Uploading and Downloading Files in Flask

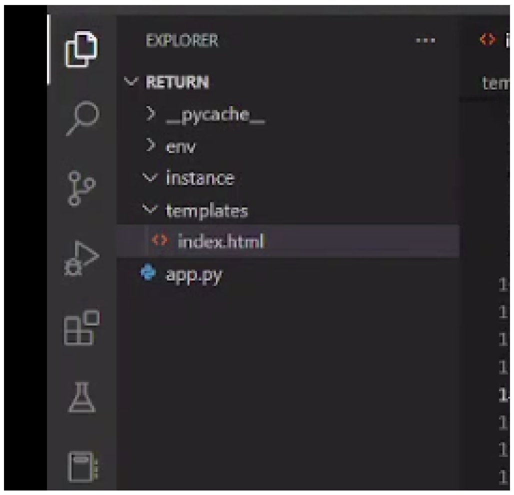
Last Updated : 27 Feb, 2023

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 upload files that will automatically save in our database. When we upload a file and submit it, a message stating that your file has been uploaded and displaying the file name on the screen appears. When we view our Flask SQLite database, we can see that our file has been automatically saved in the database. We can also download our files by using the /download /id number using the link. Consequently, we shall comprehend each phase of this method in this post.

Uploading and Downloading Files in Flask

For our upload and return files with the database in a Flask, first, we create a templates folder for making choose file and submit button in HTML file form so let's get started. To upload and download files with the database in Flask, first we need to download SQLite DB browser to save our data in SQLite.

File structure



Templates File

In the templates file, we will create only one HTML file which will operate our all frontend part code.

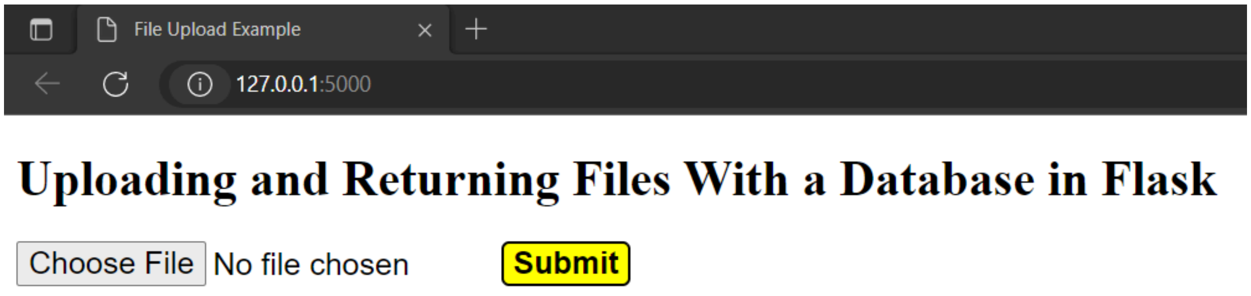
index.html
In the index.html file, we make the first heading using h1 for showing exactly what we are doing and after that, we write one form in which we declare the method POST action by using URL '/' and after that, we initialize data type by using Enctype and we take the multipart/form-data in this so our all file will safely save in database and after that, we write code for simple input button and also submit button for showing all these functionalities clearly we are using some CSS in style tag for making our frontend part beautiful

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>File Upload Example</title>
    <style>
      .ok{

           font-size: 20px;
      }
      .op{
        font-size: 20px;
        margin-left: -70px;
        font-weight: bold;
        background-color: yellow;
        border-radius: 5px;
        cursor: pointer;
      }


    </style>
</head>
<body>
    <div class="center">
        <h1> Uploading and Returning Files With a Database in Flask  </h1>
        <form method="POST" action="/" enctype="multipart/form-data">
```

```
                <input class="ok" type="file" name="file">
                <button class="op">Submit</button>
            </form>
        </div>

    </body>
    </html>
```

Output:



Uploading and Returning Files With a Database in Flask

Html Output

app.py

After writing code for templates we create an app.py file outside of the templates folder and create app,.py in which we will write our main code of uploading and returning files with a database in a Flask in Python language.

Step 1: Import all libraries

In the app.py file we will write our main part of the code through these all operations will operate easily first in app.py we need to import all important libraries which are important for doing upload and returning files with database operations first we import io BytesIO this module will convert our all pdf files binary in below you can see what type of output will show in the database when we upload any pdf in the database and we import render_template for rendering templates and after that, we are importing send_file module which will help us to send the file to database and after that, we are importing flask_sqlalchemy for our SQL data

Python

```
from io import BytesIO
from flask import Flask, render_template, request, send_file
from flask_sqlalchemy import SQLAlchemy
```

Step 2: Create a database

After importing all libraries we create an SQL database for uploading and returning our file we initialize the Flask function and after that, we make a database sqlite:///db.sqlite3 to save our uploading files and we create one DB as SQLAchemy saving database

Python3

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///db.sqlite3'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

Step 3: Create a table for the database

after creating the database we need to make a table database for which we create one ID which we make the primary key after making the id we crate the filename and in the filename we initialize a varchar limit of 50 and after that, we create one data which is a blob which means we can see our file by click on the blob and for making all these functionalities we write following lines of code in our app.py file

Python3

```
class Upload(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(50))
    data = db.Column(db.LargeBinary)
```

Step 4: Create an index function

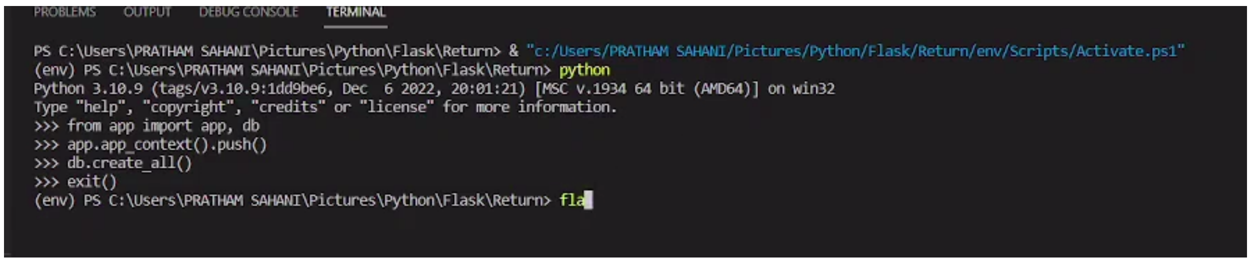
We create one index function in which we set the first path '/' and after that, we pass the request for method functionality in method functionality we create one var upload in which we initialize the filename and read the file if we upload a pdf so it will read it in binary form after writing this we create DB session in our database and also we commit our DB session after creating and after that we return file name with flashing massage uploaded ' filename' and we return our render template on index,.html file

Python3

```
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['file']
        upload = Upload(filename=file.filename, data=file.read())
        db.session.add(upload)
        db.session.commit()
        return f'Uploaded: {file.filename}'
    return render_template('index.html')
```

After writing the code we need to create db.sqlite in our system to create a database we need to run the following command in the terminal:

```
python
from app import app, db
app.app_context().push()
db.create_all()
exit()
```



```
PS C:\Users\PRATHAM SAHANI\Pictures\Python\Flask\Return> & "c:/Users/PRATHAM SAHANI/Pictures/Python/Flask/Return/env/Scripts/Activate.ps1"
(env) PS C:\Users\PRATHAM SAHANI\Pictures\Python\Flask\Return> python
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from app import app, db
>>> app.app_context().push()
>>> db.create_all()
>>> exit()
(env) PS C:\Users\PRATHAM SAHANI\Pictures\Python\Flask\Return> fla
```

by running the above command in the terminal we can create a database in which our file will save after uploading and submitting the database and we can see our database in the instance folder automatically created for running how to run these commands watch a video which is attached below.

Step 5: Create a download function for downloading the file.

After uploading our file we need to make a download function for download our uploading file for download file we make one upload variable in which we add an upload query and we filter our files by id as we set every id unique so every file has a unique id and after that, we return our file by using send_file and also for download pdf file we convert it into binary to our data file by using BytesIO module and after we can download our file by using the following link

```
http://127.0.0.1:5000/download/id_number
```

Python

```python
@app.route('/download/<upload_id>')
def download(upload_id):
    upload = Upload.query.filter_by(id=upload_id).first()
    return send_file(BytesIO(upload.data), download_name=upload.filename, as_attac
```

Complete Code

Python3

```python
# import all libraires
from io import BytesIO
from flask import Flask, render_template, request, send_file
from flask_sqlalchemy import SQLAlchemy

# Initialize flask  and create sqlite database
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///db.sqlite3'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

```python
# create datatable
class Upload(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(50))
    data = db.Column(db.LargeBinary)

# Create index function for upload and return files
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['file']
        upload = Upload(filename=file.filename, data=file.read())
        db.session.add(upload)
        db.session.commit()
        return f'Uploaded: {file.filename}'
    return render_template('index.html')

# create download function for download files
@app.route('/download/<upload_id>')
def download(upload_id):
    upload = Upload.query.filter_by(id=upload_id).first()
    return send_file(BytesIO(upload.data),
                     download_name=upload.filename, as_attachment=True)
```

To run the above code we need to run the following command in the terminal:

```
flask run
```

Output:

After running these commands in the terminal when we upload files like pdf and images so following output will show in the database when we blob SQLite database.

Image upload

When we upload images so following image type interface will display on the screen when we blob our database.

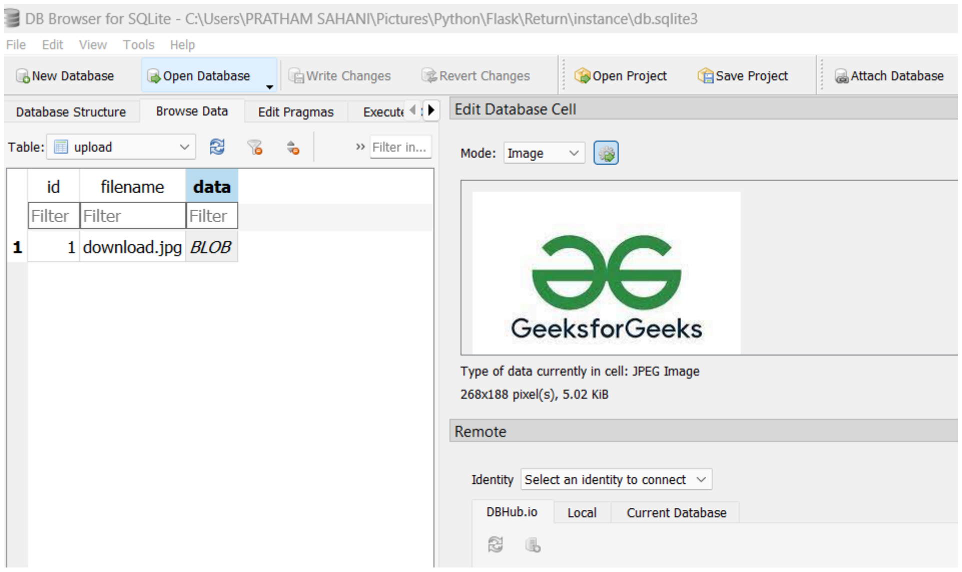

DB Browser for SQLite - C:\Users\PRATHAM SAHANI\Pictures\Python\Flask\Return\instance\db.sqlite3

Table: upload

	id	filename	data
1	1	download.jpg	*BLOB*

Edit Database Cell

Mode: Image

Type of data currently in cell: JPEG Image
268x188 pixel(s), 5.02 KiB

Remote

Identity: Select an identity to connect

Database for image

Pdf upload

When we upload any pdf file so database converts our pdf file into the binary form so the following type of interface will show when we upload any pdf file in the database and the blob.

