\_file\_\_))**  **print(project\_path)**  **print(project\_path+os.sep+"log")**  **log\_path=project\_path+os.sep+"log"**  **print(os.path.join(log\_path,"jw.log"))**  **print(time.localtime())**  **print(time.strftime("%Y\_%m\_%d %H:%M:%S",time.localtime()))**  **print(time.strftime("%Y\_%m\_%d",time.localtime()))** |

**优化完成后的工具类**

|  |
| --- |
| **# 日志系统？（帮派一个道理）**  **# 类---\_\_init\_\_---》对象**  **import logging**  **import os**  **import time**  **class logutil:**  **def \_\_init\_\_(self):# 想表达 使用工具（对象）帮派 已经产生**  **# 1# 老大logger = getLogger（“名字”）**  **self.logger=logging.getLogger("坤坤")**  **# 1.1# setLevel(ERROR)# 设置总级别 - --安全**  **self.logger.setLevel(logging.INFO)**  **# 2# 控制台小弟1StreamHandler()**  **sh=logging.StreamHandler()**  **# 2.1# setLevel(显示级别# 子级别)# 不可以比总级别详细**  **sh.setLevel(logging.INFO)**  **# 2.2# setFormat(设置格式)**  **sh.setFormatter(logging.Formatter('%(filename)s | %(asctime)s | [hanghao:%(lineno)d] - %(levelname)s: %(message)s'))**  **## 3**  **# 文件小弟1# 文件模式 # append 日志不断追加**  **# 相对或者绝对路径都可以**  **#log\_path\_name='/root/PycharmProjects/jwtest\_day02/log/jw.log'**  **project\_path=os.path.dirname(os.path.dirname(\_\_file\_\_))**  **log\_path=project\_path+os.sep+"log"**  **log\_name=f"{time.strftime('%Y\_%m\_%d',time.localtime())}.log"**  **log\_path\_name=os.path.join(log\_path,log\_name)**  **fh=logging.FileHandler(log\_path\_name,mode='a',encoding='utf-8')**  **# 3.1# setLevel(显示级别# 子级别)**  **# 3.2# setFormat （设置格式）**  **fh.setLevel(logging.WARNING)**  **# 2.2# setFormat(设置格式)**  **fh.setFormatter(**  **logging.Formatter(' %(asctime)s - %(filename)s - [line:%(lineno)d] - %(levelname)s: %(message)s'))**  **fh.close()**  **# 4# 老大**  **# 派小弟干活**  **# 老大.addHandler(小弟)**  **self.logger.addHandler(sh)**  **self.logger.addHandler(fh)**  **#**  **# 5**  **# 对外# 返回出对象# 老大 （给# 其他系统比如pytest# 开发系统进行调用# ~）**  **def log(self):**  **return self.logger**  **# 获取老大**  **# 注释：**  **# 1 日志系统 代替print 完成日志终端分发**  **# 2 日志级别划分**  **# 3 日志支持 格式设置**  **logger=logutil().log()**  **if \_\_name\_\_ == '\_\_main\_\_':**  **# 自测**  **# 日志系统 默认的级别 是WARNING 级别**  **logger.debug("我是代码调试信息11～")**  **logger.info("我是系统正常输出信息11")**  **logger.warning("----我是警告信息11，比如内存不够～-----")**  **logger.debug("我是代码调试信息22～")**  **logger.error("-----我是一个小bug11！！------")**  **logger.info("我是系统正常输出信息22")**  **logger.warning("----我是警告信息22，比如内存不够～-----")**  **logger.error("-----我是一个小bug22！！------")**  **logger.warning("----我是警告信息33，比如内存不够～-----")**  **logger.critical("----我是bug 严重事故～导致系统崩溃-----")** |

