



Subdomain in Flask | Python

Last Updated : 17 Apr, 2019

Prerequisite: [Introduction to Flask](#)

In this article, we will learn how to setup subdomains in Flask. But first, let's go through the basic like what is DNS and subdomains.

Domain Name System (DNS):

The Domain Name System (DNS) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.

DNS is basically using words (Domain Names) in place of numbers (IP addresses) to locate something. For example, 127.0.0.1 is used to point the local computer address, *localhost*.

Subdomain:

A subdomain is a domain that is part of a larger domain. Basically, it's a sort of child domain which means it is a part of some parent domain. For example, `practice.geeksforgeeks.org` and `write.geeksforgeeks.org` are subdomains of the `geeksforgeeks.org` domain, which in turn is a subdomain of the `org` top-level domain (TLD).

These are different from the path defined after TLD as in `geeksforgeeks.org/basic/`.

Further, we will discuss how to set endpoints in your web application using Python's micro-framework, Flask.

Adding alternate domain name for local IP –

Prior to the coding part, we got to setup *hosts* file in order to provide alternate

names to local IP so that we are able to test our app locally. Edit this file with root privileges.

Linux: /etc/hosts

Windows: C:\Windows\System32\Drivers\etc\hosts

Add these lines to set up alternate domain names.

```
127.0.0.1      vibhu.gfg
127.0.0.1      practice.vibhu.gfg
```

In this example, we're considering vibhu.gfg as our domain name, with gfg being the TLD. practice would be a subdomain we're targeting to set in our web app.

Setting up the Server –

In the app's configuration SERVER_NAME is set to the domain name, along with the port number we intend to run our app on. The default port, flask uses is 5000, so we take it as it is.

```
from flask import Flask

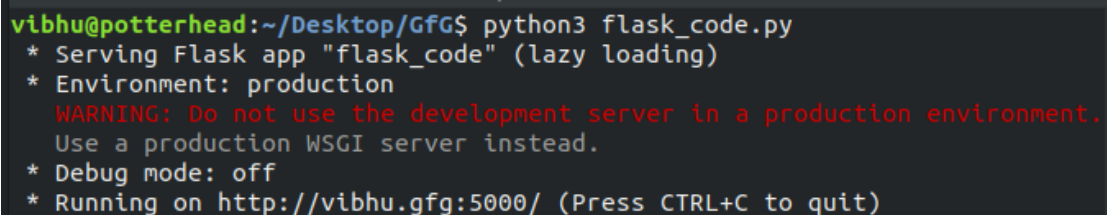
app = Flask(__name__)

@app.route('/')
def home():
    return "Welcome to GeeksForGeeks !"

if __name__ == "__main__":
    website_url = 'vibhu.gfg:5000'
    app.config['SERVER_NAME'] = website_url
    app.run()
```

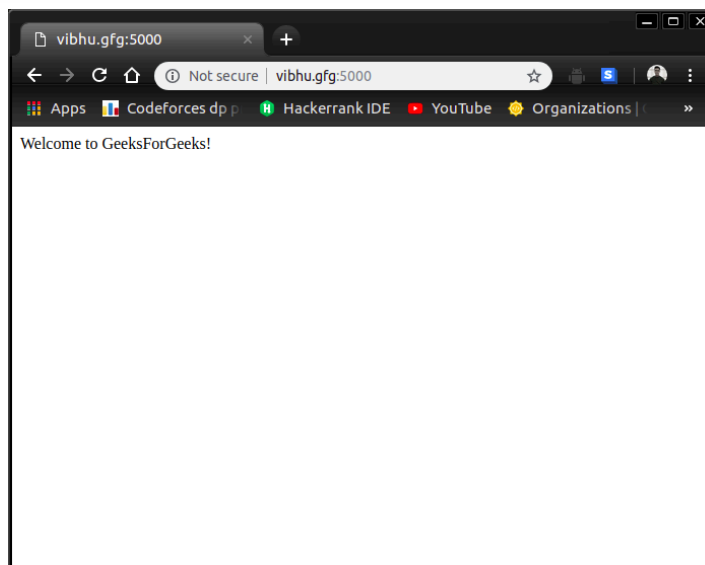
Output:

Run the app and notice the link on which the app is running.

