Restful API And Microservices with Python

Day 10

Day 10 - Overview

- SQLAlchemy ORM integration
- Updating User and Address Model to extend SQLAlchemy Models
- Automatically creating tables using create_all() method
- SQLAlchemy based queries to perform CRUD operation
- List Relationship Items User Address one-to-many relationship

Prerequisite

- VM with windows OS
- Python 3.8 or >
- Visual Studio Code - Code Editor
- Postman
- GIT

https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf

- Docker

Sync your fork for Day 10 activities

- Follow the below document to sync your fork and update local repository.

  https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf

Local Setup for Day 10

- Open a separate powershell and start the docker containers

  cd *C:\workspace\flask-microservices-training\day10\user-management-service*

  docker-compose up

- Navigate to the below folder

  C:\workspace\flask-microservices-training\day10\user-management-service

- Create a virtual environment and activate it

  python -m venv venv

  .\venv\Scripts\activate

- Install the dependencies and start server

  pip install -r requirements.txt

SQLAlchemy ORM

- SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
- It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

Integrating with SQLAlchemy

- Flask-SQLAlchemy-> Library for Flask SQLAlchemy integration
- Psycopg2-binary -> Library for Python and Postgres integration
- Wrapping the flask application with SQLAlchemy
  - db = SQLAlchemy(app)
- Initializing the database on startup
  - db.create_all()

Updating the User Model

```
from app import db
from sqlalchemy.orm import relationship

class User(db.Model):
    __tablename__ = 'UMS_USER'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50))
    email = db.Column(db.String(50))
    age = db.Column(db.Integer)
    password = db.Column(db.String(100))
    role = db.Column(db.String(50))
    addresses = relationship('Address', cascade='all, delete')
```

Database Query using SQLAlchemy

- Fetching All the records of a model/table
  - *db.session.query(User).all()* or *User.query.all()*
- Fetching a single record based on ID
  - *User.query.get(id)*
- Fetching a single record based on email field
  - *User.query.find_by(email='saurav@gmail.com').first()*
- Creating a new User
  - db.session.add(user)
  - db.session.commit()
- Updating an existing user
  - Update the user object and then call below
  - *db.session.commit()*
- Deleting a record
  - *db.session.delete(user)*

Code Walk Through

Q and A