**运行[jwtest1.py](https://jwtest1.py)文件**

**效果：**

****

### 内容2： Pymysql灵魂三问：

**什么是Pymysql？**

Python 连接mysql数据库 对数据 SQL 操作（类似 mysqldb）

**为什么Pymysql？**

**需求： 测试用例----》表 -----》Excel------》Mysql？**

**测试人员： 图形操作 （DBeaver或者Navicat）**

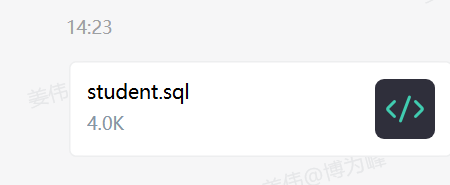
**测试开发： 1 Mysql 支持多人远程连接（共享用例--实时更新）**

**2 Excel 点点点 Mysql sql语言交给 Python更容易自动化**

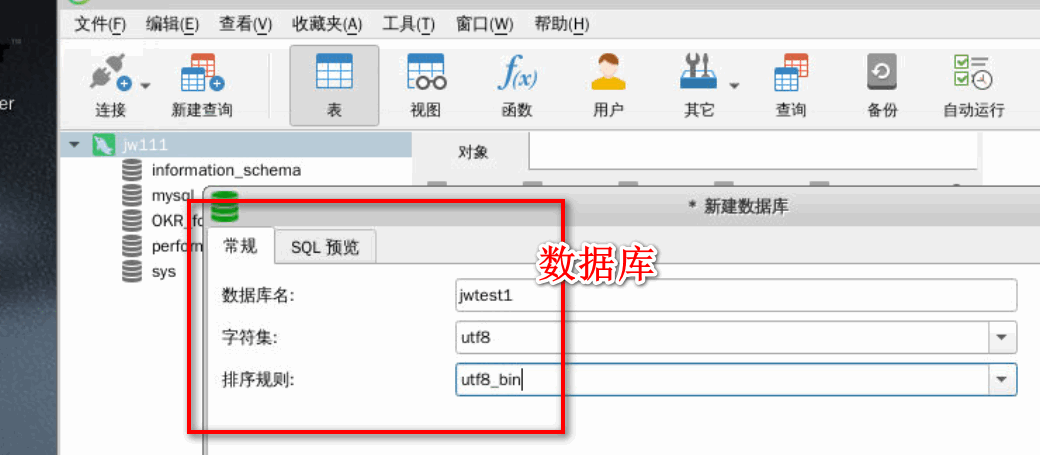
**3 大量用例 （微服务接口用例太多）==》Mysql 速度相对快**

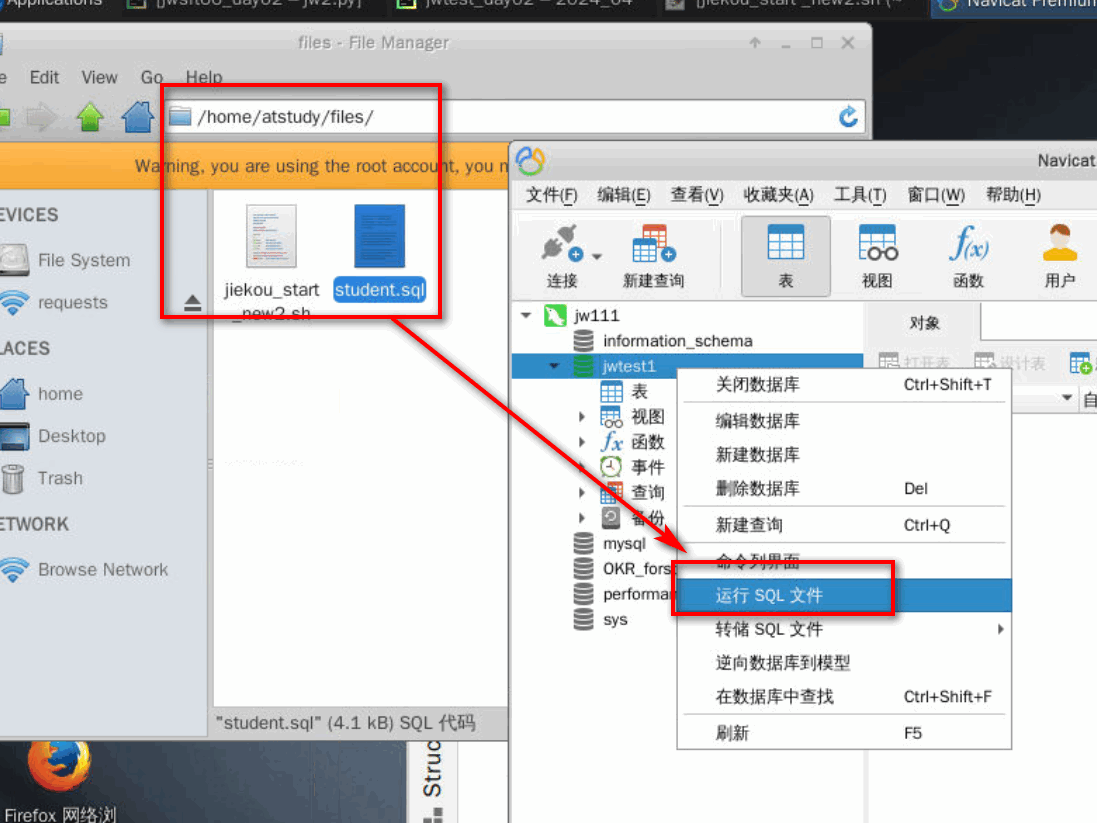
**如何掌握Pymysql？**

1环境 +准备数据









2 python 能对这些数据增删改查即可~

### 内容3： 快速入门

**例子 Python 对mysql数据库进行查询**

**Pymysql 就是一个 代码形式客户端**

**Navicat 是 图形界面形式客户端**

|  |  |
| --- | --- |
| **对象名** | **对象API** |
| **Pymysql** | **1创建db连接**  **connect**  **2 创建游标cursor() ？**  **默认：元组游标**  **推荐： 字典游标**  **[{数据1}，{数据2}，{数据3}]**  **pymysql.cursors.DictCursor**  **2.1 执行SQL语句**  **2.2 保存 select 语句的结果**  **3 从游标 提取select 结果**  **3.1 fetchone() 提取一行数据**  **3.2 fetchmany(n) 提取 n行数据**  **3.3 fetchall() 提取所有数据**  **4 更新类sql（insert update delete）**  **4.1 commit() 数据库提交保存修改**  **4.2 rollback() 回滚 撤销 保存之前的修改**  **5 close()**  **先关闭游标 再关数据库** |
|  |  |

**为什么要叫游标？来自于 游标卡尺**

****

**实现：**

|  |
| --- |
| **#**  **# 1创建db连接**  **# connect**  **import pymysql**  **# 本地机器**  **jwdb=pymysql.connect(host="127.0.0.1",port=3306,**  **user="root",password="123456",database="jwtest1",**  **charset="utf8")**  **# 2 创建游标cursor() ？**  **jwcursor=jwdb.cursor(pymysql.cursors.DictCursor) # 字典游标方便 pytest的 ddt 对接~**  **# 2.1 执行SQL语句**  **sql="select \* from xsb;"**  **jwcursor.execute(sql)**  **# 2.2 保存 select 语句的结果**  **# print(jwcursor.fetchone())**  **# print(jwcursor.fetchmany(2))**  **# print(jwcursor.fetchone())**  **# print(jwcursor.fetchall())**  **# print(jwcursor.fetchone())**  **all\_data=jwcursor.fetchall() #元组数据～**  **# 取出班级？--->**  **# 元组游标缺点 1 数数麻烦 2 记录长度变化的 3 可读性差**  **print(all\_data)**  **print(all\_data[2]['bj'])**  **print(all\_data[2])**  **# print(all\_data[0])**  **# 3 从游标 提取select 结果**  **# 3.1 fetchone() 提取一行数据**  **# 3.2 fetchmany(n) 提取 n行数据**  **# 3.3 fetchall() 提取所有数据**  **# 5 close()**  **# 先关闭游标 再关数据库**  **jwcursor.close()**  **jwdb.close()** |

### 内容4： 更新类sql的使用（insert 为例）

|  |
| --- |
| 思路：  1fetchXXXAPI 不需要  2 commit() 保存API  #  # 1创建db连接  # connect  import pymysql  # 本地机器  jwdb=pymysql.connect(host="127.0.0.1",port=3306,  user="root",password="123456",database="jwtest1",  charset="utf8")  # 2 创建游标cursor() ？  jwcursor=jwdb.cursor(pymysql.cursors.DictCursor) # 字典游标  # 2.1 执行SQL语句  sql="insert into xsb set xh='012',xm='奥特曼',xb='男';"  jwcursor.execute(sql)# 游标执行sql  jwdb.commit() # 数据库保存  jwcursor.close()  jwdb.close() |

