DATA
SCIENCE LAB

빅데이터를 지탱하는 기술

Week 5

FastAPI 심화

23.08.23 / 엄소은

CONTENTS

1. Database

Database ?

- 데이터의 집합
- 데이터베이스는 데이터의 관리를 도와준다
- 데이터베이스는 데이터를 찾아오고, 저장하고, 수정하는 것을 기준으로 정리되어 있다
- DBMS(DataBase Management Systems) : 데이터베이스를 운영하고 관리하는 소프트웨어
- DBMS 의 종류에는 여러가지가 있다 -> MySQL, Oracle, SQL Server, MariaDB
- DBMS 의 유형에는 관계형(relational) 이 있다.

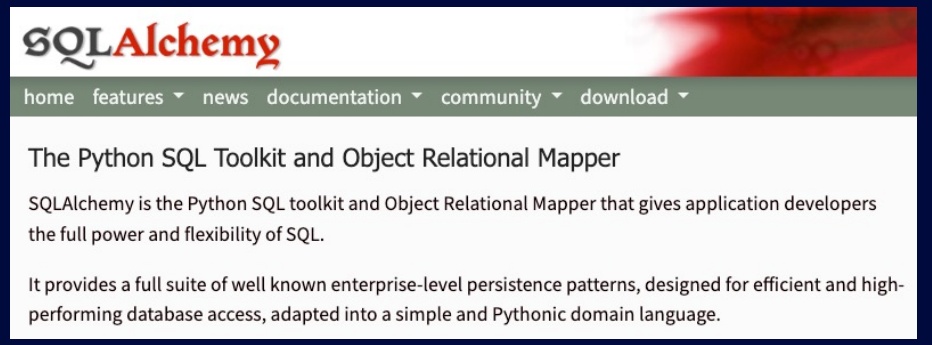
1. Database

SQL ?

- SQL = Structured Query Language
- SQL 은 relational database 를 다루는 언어이다
- SQL 은 database record 를 이용해 여러가지 일을 할 수 있다 (CRUD - Create, Read, Update, Delete)

1. Database

python 으로 SQL 을 사용할 수 있다 ?



<SQLAlchemy>

- ORM(Object Relational Mapping) 을 통해 객체-관계형데이터베이스 매핑
- 객체의 Class 와 관계형 데이터베이스의 Table 을 자동으로 매핑 해줌
- RESTful API 의 한 특징이었던 'statelessness' (Server 는 Client 가 보낸 request 에 대해 기억하지 못한다) 와 반대로 생성한 request 가 종료되도 데이터가 영구적으로 남아있도록 함

tutorial

1. Database

Declare Models

- 테이블의 틀을 만든다 → 객체의 Class 로 !

```
You, 18시간 전 | 1 author (You)
from database import Base
from sqlalchemy import Column, Integer, String, Boolean

You, 18시간 전 | 1 author (You)
class Results(Base):
    __tablename__ = 'Text_Sentiment_Results'

    id = Column(Integer, primary_key=True,index=True)
    sentence = Column(String)
    label_1 = Column(String)
    score_1 = Column(Integer)
    label_2 = Column(String)
    score_2 = Column(Integer)
```

```
>>> from typing import List
>>> from typing import Optional
>>> from sqlalchemy import ForeignKey
>>> from sqlalchemy import String
>>> from sqlalchemy.orm import DeclarativeBase
>>> from sqlalchemy.orm import Mapped
>>> from sqlalchemy.orm import mapped_column
>>> from sqlalchemy.orm import relationship

>>> class Base(DeclarativeBase):
...     pass

>>> class User(Base):
...     __tablename__ = "user_account"
...
...     id: Mapped[int] = mapped_column(primary_key=True)
...     name: Mapped[str] = mapped_column(String(30))
...     fullname: Mapped[Optional[str]]
...
...     addresses: Mapped[List["Address"]] = relationship(
...         back_populates="user", cascade="all, delete-orphan"
...     )
...
...     def __repr__(self) -> str:
...         return f"User(id={self.id!r}, name={self.name!r}, fullname={self.fullname!r})"

>>> class Address(Base):
...     __tablename__ = "address"
...
...     id: Mapped[int] = mapped_column(primary_key=True)
...     email_address: Mapped[str]
...     user_id: Mapped[int] = mapped_column(ForeignKey("user_account.id"))
...
...     user: Mapped["User"] = relationship(back_populates="addresses")
...
...     def __repr__(self) -> str:
...         return f"Address(id={self.id!r}, email_address={self.email_address!r})"
```

Annotated Example

1. Database

Create an Engine

- Database connection 을 연결해주는 engine 을 만든다
- 여기서는 sqlite 사용

```python
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base

SQLALCEMY_DATABASE_URL = 'sqlite:///./results.db'

engine = create_engine(SQLALCEMY_DATABASE_URL,connect_args={'check_same_thread':False})

SessionLocal = sessionmaker(autocommit=False,autoflush=False, bind=engine)

Base = declarative_base()
```

1. Database

FastAPI 와 연동하기

```python
@app.get("/",status_code = status.HTTP_200_OK)
async def read_all(db:db_dependency):
    return db.query(Results).all()
```

```python
@app.delete('/sentiment/{id}',status_code=status.HTTP_204_NO_CONTENT)
async def delete_text(db:db_dependency,id : int = Path(gt=0)):
    text_model = db.query(Results).filter(Results.id == id).first()
    if text_model is None:
        raise HTTPException(status_code = 404, detail = 'ID not found')
    db.query(Results).filter(Results.id == id).delete()
    db.commit()
```

```python
@app.post('/sentiment',status_code = status.HTTP_201_CREATED)
async def create_text_sentiment(db:db_dependency,text_request:SentimentRequest):
    if len(text_request.text) <= 3:
        raise HTTPException(status_code = 201, detail = "Please put in valid text")

    label_1, label_2 ,score_1, score_2 = sentiment(text_request.text)
    text_result = SentimentResult(sentence = text_request.text,
                                  label_1 = label_1,
                                  score_1 = score_1,
                                  label_2 = label_2,
                                  score_2 = score_2
    )
    text_model = Results(**text_result.dict())

    db.add(text_model)
    db.commit()
```

1. Database

Users Table 과 Todos Table 관계

```python
class Users(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True,index=True)
    email = Column(String,unique=True)
    username = Column(String,unique=True)
    first_name = Column(String)
    last_name = Column(String)
    hashed_password = Column(String)
    is_active = Column(Boolean,default=True)
    role = Column(String)
```

```python
class Todos(Base):
    __tablename__ = 'todos'

    id = Column(Integer, primary_key=True,index=True)
    title = Column(String)
    description = Column(String)
    priority = Column(Integer)
    complete = Column(Boolean,default=False)
    owner_id = Column(Integer,ForeignKey("users.id"))
```

여기에 User 정보 저장

owner_id = ForeignKey

2. Authenticate

User 만들기

```
You, 4시간 전 | 1 author (You)
class CreateUserRequest(BaseModel):
    email : str
    username : str
    first_name : str
    last_name : str
    password : str
    role : str

You, 4시간 전 | 1 author (You)
class Token(BaseModel):
    access_token : str
    token_type : str
```

JWT(Json Web Token) 을 이용하여 password 를 암호화(hashing) 한다

코드 보면서 실습 !

3. Reference

- Udemy, FastAPI - The complete course 2023