**pip的安装(非课程)**

yum provides pip

5个

curl https://bootstrap.pypa.io/get=pip.py -o get-pip.py

python3 get-pip.py

**pymysql的安装**

pip install pymysql

#############################################

**pymysql模块的使用**

import pymysql

# 建立连接

conn = pymysql.connect(

........

)

# 建立游标

cursor = conn.cursor()

# 执行sql语句

cursor.execute(sql语句)

cursor.executemany(insert,dr) #执行多条

#读取

cursor.fetchone() # 读一行

cursor.fetchmany() # 读取多行

cursor.fetchall() # 读取剩下所有

#移动游标

cursor.scroll(1) # 默认从当前移动

cursor.scroll(2, mode='absolute') # absoute 总是从头开始

# 对数据库有改动,需要确认

conn.commit()

# 关闭

cursor.close() # 关闭游标

conn.close() # 关闭连接

################################################################

SQLALchemy

--不需要书写SQL语句

--不限于mysql,只要是关系型数据库

--使用ORM

ORM即对象关系映射

--O:Object: 面向对象变成中对象

--R:Relationship: 关系, 关系型数据库

--M:

一个class对应一个表

每个属性都是表中的一个字段，是类属性

pip3 install sqlalchemy

CREATE DATABASE tarena DEFAULT CHARSET 'utf8';

----------------------------------------------------------------------------------------

创建表

格式:

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

1.创建连接到数据库的引擎

engine = create\_engine(

'mysql+pymysql://用户名:密码@数据库地址/库?charset=utf8',

encoding='utf8',

#echo=True #表示将日志输出到终端屏幕,默认为False

2.创建ORM映射，生成ORM映射所需的基类

Base = declarative\_base()

3.自定义映射类型

#在数据库中创建表，如果库中已有同名的表，将不会创建

from sqlalchemy import Column, String, Integer, ForeignKey

(需什么类型就导入什么类型)

class 表名(Base) # 必须继承于Base

\_\_tablename\_\_ = 'employees'

字段名 = Column(字段的约束)

class 表名(Base)

\_\_tablename\_\_ = 'employees'

字段名 = Column(字段的约束)

.......

字段的约束:

ForeignKey('departments.dep\_id')

primary\_key=True

nullable非空约束，unique唯一性约束

1. 通过将sessionmaker与数据库引擎绑定，创建会话类Session

-- ORM访问数据库的句柄被称作Session

from sqlalchemy.orm import sessionmaker

Session = sessionmaker(bind=engine)

如果在创建session前还未创建engine,操作如下

Session = sessionmaker()

Session.configure(bind=engine) //创建engine后执行

5.通过表的映射类,在数据库中创建表

**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 Base.metadata.create\_all(engine)

---------------------------------------------------------------------------------------------------------

查询(另外建表)

-- 通过作用于session的query()函数创建查询对象

通这order\_by()函数可以实现按指定字段排序

session.query(Departments).order\_by(Departments.dep\_id)

通过filter()函数实现结果过滤

session.query(Departments.dep\_name).filter(Departments.dep\_id==2)

query.filter(Employees.name.like(' %j '))

通过“切片”的方式,实现部分数据的提取

session.query(Departments,Departments.dep\_name)[2:5]

通过join()方法实现多表查询

使用:

from myorm import Session , Departments

session = Session()

for instance in session.query(Departments).order\_by(Departments.dep\_id):

print(instance.dep\_id, instance.dep\_name)

----------------------------------------------------------------------------------------------------------

多表查询

from 上面的文件名 import 属性, Session

qset10=

session.query(Employee.emp\_name,Department.dep\_name).join(Department)

for row in qset10:

print(row)

-----------------------------------------------------------------------------------------------------------

添加数据

from 上面的文件名 import 映射, Session

hr = 映射(dep\_id=1, dep\_name='hr') #创建映射的实例

.........

des = [hr,.......]

session = Session() #创建会话类实例

session.add(一条数据) #向实例绑定的数据库添加数据

session.add\_all(多条数据) #向实例绑定的数据库添加数据

session.commit() #将数据提交到实例对应数据库

session.close() #关闭session类

添加过的数据不要再添加

--------------------------------------------------------------------------------------------------------

修改/删除 数据

from 上面的文件名 import 属性, Session

# 先查找数据

session = Session()

qset12 = session.query(Departments).filter(Departments.dep\_name=='财务部')

hr = qset12.one()

#修改/删除 数据

hr.dep\_name = '人力财务部' <--修改, 只要把映射的实体类对象重新赋值保存即可

session.delete(hr) <--删除, 记录与修改类似，先找到记录再执行删除

session.commit()

-----------------------------------------------------------------------------------------------------