7회차 과제 - DB

Date	@2024년 8월 1일 오후 11:59
Tag	과제

SQLAlchemy를 활용하여 ORM을 실습해보았습니다.

환경 설정

1. Docker Desktop 켜기
2. 터미널에서 docker-compose.yml 파일이 있는 디렉토리로 이동 후 컨테이너 생성 및 시작

```
docker-compose up -d
```

▼ cf. docker-compose.yml

```
version: "3"
services:
  db:
    image: mysql:8.4
    container_name: ybigta-mysql
    restart: always
    ports:
      - 3310:3306
    environment:
      MYSQL_DATABASE: ybigta
      MYSQL_ROOT_PASSWORD: test1234
      TZ: Asia/Seoul
    volumes:
      - ./db/mysql/data:/var/lib/mysql
      - ./db/mysql/init:/docker-entrypoint-initdb.d
```

- docker compose: 단일 서버에서 여러 개의 컨테이너를 하나의 서비스로 정의해 컨테이너의 묶음으로 관리할 수 있는 작업 환경을 제공하는 관리 도구
- docker compose 파일 버전 및 서비스 정의

```
version: "3"
services:
  db:
```

  - `version: "3"`: Docker Compose 파일의 버전
  - `Services` 내에 여러 개의 서비스를 정의할 수 있음. 여기서는 `db` 라는 하나의 서비스만 정의

- db 서비스 설정

```
    image: mysql:8.4
    container_name: ybigta-mysql
    restart: always
    ports:
      - 3310:3306
    environment:
      MYSQL_DATABASE: ybigta
      MYSQL_ROOT_PASSWORD: test1234
      TZ: Asia/Seoul
    volumes:
      - ./db/mysql/data:/var/lib/mysql
      - ./db/mysql/init:/docker-entrypoint-initdb.d
```

  - `ports: - 3310:3306`: 호스트의 3310 포트를 컨테이너의 3306 포트에 매핑 (MySQL은 기본적으로 3306 포트 사용)
  - `volumes`
    - `./db/mysql/data:/var/lib/mysql`
      - 호스트의 `./db/mysql/data` 디렉토리를 컨테이너의 `/var/lib/mysql` 디렉토리에 마운트
      - MySQL 데이터 저장
    - `./db/mysql/init:/docker-entrypoint-initdb.d`

        - 호스트의 `./db/mysql/init` 디렉토리를 컨테이너의 `/docker-entrypoint-initdb.d` 디렉토리에 마운트
        - 컨테이너가 초기화될 때 실행할 스크립트 저장
3. VScode Extension 설치
    - SQLTools
    - SQLTools MySQL/MariaDB/TiDB
4. Python 가상환경 생성 후 필요한 라이브러리 설치
    - sqlalchemy
    - pymysql
5. sql_exercise.sql을 실행하여 실습에 사용할 테이블들 생성

디렉토리 구조

```
├── db
├── .env
├── config.py
├── crud.py
├── database.py
├── docker-compose.yml
├── main.py
├── models.py
├── requirements.txt
└── sql_exercise.sql
```

- .env
    - 환경변수 정의
- config.py
    - 환경 변수를 로드 및 애플리케이션 설정 관리
    - pydantic의 Basesettings
        - pydantic: 파이썬의 데이터 검증(validation) 라이브러리
        - Basesettings: Basesettings를 상속한 클래스를 통해서 설정 데이터들을 검증하고, 환경 변수를 설정 값으로 사용할 수 있도록 함

- database.py
  - 데이터베이스 연결 및 세션 관리
  - engine, session
    - engine: 데이터베이스와의 연결을 관리하는 객체
    - cf. engine을 생성할 때 사용하는 URL의 구성
      - mysql+pymysql://[user]:[password]@[localhost]/[dbname]
        - mysql: SQLAlchemy가 연결할 데이터베이스의 종류
        - pymysql: MySQL 데이터베이스에 연결하기 위해 SQLAlchemy가 사용할 Python 드라이버
    - session: 데이터베이스 트랜잭션을 관리하고, ORM 매핑된 객체를 데이터베이스에 저장하거나 조회하는 작업을 수행하는 객체
- model.py
  - 데이터베이스 테이블 구조 정의
  - declarative_base: ORM 모델 클래스의 베이스 클래스 생성 → 베이스 클래스는 데이터베이스 테이블과의 매핑 정보를 저장함으로써 ORM 기능을 활용할 수 있게 함
- crud.py
  - 데이터베이스의 CRUD (Create, Read, Update, Delete) 작업을 수행하는 함수들 정의

main.py 실행 시 출력 내용