### 内容5： Pymysql 事务：

需求： mysql 要求 多条更新类 SQL 语句 同时成功 同时失败？

例子 冠希 转账阿娇 100块

sql1 update 冠希 money-100

sql2 update 阿娇 money+100 失败 撤销 冠希 -100

。。。。sql3

sql2 是否成功决定 sql1 是否成功~

python 事务实现：

|  |
| --- |
| try:  执行sql1  执行sql2  commit()  except:# 执行try失败分支  rollback（） 产生回滚 已经成功sql撤销影响  else:  不回滚 |

代码示例:

|  |
| --- |
| #  # 1创建db连接  # connect  import pymysql  # 本地机器  jwdb=pymysql.connect(host="127.0.0.1",port=3306,  user="root",password="123456",database="jwtest1",  charset="utf8")  # 2 创建游标cursor() ？  jwcursor=jwdb.cursor(pymysql.cursors.DictCursor) # 字典游标  # 2.1 执行SQL语句  try:  sql1="insert into xsb set xh='102',xm='杨过',xb='男';"  jwcursor.execute(sql1)# 游标执行sql  sql2="insert into xsb set xh='103',xm='小龙女',xb='女';"  jwcursor.execute(sql2)# 游标执行sql  jwdb.commit() # 数据库保存  except:  print("有sql失败，开始回滚。。")  jwdb.rollback()  else:  print("全部sql执行成功")  finally:  print("事务结束～")  jwcursor.close()  jwdb.close() |

**pymysqlapi结束 下次出现以工具类方式出现**

**内容6： Python 语法加强 （字符串方便加强）**

**1字符串切割**

**2 字符串查询**

**3字符串类型替换**

**4 字符串类型转换**

|  |
| --- |
| **# 1字符串切割---》列表～**  **import re**  **from string import Template**  **jwstr="zhangsan=lisi=wangwu"**  **jwlist=jwstr.split("=")**  **print(jwlist)**  **print(jwlist[1])**  **# 2 字符串查询查找 提取目标数据～**  **jwstr2="上游接口授权${aaaaaaa},我爱${大s}，我爱${志玲}，我爱${一博} ，我爱${社会主义}"**  **# 正则表达式 (.\*?) 0~多个字符**  **# 根据目标数据的左右边界策略 提取数据呀～**  **# 左边界 ${ 右边界}**  **zhengze="\\${(.\*?)}" #python \\ 代表正则表达式～**  **# Python re库 findall api 专门是根据正则从数据中完成查找 存放到列表～**  **list=re.findall(zhengze,jwstr2)**  **print(list)**  **# 3字符串类型替换**  **# 被替换数据 要求必须字符串 包含 ${}**  **# 替换数据 要求 必须是字典 ${} 和字典键必须一致～**  **jwstr3="我是下游接口用例 需要授权码${jwtoken}"**  **# 运行了接口**  **shangyou\_jieguo={"jwtoken":"aaaaaaaaaa"}**  **#需要将 授权码aaaa 替换到 字符串中下游接口才可以运行～**  **# Python 模板字符串 Template(字符串)==》支持替换**  **# safe\_substitute ==>发生替换**  **genghuan=Template(jwstr3)**  **new\_shiji=genghuan.safe\_substitute(shangyou\_jieguo)**  **print(new\_shiji)**  **# 4 字符串类型转换**  **#API**  **# 字符串 --json.loads() eval()---》字典**  **data="{'username':'admin','password':'123'}"**  **print(type(data)) # 为什么转字典？ 答：键值对取值 +才能交给requests发送**  **print(data)**  **data2=eval(data)**  **print(data2)**  **print(type(data2),data2['username'])**  **# 字典 --json.dumps() str()--》字符串** |

**小结： 测试开发自动化框架 所有准备工作到此结束**

**预告 1自动化框架 架构 2综合用前四次语法 和库 基于工具类 完成框架开发~**