简介

1. SQLAlchemy是用Python编程语言开发的一个开源项目。它提供了SQL工具包和ORM（对象关系映射）工具，使用MIT许可证发行。
2. SQLAlchemy最初在2006年2月发行，发行后便很快的成为Python社区中最广泛使用的ORM工具之一，丝毫不亚于Django自带的ORM框架。
3. SQLAlchemy采用简单的Python语言，提供高效和高性能的数据库访问，实现了完整的企业级持久模型。它的理念是，SQL数据库的量级和性能比对象集合重要，而对象集合的抽象又重要于表和行。

安装sqlalchemy

```
pip3 install sqlalchemy
pip3 install pymysql
```

本文使用MySQL作为数据库，使用pymysql作为驱动，因此需要安装pymysql

连接数据库

配置信息

在连接数据库前，需要使用到一些配置信息，然后把它们组合成满足以下条件的字符串：

dialect+driver://username:password@host:port/database

- dialect：数据库，如：sqlite、mysql、oracle等
- driver：数据库驱动，用于连接数据库的，本文使用pymysql
- username：用户名
- password：密码
- host：IP地址
- port：端口
- database：数据库

```
HOST = 'localhost'
PORT = 3306
USERNAME = 'root'
PASSWORD = '123456'
DB = 'myclass'
DB_URI = f'mysql+pymysql://{USERNAME}:{PASSWORD}@{HOST}:{PORT}/{DB}'
# dialect + driver://username:passwor@host:port/database
```

建议将配置信息放到你的配置文件中，如config.py

创建引擎并连接数据库

```
from sqlalchemy import create_engine
from config import DB_URI
engine = create_engine(DB_URI)  # 创建引擎
conn = engine.connect()  # 连接
result = conn.execute('SELECT 1')  # 执行SQL
print(result.fetchone())
conn.close()  # 关闭连接
```

创建ORM模型并映射到数据库中

```
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.orm import sessionmaker
from config import DB_URI

engine = create_engine(DB_URI)
Base = declarative_base(engine)  # SQLORM基类
session = sessionmaker(engine)()  # 构建session对象


class Student(Base):
    __tablename__ = 'student'  # 表名
    id = Column(Integer, primary_key=True, autoincrement=True)
    name = Column(String(50))
    age = Column(Integer)
    sex = Column(String(10))


Base.metadata.create_all()  # 将模型映射到数据库中
```

执行上面代码，将会在数据库中生成对应的映射表student。

新增数据

创建表后，接下来我们要添加数据，代码如下：

```
student = Student(name='Tony', age=18, sex='male')  # 创建一个student对象
session.add(student)  # 添加到session
session.commit()  # 提交到数据库
# 也可以批量添加数据：

session.add_all([
Student(name='Jane', age=16, sex='female'),
Student(name='Ben', age=20, sex='male')
```

```
])
session.commit()
```

查询数据

sqlalchemy提供了query()方法来查询数据

获取所有数据

```
item_list = session.query(Student).all()
print(item_list)
for item in item_list:
    print(item.name, item.age)
```

执行结果如下

```
[<mymodel.Student object at 0x000002A0E6A38088>, <mymodel.Student object at 0x000002A0E6A
Tony 18
Jane 16
Ben 20
```

查询得到的item_list是一个包含多个Student对象的列表

指定查询列

```
item_list = session.query(Student.name).all()
print(item_list)

# [('Tony',), ('Jane',), ('Ben',)]
```

获取返回数据的第一行

```
item = session.query(Student.name).first()
print(item)

# ('Tony',)
```

使用filter()方法进行筛选过滤

```
item_list = session.query(Student.name).filter(Student.age >= 18).all()
print(item_list)

# [('Tony',), ('Ben',)]
```

使用order_by()进行排序

```
item_list = session.query(Student.name, Student.age).order_by(Student.age.desc()).all()
print(item_list)
# [('Ben', 20), ('Tony', 18), ('Jane', 16)]
```

多个查询条件（and和or）

```
# 默认为and, 在filter()中用,分隔多个条件表示and
item_list = session.query(Student.name, Student.age, Student.sex).filter(
Student.age >= 10, Student.sex == 'female'
).all()
print(item_list)  # [('Jane', 16, 'female')]



from sqlalchemy import or_

# 使用or_连接多个条件
item_list = session.query(Student.name, Student.age, Student.sex).filter(
or_(Student.age >= 20, Student.sex == 'female')
).all()
print(item_list)  # [('Jane', 16, 'female'), ('Ben', 20, 'male')]









# equal/like/in

# 等于
item_list = session.query(Student.name, Student.age, Student.sex).filter(
Student.age == 18
).all()
print(item_list)  # [('Tony', 18, 'male')]
# 不等于
item_list = session.query(Student.name, Student.age, Student.sex).filter(
Student.age != 18
).all()
print(item_list)  # [('Jane', 16, 'female'), ('Ben', 20, 'male')]

# like
```

```
item_list = session.query(Student.name, Student.age, Student.sex).filter(
Student.name.like('%To%')
).all()
print(item_list)  # [('Tony', 18, 'male')]

  # in
 item_list = session.query(Student.name, Student.age, Student.sex).filter(
Student.age.in_([16, 20])
).all()
 print(item_list) # [('Jane', 16, 'female'), ('Ben', 20, 'male')]
```

count计算个数

```
count = session.query(Student).count()
print(count)  # 3
```

切片

```
item_list = session.query(Student.name).all()[:2]
print(item_list)  # [('Tony',), ('Jane',)]
```

修改数据

修改数据可以使用update()方法，update完成后记得执行session.commit()

```
# 修改Tony的age为22
session.query(Student).filter(Student.name == 'Tony').update({'age': 22})
session.commit()

item = session.query(Student.name, Student.age).filter(Student.name == 'Tony').first()
print(item)
```

执行结果如下

```
('Tony', 22)
```

删除数据

删除数据使用delete()方法，同样也需要执行session.commit()提交事务

```
# 删除名称为Ben的数据
session.query(Student).filter(Student.name == 'Ben').delete()
session.commit()
```

```
item_list = session.query(Student.name, Student.age).all()
print(item_list)
```

执行结果如下

```
[('Tony', 22), ('Jane', 16)]
```