技术组 ToolKit db_api 文档

执行 api 时，如果报错，我写的 api 会自动进行 RollBack 回滚到链接到数据库前的版本。

我们这样描述一个 api：

名字（大驼峰，长点没关系）

1. 输入参数类型与描述
2. 该 api 执行的效果
3. 该 api 执行的返回值
4. api 的位置

请大家提出 api 需求时，也这样提出。这样大家可以在我实现 api 前就开始写相关代码了。

如果我在实现 api 时，对大家的提出有出入，我会用红色的标出来，如果大家觉得原本那样更好，可以讨论；如果觉得我改的更好，请更改一下代码，符合我写的 api（我会尽量保持完全还原大家的需求的！大家辛苦啦）。

例如：

api 描述方法举例

GetTeacherInfoByName

1. 共一个输入，无默认值。Name: str，表示将要查找的用户的名字
2. 执行的效果：是 DQL 语句，对数据库不做更改。
3. 返回值：List[dict()]，每个列表中有一些 item，每个 item 是一个字典。这个字典包含：'id':int，该用户在表中的 id,'p_id':int：该用户如果是教练，其上负责人的 id，如果不是教练就是 0；"wechat_nickname":str,微信昵称,"school_id":int，其学校 id, "upload_limit":int，上传限制, "viewing_problem":int，当前批改题目, "type":"str"，仲裁或教练或负责人。
4. api 位置：db_api/DataQueryApis/GetTeacherInfoApis.py

api 使用方法举例：

```
from db_api import customTransaction
#在 db_api/__init__.py 中，我给大家自动创建了一个 customTransaction 实例。
#在其初始化时，会自动与数据库连接，并要求大家输入数据库各参数。千万不要图方便把密码等放在代码里面！不然你 git push 之后很可能就 暴露了（）
#引入实例化的 CustomTransaction 对象，该对象用于与数据库交互。请大家不要自行创建新的实例，就用这个就好。

from db_api.DataQueryApis.GetTeacherInfoApis import GetTeacherInfoByName
#引入 GetTeacherInfoApis 中的 GetTeacherInfoByName 这个类

getTeacherInfoApi=GetTeacherInfoByName(Name = "史景喆")
#实例化一个 getTeacherInfoApi 对象. 输入是字符串"史景喆"


returned=customTransaction.executeOperation(getTeacherInfoApi)
#customTransaction.executeOperation()会调用实例化的 getTeacherInfoApi 中的 execute 方法，去对数据库进行操作，并且返回其返回值。


print(returned)
#可以观察返回值
sjzid = returned[0]['id']
#returned 是一个列表，其中每一个 item 表示搜索到了一个符合要求（也就是名字是史景喆的）用户。取出其中第零个，这是一个字典。找到"id"键所对应的键值，这就是这一个名叫史景喆的用户在用户表中的 id。

#customTransaction.commit()
#这个请大家不要加。是用来保存更改的。debug 时，请用查询 api 查询之前的更改是否正确。
#只用调用了 commit 这之后，更改才会被提交，保存在数据库里。如果没有调用这一函数，最后退出时，析构函数将会自动将 rollback（回滚，也就是撤销所做的改变）。同时，如果在执行过程中，有任何的问题或报错，都会 rollback，将数据库回滚到本次连接发生前（也就是 from db_api import customTransaction 时实例化 customTransaction 前）数据库的状态。
#但是这个例子里，只是做了查询而没有做更改，所以是否提交对于数据库的值而言没有影响。
```

dp_api 中的子文件夹：

1. DataManagingApis 这里面有对 data 作修改的 api
    a. ChangeTeacherInfoApis.py 这里是对老师用户（三类）作修改的 api
        i. MakeAllTypesToBeArbiter(ChangedUserId:int)
            1. 效果：将任意类型的老师转换为仲裁者，注意有题目未批改的负责人不可。
            2. 返回值：无
        ii. MakeAllTypesToBeSubCoach(ChangedUserId:int, ItsNewSupId:int)
            1. 效果：将任意类型的老师转换为另一老师的下属教练，注意有题目未批改的负责人不可，且 itsnewsup 必须是负责人。
            2. 返回值：无
        iii. MakeAllTypesToBeSupTeacher(Id:int)
            1. 效果：将任意类型的老师转换为负责人
            2. 返回值：无
        iv. ChangeAllTypesMarkingSubject(ChangedUserId:int, ItsNewMarkingSubject:int in [1,10])
            1. 效果：将任意类型的老师的批改题目变换为 ItsNewMarkingSubject。注意，不会改变 teacher 用户的类型（原本是负责人、教练、仲裁者，改变后仍然是原类型）。然而，如果是存在未批改题目的负责人，将不可，会报错。
            2. 返回值：无
        v. ChangeAllTypesSchool(ChangedUserId:int, NewSchoolId:int)
            1. 效果：将任意类型的老师的学校变成 NewSchoolId.注意，schoolid 可以通过查询获得。
            2. 返回值：无
        vi. ChangeAllTypesUploadLimit(ChangedUserId:int, NewUploadLimit: int)
            1. 效果：将任意类型的老师的上传限制变成 NewUploadLimit。
            2. 返回值：无
            3. 注释：例如说，把个人用户老师（因为我们系统里面，个人报名者是负责人，其下学生就他一个）的上传限制都变成 1，可以这样做：先通过 GetSchoolInf//////////////oByName 获得名叫 “个人” 的学校的 schoolid，然后通过 GetTeacherInfoBySchoolId 获得 schoolid 等于之的所有教师的 id，然后调用 ChangeAllTypesUploadLimit 传入该 id，然后更改其上传限制为 NewUploadLimit。

vii. VerifyTeacherUserToBeSupTeacher(ToBeVerifiedId:int, Name:str, SchoolId:int, ViewProblem: int in [1,10] default = 1, UploadLimit: int default = 100,)