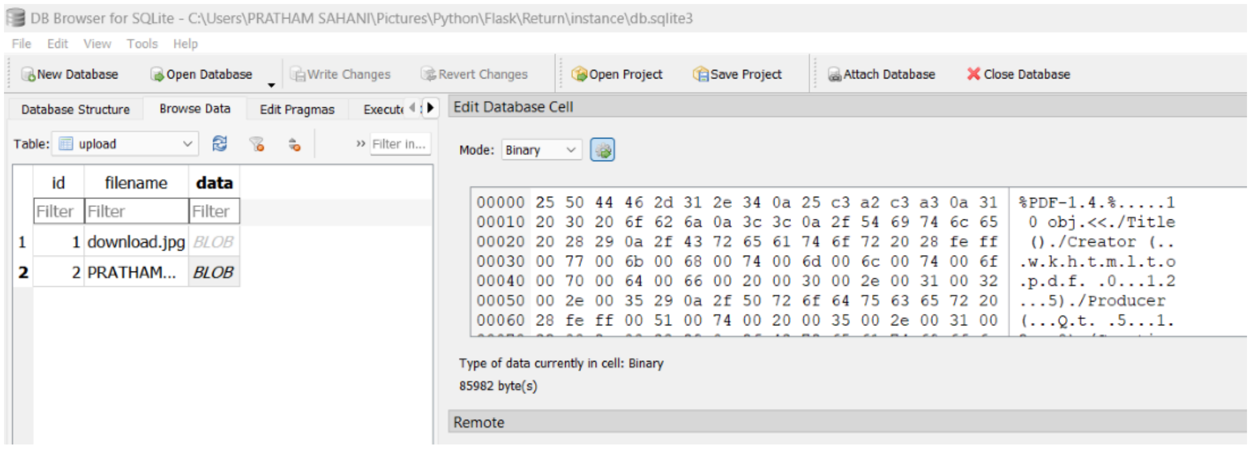

DB Browser for SQLite - C:\Users\PRATHAM SAHANI\Pictures\Python\Flask\Return\instance\db.sqlite3

Table: upload

	id	filename	data
1	1	download.jpg	*BLOB*
2	2	PRATHAM...	*BLOB*

Edit Database Cell

Mode: Binary

```
00000 25 50 44 46 2d 31 2e 34 0a 25 c3 a2 c3 a3 0a 31  %PDF-1.4.%.....1
00010 20 30 20 6f 62 6a 0a 3c 3c 0a 2f 54 69 74 6c 65   0 obj.<<./Title
00020 20 28 29 0a 2f 43 72 65 61 74 6f 72 20 28 fe ff   ()./Creator (..
00030 00 77 00 6b 00 68 00 74 00 6d 00 6c 00 74 00 6f  .w.k.h.t.m.l.t.o
00040 00 70 00 64 00 66 00 20 00 30 00 2e 00 31 00 32  .p.d.f. .0...1.2
00050 00 2e 00 35 29 0a 2f 50 72 6f 64 75 63 65 72 20  ...5)./Producer
00060 28 fe ff 00 51 00 74 00 20 00 35 00 2e 00 31 00  (...Q.t. .5...1.
```

Type of data currently in cell: Binary
85982 byte(s)

Remote

Database of pdf file in binary form

Output Video:

00:00 03:37

7

Previous Article

How to serve static files in Flask

Next Article

How to Upload File in Python-Flask

Similar Reads

Uploading and Reading a CSV File in Flask

Flask is a flexible, lightweight web-development framework built using python. A Flask application is a Python script that runs on a web server, which listens to...

3 min read

Downloading files from web using Python

Requests is a versatile HTTP library in python with various applications. One of its applications is to download a file from web using the file URL. Installation:...

4 min read

Uploading files on Google Drive using Python

In the modern era of the internet, a large number of mundane or work-related tasks are carried out over the internet. Compromise of data stored on a device...

3 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

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

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

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

Article Tags : Python Technical Scripter Python Flask Technical Scripter 2022

Practice Tags : python

GeeksforGeeks
Sanchhaya Education Private Limited

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

GET IT ON Google Play
Download on the App Store

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

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved