How to Build a Web App using Flask and SQLite in Python

Python-based Flask is a microweb framework. Typically, a micro-framework has little to no dependencies on outside frameworks. Despite being a micro framework, practically everything may be developed when and as needed utilizing Python libraries and other dependencies. In this post, we'll develop a Flask application that collects user input in a form and shows it on an additional web page using SQLite in Python.

Package Required

Install flask to proceed with the Front End of the Web App.

```
pip install flask
pip install db-sqlite3
```

Steps to Build an App Using Flask and SQLite

Step 1: Create Virtual Environment

Step 2: Install the required modules inside Virtual Environment.

Step 3: Build a Front End of the Web App.

- **index.html**

The index.html file will contain two buttons, one button to check all the participant's lists (taken from the database). And the other button to create a new entry.

HTML

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <title>Flask and SQLite </title>
    </head>
    <body>
        <h1>Build Web App Using Flask and SQLite</h1>
        <button class="btn" type="button" onclick="window.location.href='{{ url_f
        <button class="btn" type="button" onclick="window.location.href='{{ url_f
    </body>
</html>
```

- join.html

In the join.html, create a simple form that takes Name, Email, City, Country and Phone as the input to store in the database. By the POST method, receive the form request of all the columns and commit the changes in the database after inserting the details in the table.

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Flask and SQLite </title>
    </head>
    <body>
        <form method="POST">
            <label>Enter Name:</label>
            <input type="name" name="name" placeholder="Enter your name" required
            <label>Enter Email:</label>
            <input type="email" name="email" placeholder="Enter your email" requi
            <label>Enter City:</label>
            <input type="name" name="city" placeholder="Enter your City name" req
            <label>Enter Country:</label>
            <input type="name" name="country" placeholder="Enter the Country name
            <label>Enter phone num:</label>
            <input type="name" name="phone" placeholder="Your Phone Number" requi
            <input type = "submit" value = "submit"/><br/>
        </form>
    </body>
</html>
```

- participants.html

Use table tag and assign the heading using <th> tag. To auto increment, the table row on the new entry, use a For loop jinja template. Inside For loop add <tr> and <td> tags.

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Flask and SQLite </title>
    </head>
    <style>
        table, th, td {
          border:1px solid black;
        }
        </style>
    <body>
        <table style="width:100%">
            <tr>
              <th>Name</th>
              <th>Email</th>
              <th>City</th>
              <th>Country</th>
              <th>Phone Number</th>
            </tr>
            {%for participant in data%}
               <tr>
                 <td>{{participant[0]}}</td>
                 <td>{{participant[1]}}</td>
                 <td>{{participant[2]}}</td>
                 <td>{{participant[3]}}</td>
                 <td>{{participant[4]}}</td>
                </tr>
            {%endfor%}
          </table>
    </body>
</html>
```

Step 4: Create app.py

Create a new file named app.py and build a Front End of the Web App by rendering HTML templates. From here we shall go function by function explanation as in points:

- To insert the data into the database, we first need to create a new database table. The column to be inserted in the database is Name, Email, City, Country, and Phone Number.
- The basic syntax to start with sqlite3 is to first connect to the database. sqlite3.connect("database.db") will create a new database. The next step is to create a new table, but it will first check if the table already exists or not.
- One button in the index.html prompts to the participant's list, and thus using the existing database select * from the table and display it using a Python template i.e., Jinja template to run through the loop within HTML. In the following code, we have created a table tag, inside the table tag for every new insertion in the database, we add a Loop Jinja Template to auto increment the new table row.
- In the participants function, we use select all columns from the table name, we use fetchall() method you retrieve the data.

Python3

```
from flask import Flask, render_template, request
import sqlite3

app = Flask(__name__)


@app.route('/')
@app.route('/home')
def index():
    return render_template('index.html')


connect = sqlite3.connect('database.db')
connect.execute(
    'CREATE TABLE IF NOT EXISTS PARTICIPANTS (name TEXT, \
    email TEXT, city TEXT, country TEXT, phone TEXT)')


@app.route('/join', methods=['GET', 'POST'])
def join():
```

```
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        city = request.form['city']
        country = request.form['country']
        phone = request.form['phone']

        with sqlite3.connect("database.db") as users:
            cursor = users.cursor()
            cursor.execute("INSERT INTO PARTICIPANTS \
            (name,email,city,country,phone) VALUES (?,?,?,?,?)",
                           (name, email, city, country, phone))
            users.commit()
        return render_template("index.html")
    else:
        return render_template('join.html')


@app.route('/participants')
def participants():
    connect = sqlite3.connect('database.db')
    cursor = connect.cursor()
    cursor.execute('SELECT * FROM PARTICIPANTS')

    data = cursor.fetchall()
    return render_template("participants.html", data=data)


if __name__ == '__main__':
    app.run(debug=False)
```

Output:

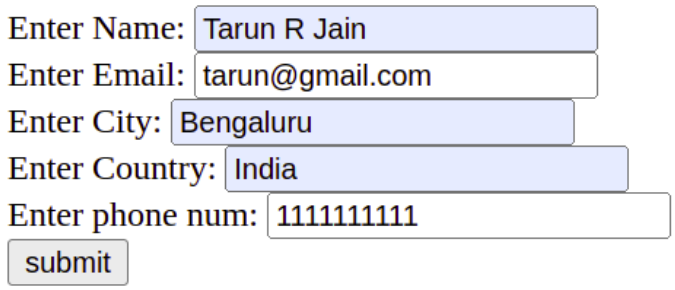
For route: http://127.0.0.1:5000/

Build Web App Using Flask and SQLite

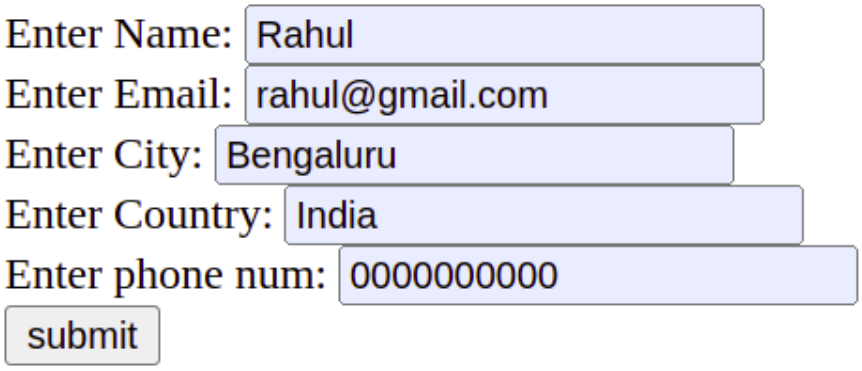
Fill form to get updates
Check participant list

For route: http://127.0.0.1:5000/join

Here we are adding two new data to the database.



data 1



data 2

For route: http://127.0.0.1:5000/participants

Name	Email	City	Country	Phone Number
Tarun R Jain	tarun@gmail.com	Bengaluru	India	1111111111
Rahul	rahul@gmail.com	Bengaluru	India	0000000000

Looking to dive into the world of programming or sharpen your Python skills? Our **Master Python: Complete Beginner to Advanced Course** is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and

expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!

3

Previous Article

Flask and Sqlalchemy Tutorial for Database

Next Article

Sending Data from a Flask app to MongoDB Database

Similar Reads

How to Build a Simple Android App with Flask Backend?

Flask is an API of Python that allows us to build up web applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's...

8 min read

Documenting Flask Endpoint using Flask-Autodoc

Documentation of endpoints is an essential task in web development and being able to apply it in different frameworks is always a utility. This article discusses...

4 min read

Minify HTML in Flask using Flask-Minify

Flask offers HTML rendering as output, it is usually desired that the output HTML should be concise and it serves the purpose as well. In this article, we would...

12 min read

How to use Flask-Session in Python Flask ?

Flask Session - Flask-Session is an extension for Flask that supports Server-side Session to your application.The Session is the time between the client logs in to...

4 min read

How to Integrate Flask-Admin and Flask-Login

In order to merge the admin and login pages, we can utilize a short form or any other login method that only requires the username and password. This is know...

8 min read

Flask URL Helper Function - Flask url_for()

In this article, we are going to learn about the flask url_for() function of the flask URL helper in Python. Flask is a straightforward, speedy, scalable library, used f...

11 min read

How to Use SQLite Viewer Flask

SQLite is a popular lightweight relational database management system that is used in a variety of applications. Flask is a popular web framework for building...

3 min read

Python | Build a REST API using Flask

Prerequisite: Introduction to Rest API REST stands for REpresentational State Transfer and is an architectural style used in modern web development. It define...

3 min read

Build Blog website using Flask

In this article, we'll explore how to build a dynamic blog website using Flask, a lightweight and versatile Python web framework. Flask provides developers wit...

15+ min read

Create a Weather app using Flask | Python

Prerequisite : Flask installation Flask is a lightweight framework written in Python. It is lightweight because it does not require particular tools or libraries...

2 min read

Article Tags : Python Technical Scripter Technical Scripter 2022

Practice Tags : python

Corporate & Communications Address:- A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305