



Flask – (Creating first simple application)

Last Updated : 18 Aug, 2024

Building a webpage using python.

There are many modules or frameworks which allow building your webpage using python like a bottle, Django, Flask, etc. But the real popular ones are Flask and Django. Django is easy to use as compared to Flask but Flask provides you with the versatility to program with.

To understand what Flask is you have to understand a few general terms.

1. **WSGI** Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.
2. **Werkzeug** It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.
3. **jinja2** jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask is a web application framework written in Python. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Installation:

We will require two packages to set up your environment. *virtualenv* for a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries and the next will be *Flask* itself.

virtualenv

```
pip install virtualenv
```

Create Python virtual environment

```
virtualenv venv
```

Activate virtual environment

```
windows > venv\Scripts\activate
```

```
linux > source ./venv/bin/activate
```

For windows, if this is your first time running the script, you might get an error like below:

```
venv\Scripts\activate : File C:\flask_project\venv\Scripts\Activate.ps1 cannot be loaded because running scripts is disabled on this system. For more information, see about_Execution_Policies at https://go.microsoft.com/fwlink/?LinkID=135170.
```

```
At line:1 char:1
```

```
+ venv\Scripts\activate
```

```
+ FullyQualifiedErrorId : UnauthorizedAccess
```

This means that you don't have access to execute the scripts.

To solve this error, run the powershell as admin, when you right click on powershell icon, choose the option 'Run as administrator'. Now, the powershell will open in the admin mode.

Type the following command in Shell

```
set-executionpolicy remotesigned
```

Now, you will be prompted to change the execution policy. Please type A. This means Yes to all.

Flask

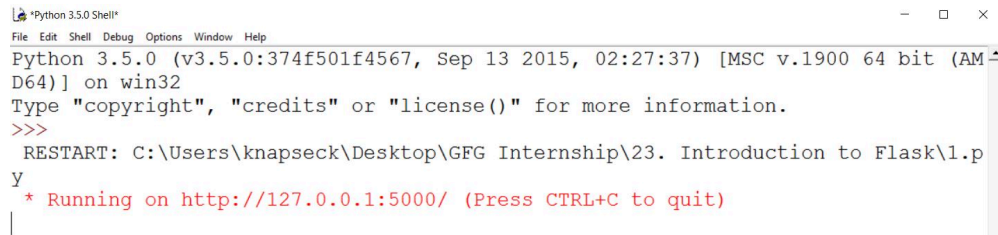
```
pip install Flask
```

After completing the installation of the package, let's get our hands on the code.

Python

```
1  # Importing flask module in the project is mandatory
2  # An object of Flask class is our WSGI application.
3  from flask import Flask
4
5  # Flask constructor takes the name of
6  # current module (__name__) as argument.
7  app = Flask(__name__)
8
9  # The route() function of the Flask class is a decorator,
10 # which tells the application which URL should call
11 # the associated function.
12 @app.route('/')
13 # '/' URL is bound with hello_world() function.
14 def hello_world():
15     return 'Hello World'
16
17 # main driver function
18 if __name__ == '__main__':
19
20     # run() method of Flask class runs the application
21     # on the local development server.
22     app.run()
```

Save it in a file and then run the script we will be getting an output like this.



```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\knapseck\Desktop\GFG Internship\23. Introduction to Flask\1.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Then go to the URL given there you will see your first webpage displaying hello world there on your local server.

Digging further into the context, the **route()** decorator in Flask is used to bind a URL to a function. Now to extend this functionality our small web app is also equipped with another method **add_url_rule()** which is a function of an application object that is also available to bind a URL with a function as in the above example, **route()** is used.

Example:

```
def gfg():
    return 'geeksforgeeks'
app.add_url_rule('/', 'g2g', gfg)
```

Output:

geeksforgeeks

You can also add variables in your web app, well you might be thinking about how it'll help you, it'll help you to build a URL dynamically. So let's figure it out with an example.

Python



```
1 from flask import Flask
2 app = Flask(__name__)
3
```



```
4 @app.route('/hello/<name>')
5 def hello_name(name):
6     return 'Hello %s!' % name
7
8 if __name__ == '__main__':
9     app.run()
```

And go to the URL `http://127.0.0.1:5000/hello/geeksforgeeks` it'll give you the following output.



Hello geeksforgeeks!

We can also use HTTP methods in Flask let's see how to do that

The HTTP protocol is the foundation of data communication on the world wide web. Different methods of data retrieval from specified URL are defined in this protocol. The methods are described down below.

GET : Sends data in simple or unencrypted form to the server.

HEAD : Sends data in simple or unencrypted form to the server without body.

PUT : Replaces target resource with the updated content.

DELETE : Deletes target resource provided as URL.

By default, the Flask route responds to the GET requests. However, this preference can be altered by providing methods argument to route() decorator. In order to demonstrate the use of the POST method in URL routing, first, let us create an HTML form and use the POST method to send form data to a URL. Now let's create an HTML login page.

Below is the source code of the file:

HTML



```
1 <html>
2   <body>
3     <form action = "http://localhost:5000/login" method =
      "post">
4       <p>Enter Name:</p>
5       <p><input type = "text" name = "nm" /></p>
6       <p><input type = "submit" value = "submit" /></p>
7     </form>
8   </body>
9 </html>
```

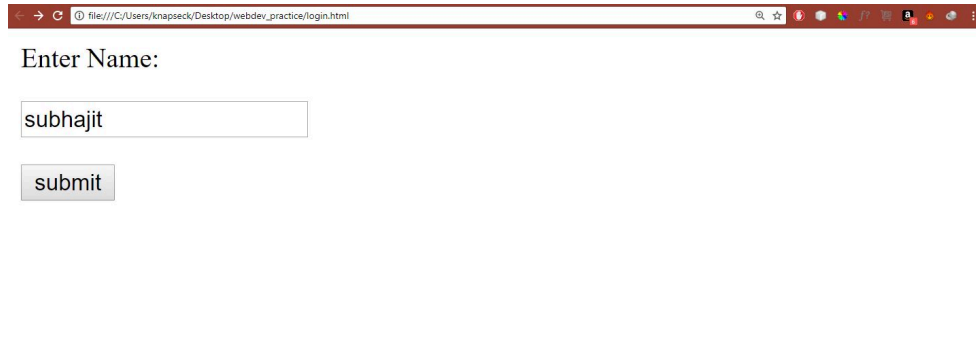
Now save this file HTML and try this python script to create the server.

Python



```
1 from flask import Flask, redirect, url_for, request
2 app = Flask(__name__)
3
4
5 @app.route('/success/<name>')
6 def success(name):
7     return 'welcome %s' % name
8
9
10 @app.route('/login', methods=['POST', 'GET'])
11 def login():
12     if request.method == 'POST':
13         user = request.form['nm']
14         return redirect(url_for('success', name=user))
15     else:
16         user = request.args.get('nm')
17         return redirect(url_for('success', name=user))
18
19
20 if __name__ == '__main__':
21     app.run(debug=True)
```

After the development server starts running, open login.html in the browser, enter your name in the text field and click *submit* button. The output would be the following.



The result will be something like this



And there's much more to Flask than this. If you are interested in this web framework of Python you can dig into the links provided down below for further information.

Flask – (Creating first simple application) – FAQs

What are the basic components of a Flask application?

1. Flask Object: Main app instance.

```
from flask import Flask
app = Flask(__name__)
```

2. Routes: Define URL mappings.

```
@app.route('/')
def home():
    return 'Hello, World!'
```

3. Templates: HTML files for rendering.

```
<!-- templates/hello.html -->
<h1>Hello, {{ name }}!</h1>
```

4. Static Files: Assets like CSS.

```
<link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
```

5. Request and Response: Handle data.

```
from flask import request, jsonify

@app.route('/json')
def json_example():
    return jsonify({"message": "Hello, JSON!"})
```

How to handle routing in Flask?

1. Basic Route:

```
@app.route('/')
def home():
    return 'Hello, World!'
```

2. Dynamic Route:

```
@app.route('/user/<username>')
def show_user_profile(username):
```



```
return f'User: {username}'
```

3. HTTP Methods:

```
@app.route('/login', methods=['POST'])
def login():
    return 'Logging in...'
```

4. Redirects and Errors:

```
from flask import redirect, url_for, abort

@app.route('/redirect')
def redirect_example():
    return redirect(url_for('home'))

@app.route('/error')
def error_example():
    abort(404)
```

What are the best practices for setting up a Flask environment?

1. Virtual Environments:

```
python -m venv venv
source venv/bin/activate
```

2. requirements.txt:

```
pip freeze > requirements.txt
```

3. Project Structure:

```
myapp/
├── app.py
├── templates/
├── static/
└── requirements.txt
```

4. Environment Variables:

```
import os
app.config['SECRET_KEY'] = os.getenv('SECRET_KEY')
```

5. Flask Extensions:

```
pip install Flask-SQLAlchemy
```

How to deploy a basic Flask application?

1. Local Deployment:

```
python app.py
```

2. Production Deployment (Gunicorn):

```
pip install gunicorn
gunicorn -w 4 app:app
```

3. Deploy to Cloud:

- **Heroku:**

```
heroku create
git push heroku master
```

4. Docker

```
FROM python:3.9
WORKDIR /app
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD ["gunicorn", "-w", "4", "app:app"]
```

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!



GeeksforGeeks



55

Previous Article

[How to Install Flask in Windows?](#)

Next Article

[How to Run a Flask Application](#)

Similar Reads

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

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

How to write a simple Flask API for hello world?

Prerequisites: 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

Creating Custom Commands in Flask

This article revolves around how to create custom commands in flask. Every time one runs flask using flask run, run is actually a command which initiates a...

2 min read

Creating Your First Application in Python

Python is one of the simplest programming language out there. In fact, it was developed for the sole purpose of simplifying the process of learning a...

5 min read

Creating first web application using Bottle Framework - Python

There are many frameworks in python which allow you to create a webpage like bottle, flask, django. In this article, you will learn how to create a simple app...

2 min read

Article Tags : [Python](#) [Web Technologies](#)

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



[Company](#)

[Languages](#)

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

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
OOAD

Interview Preparation

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