**[Que-43] - Describe the process of connecting a Flask app to SQLite database using SQLAlchemy**

Connecting a Flask application to an SQLite database using SQLAlchemy involves several steps. Here’s a comprehensive guide to help you through the process:

### **Step 1: Install Flask and SQLAlchemy**

First, ensure you have Flask and SQLAlchemy installed. You can install them using pip:

pip install flask flask-sqlalchemy

### **Step 2: Set Up the Flask Application**

Create a new directory for your project. Inside this directory, create the main Flask application file, app.py.

### **Step 3: Configure the Flask Application**

In app.py, configure the Flask application to use SQLAlchemy and SQLite:

from flask import Flask  
from flask\_sqlalchemy import SQLAlchemy  
  
app = Flask(\_\_name\_\_)  
  
# Configure the SQLite database, relative to the app instance folder  
app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///your\_database.db'  
app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  
  
# Create the SQLAlchemy db instance  
db = SQLAlchemy(app)  
  
# Define a simple model for demonstration  
class User(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 username = db.Column(db.String(80), unique=True, nullable=False)  
 email = db.Column(db.String(120), unique=True, nullable=False)  
  
 def \_\_repr\_\_(self):  
 return f'<User {self.username}>'  
  
# Route to display all users  
@app.route('/users')  
def users():  
 all\_users = User.query.all()  
 return {'users': [str(user) for user in all\_users]}  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 db.create\_all() # Create database and tables  
 app.run(debug=True)

### **Step 4: Create the Database and Tables**

In the above script, the db.create\_all() method is used to create the SQLite database and the defined tables. This should be done only once, usually before the first run of the application.

### **Step 5: Run the Flask Application**

Open a terminal, navigate to the directory containing app.py, and run the Flask application:

python app.py

This will start the Flask development server and create the SQLite database file (your\_database.db) in the same directory as app.py.

### **Step 6: Interact with the Database**

Now that the database and tables are set up, you can interact with the database using SQLAlchemy. Here’s how to add, query, update, and delete records:

#### **Adding Records**

To add records, you can use the Flask shell or integrate the logic into your routes.

**Using Flask Shell:**

flask shell

Within the Flask shell, you can interact with your database:

from app import db, User  
  
# Create new users  
user1 = User(username='john', [email='john@example.com](mailto:email='john@example.com)')  
user2 = User(username='susan', [email='susan@example.com](mailto:email='susan@example.com)')  
  
# Add users to the session  
db.session.add(user1)  
db.session.add(user2)  
  
# Commit the session to write changes to the database  
db.session.commit()

#### **Querying Records**

You can query the database using SQLAlchemy's query interface:

# Query all users  
users = User.query.all()  
print(users)  
  
# Query a specific user by username  
user = User.query.filter\_by(username='john').first()  
print(user)

#### **Updating Records**

To update records, modify the object and commit the session:

# Query the user to update  
user = User.query.filter\_by(username='john').first()  
user.email = '[newjohn@example.com](mailto:newjohn@example.com)'  
  
# Commit the session to save changes  
db.session.commit()

#### **Deleting Records**

To delete records, use the delete method and commit the session:

# Query the user to delete  
user = User.query.filter\_by(username='john').first()  
  
# Delete the user  
db.session.delete(user)  
  
# Commit the session to apply changes  
db.session.commit()

### **Step 7: Organizing Larger Applications**

As your application grows, it's a good practice to organize it into multiple modules. Here’s a suggested structure:

/myapp  
 /app  
 /templates  
 /static  
 \_\_init\_\_.py  
 models.py  
 routes.py  
 config.py  
 run.py

**\_\_init\_\_.py**:

from flask import Flask  
from flask\_sqlalchemy import SQLAlchemy  
  
db = SQLAlchemy()  
  
def create\_app():  
 app = Flask(\_\_name\_\_)  
 app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///your\_database.db'  
 app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  
   
 db.init\_app(app)  
   
 from .routes import main  
 app.register\_blueprint(main)  
   
 with app.app\_context():  
 db.create\_all()  
   
 return app

**models.py**:

from . import db  
  
class User(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 username = db.Column(db.String(80), unique=True, nullable=False)  
 email = db.Column(db.String(120), unique=True, nullable=False)  
  
 def \_\_repr\_\_(self):  
 return f'<User {self.username}>'

**routes.py**:

from flask import Blueprint  
from .models import User  
  
main = Blueprint('main', \_\_name\_\_)  
  
@main.route('/users')  
def users():  
 all\_users = User.query.all()  
 return {'users': [str(user) for user in all\_users]}

**run.py**:

from app import create\_app  
  
app = create\_app()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)