A terminal window with a dark background. The prompt is 'vibhu@potterhead:~/Desktop/GfG\$'. The command 'python3 flask_code.py' has been executed. The output shows: '* Serving Flask app "flask_code" (lazy loading)', '* Environment: production', a red warning 'WARNING: Do not use the development server in a production environment. Use a production WSGI server instead.', '* Debug mode: off', and '* Running on http://vibhu.gfg:5000/ (Press CTRL+C to quit)'.

```
vibhu@potterhead:~/Desktop/GfG$ python3 flask_code.py
* Serving Flask app "flask_code" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://vibhu.gfg:5000/ (Press CTRL+C to quit)
```

Test the link on your browser.



Adding Several Endpoints –

1. **basic:** An endpoint with extension to the path on the main domain.
2. **practice:** An endpoint serving on the practice subdomain.
3. **courses:** An endpoint with extension on to the path on the practice subdomain.

Subdomains in Flask are set using the `subdomain` parameter in the `app.route` decorator.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Welcome to GeeksForGeeks !"

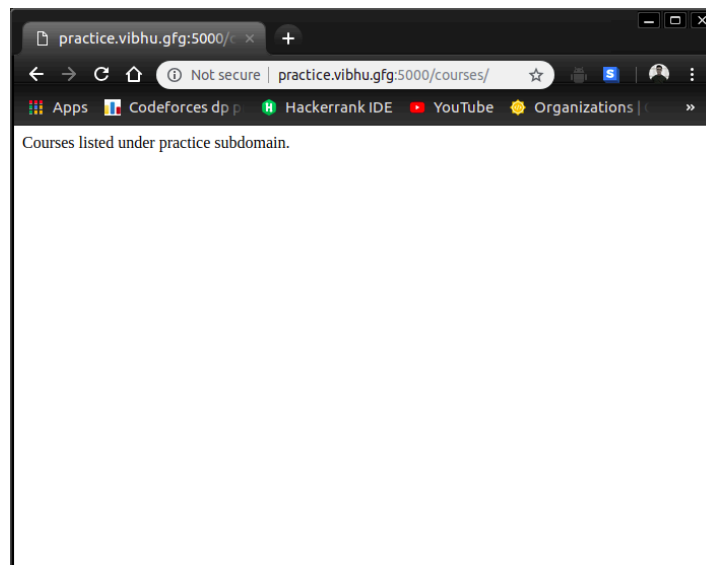
@app.route('/basic/')
def basic():
    return "Basic Category Articles " \
           "listed on this page."

@app.route('/', subdomain='practice')
def practice():
    return "Coding Practice Page"
```

```
@app.route('/courses/', subdomain='practice')
def courses():
    return "Courses listed " \
           "under practice subdomain."

if __name__ == "__main__":
    website_url = 'vibhu.gfg:5000'
    app.config['SERVER_NAME'] = website_url
    app.run()
```

Output:



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!



vibhu... + Follow



6

Previous Article

How to execute raw SQL in Flask-SQLAlchemy app

Next Article

Handling 404 Error in Flask

Similar Reads

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

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 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 Make a Subdomain Scanner in Python?

In this article, we are going to scan the subdomains using requests module in Python, which allows us to easily make HTTPS requests to get information from...

6 min read

Massc - Subdomain Scanner Tool Designed in JavaScript

Subdomain enumeration is the process of finding valid (resolvable) subdomains for one or more domain(s). The general system is to use a dictionary of common...

3 min read

Subdomain takeover from scratch to advance

Sub-domain Takeover : Sub-domain takeover is a common and most popular vulnerability. If you are not aware of such kind of vulnerability, you can...

2 min read

How to Setup a Subdomain on Your WordPress Website ?

Setting up a subdomain on your WordPress website can help you organize your content better and provide a dedicated space for specific sections of your site,...

4 min read

Python Flask - ImmutableMultiDict

MultiDict is a sub-class of Dictionary that can contain multiple values for the same key, unlike normal Dictionaries. It is used because some form elements ha...

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

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

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

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Languages

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

Data Science & ML

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

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

System Design

- High Level Design
- Low Level Design
- UML Diagrams
- Interview Guide
- Design Patterns
- OOAD
- System Design Bootcamp
- Interview Questions

School Subjects

- Mathematics
- Physics
- Chemistry
- Biology
- Social Science
- English Grammar
- Commerce
- World GK

DevOps

- Git
- Linux
- AWS
- Docker
- Kubernetes
- Azure
- GCP
- DevOps Roadmap

Interview Preparation

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

GeeksforGeeks Videos

- DSA
- Python
- Java
- C++
- Web Development
- Data Science
- CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved