Get unlimited access Open in app

Rohan Goel Follow
Jul 4, 2020 · 4 min read · Listen

Save

Flask with SQLAlchemy Database

> *Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.*

In this article, we will be making a very basic Flask-App that accepts the User Details like Name and Email ID from the user, stores that data into the SQLAlchemy Database, and then fetch and displays all the Database data into an HTML Table.

Let's get Started

> *Installation*

- Create and Activate your python virtual environment and install these following libraries,

```
pip install flask
pip install Flask-SQLAlchemy
```

> *Build Project Structure*

- Let's first create our Project Structure
- Make a Project Folder named "MyApp" and create a new "app.py" file inside it, also create a folder named "templates" inside it

Get unlimited access Open in app

```
MyApp/
     |- templates
        |- index.html
     |- app.py
```

HTML Form

- Let's first create an HTML form where a user can input his or her Name and Email.
- Go into your "index.html" file and add the following code inside it,

```
<!DOCTYPE html>
<html lang="en">

<head>
                <title> MyApp </title>
</head>

<body>

        <!-- Its the form where a user can submit his or her name and email -->

        <form class="form-inline" action="" method="POST">
                <label>Name</label>
                <input type="text" id="name" placeholder="Enter Name" name="name">
                <label>Email</label>
                <input type="email" id="email" placeholder="Enter Email Id" name="email">
                <button type="submit">Submit</button>
        </form>


        <!-- The table where all the data from the DB will be appeared -->

        <table class="data" border="1" style="width: 100%; margin-top: 20px; text-align: center;">
                <tr>
                        <th colspan="3">User Data</th>
                </tr>
                <tr>
                        <td>Sr.No.</td>
                <td>Name</td>
                <td>Email</td>
                </tr>

                <!-- The user_data is a variable containing all the user data from the DB. -->

                {% for user in user_data %}
                <tr>
                        <td>{{user._id}}</td>
                        <td>{{user.name}}</td>
                        <td>{{user.email}}</td>
                </tr>
                {% endfor %}

        </table>
</body>
```

101.html hosted with ❤ by GitHub view raw

Get unlimited access Open in app

- Double click your “index.html” file and you will see we have created an HTML form where a user can input his or her details.
- The end part of this HTML code is the Python code where the user_data variable consists of all the data from the database that we will be passing from the backend part of Flask.

> *Flask Backend*

- Now let’s create our Flask backend part where we connect it to the “index.html” file such that on the submit button click we will receive all the input details here.
- Go into your “app.py” file and add the following code into it,

```python
from flask import Flask , render_template, jsonify, request, redirect, url_for, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

# Control will come here and then gets redirect to the index function
@app.route("/")
def home():
        return redirect(url_for('index'))

@app.route("/index", methods = ["GET", "POST"])
def index():
        if request.method == 'POST': # When a user clicks submit button it will come here.
                data = request.form # gets the data from the form in index.html file
                name = data["name"]
                email = data["email"]

                user_data = [{"name":name, "email":email}] # stores them into a dictionary

                return render_template("index.html", user_data = user_data) # passes user_data variable into the index.html file.

        return render_template("index.html")

if __name__=="__main__":
        app.run(debug=True)
```

102.py hosted with ❤ by GitHub view raw

Get unlimited access Open in app

the “index.html” file where it gets read in the for loop and displays as a row in the table.

Database

- Now, what we have to do is when we receive that data from the form, we have to first store that user entry into the Database and then fetch all the entries including it from the DB, store that into a variable and pass it into the “index.html” file.
- The first step is to Create a Database structure, so into your “app.py” file add the following code after the “app=Flask(__name__)” line, to make a database and defining the table structure inside it.

```
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///User.sqlite3'

db = SQLAlchemy(app)

class User(db.Model):
        # Defines the Table Name user
        __tablename__ = "user"

        # Makes three columns into the table id, name, email
        _id = db.Column(db.Integer, primary_key=True, autoincrement=True)
        name = db.Column(db.String(100), nullable=False)
        email = db.Column(db.String(100), nullable=False)

        # A constructor function where we will pass the name and email of a user and it gets add as a new entry in the table.
        def __init__(self, name, email):
                self.name = name
                self.email = email
```

103.py hosted with ❤ by GitHub view raw

You can r
get stori
to your in
Got it

- We created an SQLAlchemy object and stored that into the db variable.
- Then we defined a User class with three columns like id, name, and email along with a constructor where we can send the name and email of the user.
- If you want to use a different name in the table you can provide an optional first argument which is a string with the desired column name.
- Types of Columns that you can define are Integer, String, Float, etc.