1. 效果：将 Id 为 ToBeVerifiedId 的待审核用户审核为负责人（这个 id 可以通过 GetToBeVerifiedTeacherInfoByWechatName 这一个 api 获得），而且其姓名设置为 Name，SchoolId 设置为 SchoolId（不是两个 L 也不是两个 i，//////////////////////////////是一个小写 L 一个大写 i !），ViewProblem 设置为 ViewProblem， 上传限制设置为 UploadLimit。

   注意！如果监测到该老师已经审核，则会在控制台进行输出，但不对其做出任何更改！

2. 返回值：无

b. ChangeSchoolInfoApis.py 这里提供增添一个学校的接口

i. AddNewSchoolByName(Name:str,AreaId:int)，分别是该学校的名字以及其所在区域的 id。

1. 效果：增添一个学校。
2. 返回值：该新增学校的 id。或者你可以在新增后，使用 GetSchoolInfoByName 来获取其 id。如果已经有这一学校，会打印出"已经有这一学校"并返回其 id，返回值是 int 类型。

c. （不再实现，不会允许多加赛区，请大家根据已有的赛区筛选）ChangeAreaInfoApis.py 这里提供着增添一个赛区的接口

i. AddNewAreaByName(Name:str)

1. 效果：增添一个赛区。
2. 返回值：该新增赛区的 id。或者你可以在新增后，使用 GetAreaInfoByName 来获取其 id。如果已经有这一赛区，会打印提示"已经有这一赛区"并返回其赛区 id。返回值是 int 类型。

2. DataQueryApis 这里面有查询 data 的 api

a. GetTeacherInfoApis.py 这里是对 teacher 用户的信息做查询的 api

i. GetTeacherInfoByName(Name:str)

1. 效果：是 DQL 语句，对数据库不做更改
2. 返回值/：List[dict()]，每个列表中有一些 item，每个 item 是一个字典。这个字典包含：'id':int，该用户在表中的 id,'p_id':int：该用户如果是教练，其上负责人的 id，如果不是教练就是 0；"wechat_nickname":str,微信昵称,"school_id":int，其学校 id, "upload_limit":int，上传限制, "viewing_problem":int，当前批改题目, "type":"str"，仲裁或教练或负责人。

ii. GetTeacherInfoByFlexibleName(Name:str)

1. 效果：无
2. 返回值：同上，不过只要教师行名字段包含 Name 中的 str，就返回。例如说，查询"bc0"时，名叫"abc01"的教师的信息也会被返回。

iii. GetTeacherInfoBySchoolId(SchoolId:int)

1. 效果：无
2. 返回值：同上，但是这时仅仅返回学校 id 等于 SchoolId 的教师。请注意，这里查询使用的是 SchoolId，所以请先通过获得学校信息的 api 获得学校信息再做查询。

iv. GetToBeVerifiedTeacherInfoByWechatName(WechatName:str)

1. 效果：无，dql 语句
2. 返回值：List[dict()],每个列表中有一些 item，每个 item 是一个字典。这个字典包含：'id':int，该待审核用户审核后的 id。

v. GetToBeVerifiedTeacherInfoByFlexibleWechatName(WechatName:str)

1. 同上
2. 类似

b. GetSchoolInfoApis.py 这里是对学校信息做查询。

i. GetSchoolInfoByName(Name:str)

1. 效果：无
2. 返回值：List[dict()]，每一个列表中有一//////////////////////////////些 item，每个 item 是一个字典。这个字典包含：'id'：int，该学校在学校表中的 id。'school_name'：str，该学校的名字。'area_id':该学校所在赛区的 id

ii. GetSchoolInfoByFlexibleName(Name:str)

1. 效果：无
2. 返回值：同上，但是做模糊搜索，只要学校名中包含有 Name，就返回。

c. GetStudentInfoApis.py 这里对学生信息做查询。

i. GetStudentInfoByName(Name:str)

1. 效果：无
2. 返回值：List[dict()]，每一个列表中有一个 item，每个 item 是一个字典。这个字典包含：'id':int，该学生的 id。'student_name'：str，该学生的名字。'teacher_id':该学生对应的老师的 id。'school_id'，该学生所在学校的 id

ii. GetStudentInfoByFlexibleName(Name:str)

1. 省略！

iii. GetStudentInfoBySchoolId(SchoolId:int)

1. 效果：无
2. 返回值：同 GetStudentInfoByName，但是是筛选 schoolid 等于 SchoolId 的。需要先使用 GetSchoolInfoByName 等获取相关的 SchoolId。

d. GetAreaInfoApis.py 这里对赛区信息做查询。

i. GetAreaInfoByName(Name:str)

1. 效果：无
2. 返回值：List[dict()]，每一个列表中有一个 item，每个 item 是一个字典，这个字典包含：'id'，int，该 area 的 id。'area_name'：str，该赛区的名字。

ii. GetAllAreaNamesAndAreaIdAsDict()

1. 效果：无
2. 返回值：dict，每一个 key 是一个赛区名称，其对应的 value 是其 areaid。e.g. '上海':13.