



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình viên Python

Bài 2: *SQLAlchemy*

Phòng LT & Mạng

<http://csc.edu.vn/lap-trinh-va-csdl>

Nội dung

1. Giới thiệu SQLAlchemy
2. SQLAlchemy

Giới thiệu SQLAlchemy

- ❑ Ở module 2, ta đã sử dụng SQL thô trong ứng dụng web với Flask để thực hiện các công việc CRUD làm việc với CSDL SQLite. Ở module 3, ta làm quen với một cách làm việc mới, đó là sử dụng SQLAlchemy.
- ❑ SQLAlchemy là bộ công cụ của Python với một OR Mapper mạnh mẽ cho phép nhà phát triển ứng dụng làm việc hiệu quả và linh hoạt hơn với SQL.
- ❑ Flask-SQLAlchemy là Flask extension có hỗ trợ SQLAlchemy cho ứng dụng viết bằng Flask.

Giới thiệu SQLAlchemy

❑ Giới thiệu ORM (Object Relation Mapping)

- Hầu hết các nền tảng ngôn ngữ lập trình đều là hướng đối tượng. Trong khi đó, dữ liệu trong RDBMS (Relational database management system) server lại lưu trữ dữ liệu theo dạng bảng. Object relation mapping là một kỹ thuật ánh xạ các tham số đối tượng (object parameter) tới cấu trúc bảng phía dưới.
- Một ORM API cung cấp các phương thức để thực hiện các công việc CRUD mà không cần phải viết các lệnh SQL thô.

Giới thiệu SQLAlchemy

- ❑ **Có 3 thành phần quan trọng nhất trong việc viết code SQLAlchemy:**
 - Một Table đại diện cho một table trong CSDL.
 - Một mapper ánh xạ một class tới một table trong a database.
 - Một class object định nghĩa cách một record trong CSDL ánh xạ tới một object.
- ❑ **Thay vì phải viết code cho Table, mapper và class object ở những nơi khác nhau thì khai báo của SQLAlchemy cho phép một Table, một mapper và một class object được định nghĩa một lần trong một khai báo class.**

Nội dung

1. Giới thiệu SQLAlchemy
2. SQLAlchemy

SQLAlchemy

❑ Để có thể làm việc với SQLAlchemy

- Cần cài đặt thư viện SQLAlchemy: `pip install flask-sqlalchemy`

SQLAlchemy

❑ Tạo các class tương ứng với các table trong CSDL

● Các bước thực hiện

- Bước 1: Trong thư mục Xu_ly tạo tập tin Xu_ly_Model.py để chứa các class cần thiết cho việc tạo và làm việc với CSDL
- Bước 2: Trong Xu_ly_Model.py
 - Chèn các thư viện cần thiết ở đầu file

```
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import Column, ForeignKey, Integer,
String, Float
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship
from sqlalchemy import create_engine
```


SQLAlchemy

■ Bước 2 (tt):

- Khai báo biến engine để lưu CSDL trong thư mục Du_lieu của ứng dụng bằng create_engine()

- Ví dụ:

```
engine =  
create_engine('sqlite:///Du_lieu/ql_truong_tieu_hoc  
.db')
```

- Khai báo biến Base = declarative_base(), là một factory function cấu trúc một base class cho việc định nghĩa các class khai báo

```
Base = declarative_base()
```

SQLAlchemy

- Bước 3: Tạo các SQLAlchemy class với các thuộc tính đối tượng làm tham số. Nó chứa các phương thức hỗ trợ cho các thao tác ORM.
- Cột (Column trong bảng có thể có các kiểu dữ liệu như String, Integer, Float, Date, Boolean, Numeric...)
- Ví dụ: Tạo class Lop (sẽ ánh xạ tới table 'Lop')

```
class Lop(Base):  
    __tablename__ = 'Lop'  
    Ma_so = Column(String(50), nullable=False,  
                    primary_key=True)  
    Ten = Column(String(200), nullable=False)  
    def __str__(self):  
        return self.Ten
```

SQLAlchemy

- Với những bảng có mối quan hệ cha-con với nhau thì sử dụng ForeignKey() và relationship() để thiết lập mối quan hệ.
- Ví dụ: Tạo class Hoc_sinh (sẽ ánh xạ tới table 'Hoc_sinh') có khóa ngoại là Ma_so của Lop







```
class Hoc_sinh(Base):
    __tablename__ = 'Hoc_sinh'
    Ma_so = Column(String(50), nullable=False,
                    primary_key= True)
    Ho_ten = Column(String(200), nullable=False)
    Ten_dang_nhap = Column(String(50), nullable=False)
    Mat_khau = Column(String(50), nullable=False)
    Ma_so_Lop = Column(String(50),
                      ForeignKey('Lop.Ma_so'))
    lop = relationship(Lop, backref="hoc_sinh")

    def __str__(self):
        return self.Ho_ten
```

- Bước 4: Thực hiện việc ánh xạ các class này vào CSDL để tạo bảng tương ứng.

```
Base.metadata.create_all(engine)
```

- Xem kết quả: nếu chưa chính xác thì điều chỉnh và chạy lại.

| Name | Type | Schema |
|--|------|---|
| ▼  Tables (5) | | |
| >  Giang_day | | CREATE TABLE "Giang_day" (id INTEGER NOT NULL, "Ma_so_Lo |
| >  Giao_vien | | CREATE TABLE "Giao_vien" ("Ma_so" VARCHAR(50) NOT NULL, |
| >  Hoc_sinh | | CREATE TABLE "Hoc_sinh" ("Ma_so" VARCHAR(50) NOT NULL, |
| >  Lop | | CREATE TABLE "Lop" ("Ma_so" VARCHAR(50) NOT NULL, "Ten" |
| >  Nhan_xet | | CREATE TABLE "Nhan_xet" (id INTEGER NOT NULL, "Noi_dung" |

SQLAlchemy

❑ Thực hiện các thao tác trên dữ liệu – CRUD

- Trong trang thực hiện các thao tác này cần bổ sung thư viện sessionmaker:

```
from sqlalchemy.orm import sessionmaker
```

- Thư viện này sẽ giúp ta thực hiện:
 - `session.add(model object)` – thêm một record vào bảng được ánh xạ
 - `session.delete(model object)` – xóa record trong bảng
 - `session.query(class object).all()` – lấy tất cả các record từ table (giống như dùng SELECT query).

SQLAlchemy

- Khi cần dùng session cho database cần khai báo và khởi tạo session:

```
Base.metadata.bind = engine
DBSession = sessionmaker(bind=engine)
session = DBSession()
```

- Thêm dữ liệu

- dùng `session.add()`, sau khi thêm cần dùng `session.commit()` để hoàn thành công việc
- Ví dụ: Thêm các lớp học trong danh sách lớp học vào bảng `Lop` trong `CSDL`

```
Danh_sach_lop_hoc = Doc_Danh_sach(Thu_muc_Lop_hoc)
# Them lop hoc vao CSDL
for lop in Danh_sach_lop_hoc:
    lop_hoc = Lop(Ma_so=lop["Ma_so"], Ten= lop["Ten"])
    session.add(lop_hoc)
    session.commit()
```

- Đọc dữ liệu:

- dùng `session.query(object class).all()`: lấy tất cả các mẫu tin
 - Ví dụ: đọc và hiển thị các lớp đang có trong bảng Lop

```
ds_lop = session.query(Lop).all()
danh_sach_lop = []
for lop in ds_lop:
    l = {"Ma_so": lop.Ma_so, "Ten": lop.Ten}
    danh_sach_lop.append(l)
print("Số lớp:", len(danh_sach_lop))
print(danh_sach_lop)
```


- Lọc dữ liệu:

- `session.query(object class).filter(Điều_kiện_lọc).one()` hoặc `.all()` hoặc `.first()` để lấy một mẫu tin/ tất cả các mẫu tin/ mẫu tin đầu tiên thỏa điều kiện lọc
 - Ví dụ: lọc danh sách các học sinh trong lớp có mã số là 'LH1'

```
ds_hs_lop_LH_1 = session.query(Hoc_sinh).filter(Hoc_sinh.lop ==
session.query(Lop).filter(Lop.Ma_so == "LH_1").one()).all()
danh_sach_hoc_sinh = []
for hs in ds_hs_lop_LH_1:
    hs1 = {"Ma_so": hs.Ma_so, "Ho_ten":hs.Ho_ten, "Ten_dang_nhap":
        hs.Ten_dang_nhap, "Mat_khau": hs.Mat_khau, "Ma_so_lop":
        hs.lop.Ma_so }
    danh_sach_hoc_sinh.append(hs1)
print("Số hs trong lớp:", len(danh_sach_hoc_sinh))
for hs in danh_sach_hoc_sinh:
    print(hs)
```

SQLAlchemy

- Xóa dữ liệu

- dùng `session.delete()`, sau khi xóa cần dùng `session.commit()`

- Ví dụ: tạo lớp học có mã số 'LH5', tên 'Lớp học 5', sau khi tạo xong thì xóa lớp học này

```
# tạo lớp học
```

```
lop_hoc = Lop(Ma_so='LH_5', Ten= "Lớp học 5")  
session.add(lop_hoc)  
session.commit()
```

```
# xóa lớp học
```

```
session.delete(session.query(Lop).filter(Lop.Ma_so ==  
"LH_5").one())  
session.commit()
```

