How to serve static files in Flask

Last Updated : 23 Jan, 2023

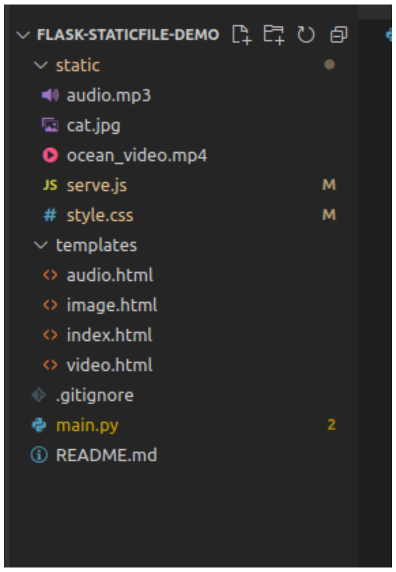
Flask is a lightweight Web Server Gateway Interface or WSGI framework for web applications written in Python. It is designed to make web application development fast and easy and can scale to complex applications. This article describes and demonstrates how to serve various static files in Flask.

Serving Static Files in Flask

Let's configure the virtual environment first. Although this step is optional, we always recommend using a dedicated development environment for each project. This can be achieved in a Python virtual environment.

Now that we have created our Flask app, let's see how to serve static files using the Flask app we just created. First, static files are files served by a web server and do not change over time like CSS and Javascript files used in web applications to improve user experience. Below you will find a demonstration of various static files served by the Flask app.

File Structure



```
FLASK-STATICFILE-DEMO
  static
    audio.mp3
    cat.jpg
    ocean_video.mp4
    serve.js          M
    style.css         M
  templates
    audio.html
    image.html
    index.html
    video.html
  .gitignore
  main.py             2
  README.md
```

HTML File

Serving HTML files using Flask is fairly simple just create a templates folder in the project root directory and create the HTML files, as **templates/index.html.** Here, we are passing text, and with the help of **Jinja {{message}}**, we are printing text that is present in the variable.

HTML

```
<html>
 <head>
   <title>Flask Static Demo</title>
 </head>
 <body>
   <h1>{{message}}</h1>
 </body>
</html>
```

main.py

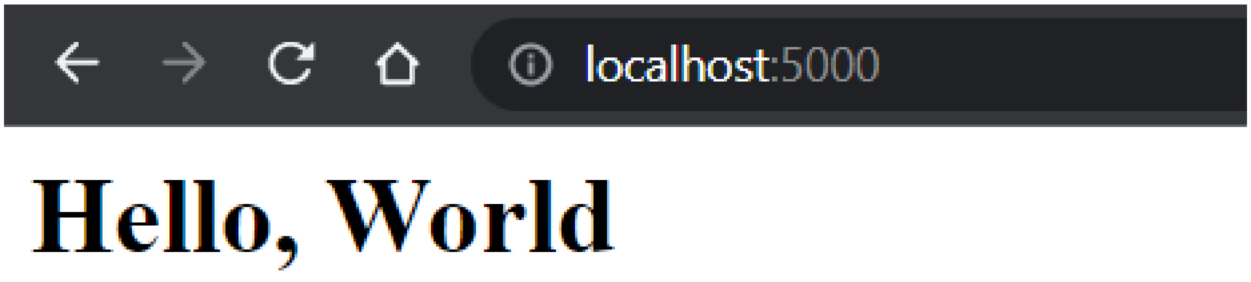
In main.py we render the HTML file when we run it, we are using the render_template() function provided by Flask to render the HTML file. The final code looks like this:

Python3

```
from flask import Flask
from flask import render_template

# creates a Flask application
app = Flask(__name__)


@app.route("/")
def hello():
    message = "Hello, World"
    return render_template('index.html',
                           message=message)

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

Output:

The Flask is up and running on localhost port http://127.0.0.1:5000/



Serve CSS file in Flask

Now serving a CSS file is the same as an HTML file but instead of /templates folder, we create a static folder in the root directory and add all CSS files to it, For simplicity, we have used a very simple CSS file.

CSS