Get unlimited access Open in app

`String(size)`	a string with a maximum length (optional in some databases, e.g. PostgreSQL)
`Text`	some longer unicode text
`DateTime`	date and time expressed as Python `datetime` object.
`Float`	stores floating point values
`Boolean`	stores a boolean value
`PickleType`	stores a pickled Python object
`LargeBinary`	stores large arbitrary binary data

Flask-SQLAlchemy Official Documentation

- Now in the second step, all we have to do is, create an object of the User class with parameters as user details and store that into the Database and then fetch all the results from it and store them into a variable,
- So let's first create an object of the User class with the user details and store them into the database, add the following code after you have got the name and email from the user,

```
1  new_data = User(name, email)
2  db.session.add(new_data)
3  db.session.commit()
```

104.py hosted with ❤ by GitHub 22 view raw

- This will add the new row into the table in name and email column(no need to pass the id value as that automatically gets incremented).
- Now its time to fetch all the data from the database, and do that using the following command,

```
user_data = User.query.all()
```

- There are many other commands if you want to query your Database based on some particular value, to find out check out this Flask-SQLAlchemy official documentation.

Get unlimited access Open in app

id	username	email
1	admin	admin@example.com
2	peter	peter@example.org
3	guest	guest@example.com

Retrieve a user by username:

```
>>> peter = User.query.filter_by(username='peter').first()
>>> peter.id
2
>>> peter.email
u'peter@example.org'
```

Same as above but for a non existing username gives *None*:

```
>>> missing = User.query.filter_by(username='missing').first()
>>> missing is None
True
```

Selecting a bunch of users by a more complex expression:

```
>>> User.query.filter(User.email.endswith('@example.com')).all()
[<User u'admin'>, <User u'guest'>]
```

Ordering users by something:

```
>>> User.query.order_by(User.username).all()
[<User u'admin'>, <User u'guest'>, <User u'peter'>]
```

Limiting users:

```
>>> User.query.limit(1).all()
[<User u'admin'>]
```

Getting user by primary key:

```
>>> User.query.get(1)
<User u'admin'>
```

From Flask-SQLAchemy Official Documentation

- And now all we have to is just simply pass that user_data variable into the “index.html” as we have done previously.
- Add the following line before the “app.run(debug=True)” line of code at the end, to make sure all your database tables are created before the app runs.

```
db.create_all()
```

- And we are DONE! Run your project from the terminal using,

```
python app.py
```

Full Code

```
from flask import Flask , render_template, jsonify, request, redirect, url_for, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
```

Get unlimited access Open in app

```
class User(db.Model):
        # Defines the Table Name user
        __tablename__ = "user"

        # Makes three columns into the table id, name, email
        _id = db.Column(db.Integer, primary_key=True, autoincrement=True)
        name = db.Column(db.String(100), nullable=False)
        email = db.Column(db.String(100), nullable=False)

        # A constructor function where we will pass the name and email of a user and it gets add as a new entry in the table.
        def __init__(self, name, email):
                self.name = name
                self.email = email

# Control will come here and then gets redirect to the index function
@app.route("/")
def home():
        return redirect(url_for('index'))


@app.route("/index", methods = ["GET", "POST"])
def index():
        if request.method == 'POST': # When a user clicks submit button it will come here.
                data = request.form # request the data from the form in index.html file
                name = data["name"]
                email = data["email"]

                new_data = User(name, email)
                db.session.add(new_data)
                db.session.commit()

                user_data = User.query.all()

                return render_template("index.html", user_data = user_data) # passes user_data variable into the index.html file.

        return render_template("index.html")


if __name__=="__main__":
        db.create_all()
        app.run(debug=True)
```

105.py hosted with ❤ by GitHub view raw

Get unlimited access Open in app

Conclusions

- So with this, we have created a very basic FLASK App where a user can input its details, it gets stored into the Database using SQLAlchemy and then all the fetched data is displayed into an HTML Table.
- Feel free to ask your queries or doubts in the comments section and in the upcoming article, I will show you how you can deploy this Flask App along with the Database on Heroku.
- Connect with me on LinkedIn.