```
h1{
    color: red;
    font-size: 36px;
}
```

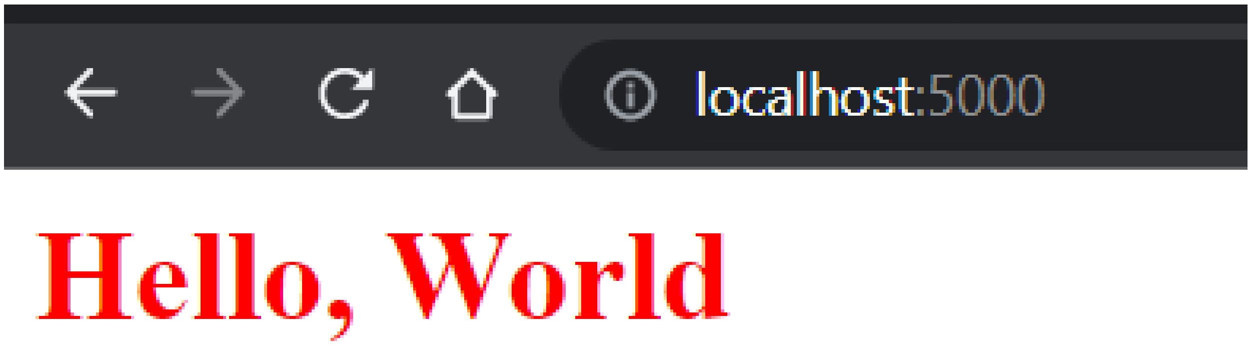
Now, let us link it with the HTML template file using the link tag referring to the CSS file in the static folder.

HTML

```
<html>
```

```
  <head>
    <title>Flask Static Demo</title>
    <link rel="stylesheet" href="/static/style.css" />
  </head>
  <body>
    <h1>{{message}}</h1>
  </body>
</html>
```

Output:



Serve JavaScript file in Flask

To serve Javascript it is the same as a CSS file create a javascript file in the static folder.

Javascript

```
document.write("This is a Javascript static file")
```

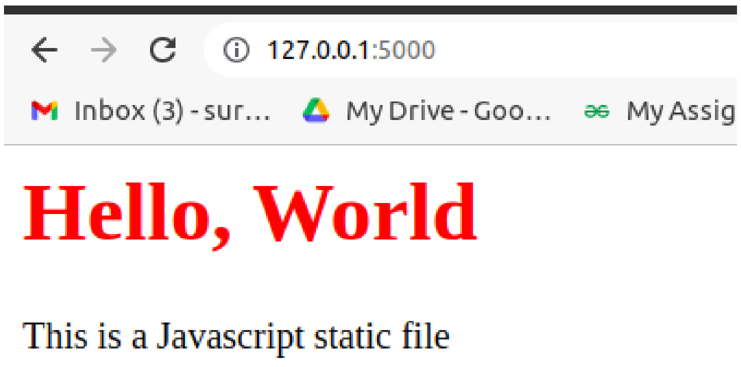
Now link it with the HTML and run the Flask app.

HTML

```
<html>
```

```
  <head>
    <title>Flask Static Demo</title>
    <link rel="stylesheet" href="/static/style.css" />
  </head>
  <body>
    <h1>{{message}}</h1>

    <script src="/static/serve.js" charset="utf-8"></script>
  </body>
</html>
```

Output:



Hello, World

This is a Javascript static file

Serve Media files in Flask (Image, Video, Audio)

You can also use Flask to serve media files such as images, videos, audio files, text files, and PDFs. You can use the same /static folder that you used for CSS and Javascript to serve these kinds of files.

Place all media files in a static folder and associate them with their respective HTML files as shown below. Once all template files have been processed, create routes in main.py for all static files you want to render.

Images

Create an image.html file in the templates folder and add the following code to the main.py and image.html respectively.

Python3

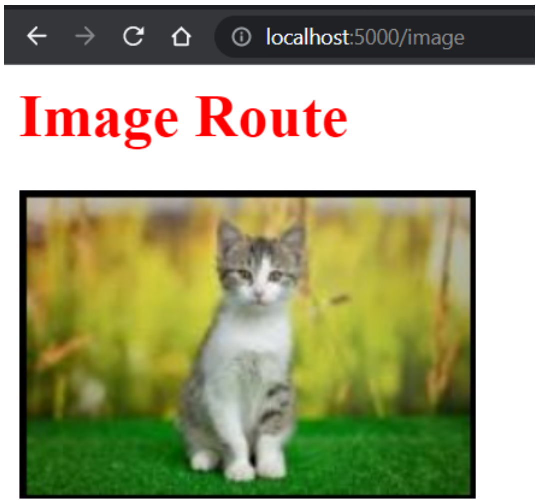
```python
# Images
@app.route("/image")
def serve_image():
    message = "Image Route"
    return render_template('image.html', message=message)
```

templates/images.html

HTML

```html
<html>
  <head>
    <title>Flask Static Demo</title>
    <link rel="stylesheet" href="/static/style.css" />
  </head>
  <body>
    <h1>{{message}}</h1>

    <img src="/static/cat.jpg" alt="Cat image" width="20%" height="auto" />

    <script src="/static/serve.js" charset="utf-8"></script>
  </body>
</html>
```

Output:



Video Files

To serve a video file, create a video.html file in your templates folder and add the following code to your main.py and video.html files.

Python3

```
# video
@app.route("/video")
def serve_video():
    message = "Video Route"
    return render_template('video.html', message=message)
```

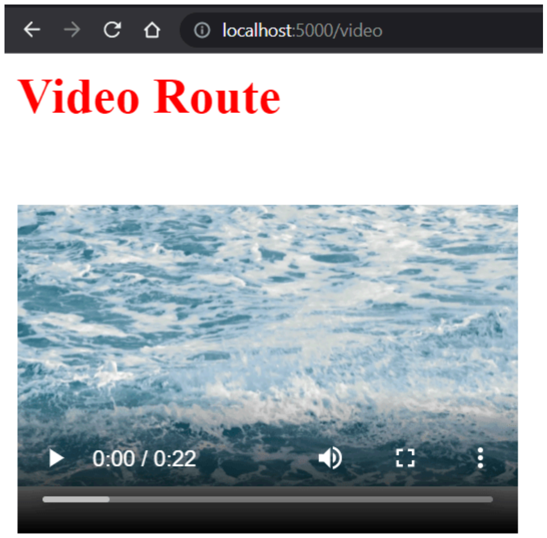
templates/video.html

As you see the mp4 video file is been served by Flask over localhost.

HTML

```
<html>
  <head>
    <title>Flask Static Demo</title>
    <link rel="stylesheet" href="/static/style.css" />
  </head>
  <body>
    <h1>{{message}}</h1>

    <video width="320" height="240" controls>
      <source src="/static/ocean_video.mp4" type="video/mp4" />
    </video>

    <script src="/static/serve.js" charset="utf-8"></script>
  </body>
</html>
```

Output:



Audio Files

Respectively an audio file can be served by creating an **audio.html** template file and adding the following code to the **main.py.**

Python3

```
# audio
@app.route("/audio")
def serve_audio():
    message = "Audio Route"
    return render_template('audio.html', message=message)
```

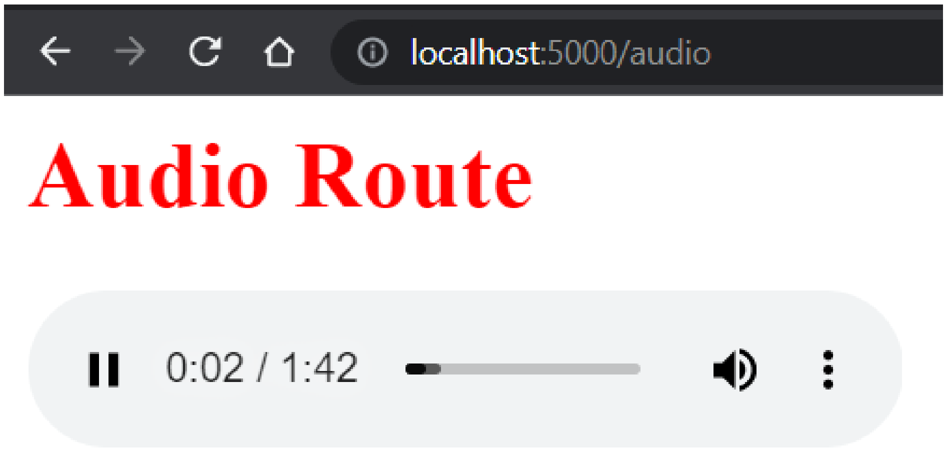
templates/audio.html

HTML

```
<html>
  <head>
    <title>Flask Static Demo</title>
    <link rel="stylesheet" href="/static/style.css" />
  </head>
  <body>
    <h1>{{message}}</h1>

    <audio controls>
```

```
        <source src="/static/audio.mp3" />
      </audio>

      <script src="/static/serve.js" charset="utf-8"></script>
    </body>
  </html>
```

Output:



Complete Flask Code

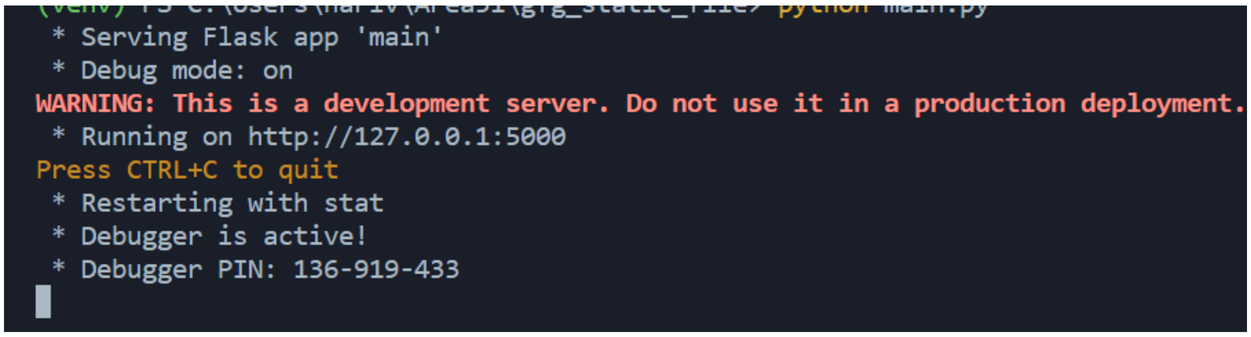
For simplicity, we have created a simple Flask application for a better understanding of how to serve static files in Flask.

Python3

```
from flask import Flask
from flask import render_template

# creates a Flask application
app = Flask(__name__)


@app.route("/")
def hello():
    message = "Hello, World"
    return render_template('index.html', message=message)
```

```
@app.route("/video")
def serve_video():
    message = "Video Route"
    return render_template('video.html', message=message)


@app.route("/audio")
def serve_audio():
    message = "Audio Route"
    return render_template('audio.html', message=message)


@app.route("/image")
def serve_image():
    message = "Image Route"
    return render_template('image.html', message=message)


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

Let's test the Flask app by running it, to run the app just run the **python main.py** which will serve output as shown above:



```
(venv) PS C:\Users\hariv\Area51\gfg_static_file> python main.py
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 136-919-433
```

H hariv... 3

Previous Article

Placeholders in jinja2 Template - Python

Next Article

Uploading and Downloading Files in Flask

Similar Reads

Why Django not Serve your Static Files in Debug False Mode

Django ships with a lightweight server called runserver that allows us to see live previews of our Django application. It promotes faster development, is great for...

3 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

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

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

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

Upload Multiple files with Flask

In online apps, uploading files is a common task. Simple HTML forms with encryption set to multipart/form information are all that is required to publish a...

2 min read

Uploading and Downloading Files in Flask

This article will go over how to upload and download files using a Flask database using Python. Basically, we have a section for uploading files where we can...

7 min read

How to create PDF files in Flask

Whether it's generating a report, exporting data, or creating catalogs, Python and the Flask web framework provide powerful tools to accomplish this task. In this...

4 min read

Working with Static and Media Files in Django

Django's collectstatic management command is a powerful feature that helps manage static files by gathering them from various locations within a project an...

6 min read

Article Tags : Python Python Flask

Practice Tags : python

Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation

OOAD

Puzzles

System Design Bootcamp

Interview Questions

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

World GK

GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects