Flask – HTTP Method

Last Updated : 28 Mar, 2024

In this article, we will learn how to handle HTTP methods, such as GET and POST in Flask using Python. Here, we will understand the concept of HTTP, GET, and HTTP POST, and then we will the example and implement each in Flask. Before starting let's understand the basic terminology:

- **GET**: to request data from the server.
- **POST**: to submit data to be processed to the server.
- **PUT**: A PUT request is used to modify the data on the server. It replaces the entire content at a particular location with data that is passed in the body payload. If there are no resources that match the request, it will generate one.
- **PATCH**: PATCH is similar to a PUT request, but the only difference is, it modifies a part of the data. It will only replace the content that you want to update.
- **DELETE**: A DELETE request is used to delete the data on the server at a specified location.

Related Article: Flask HTTP methods, Handle GET & POST requests

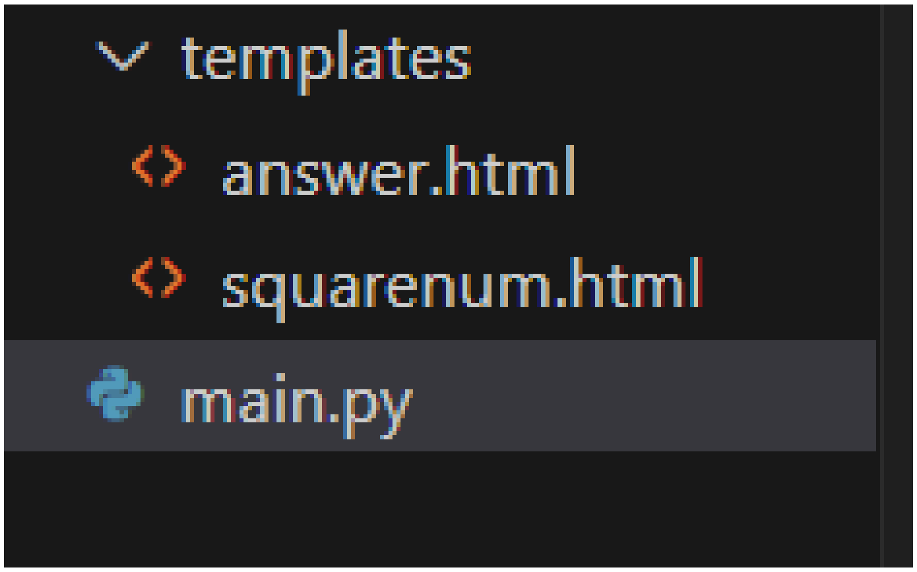
Flask HTTP Methods

In a Client-Server architecture, there is a set of rules, called a protocol, using which, we can allow the clients, to communicate with the server, and, vice-versa. Here, the Hyper Text Transfer Protocol is used, through which, communication is possible. For example, Our browser, passes our query, to the Google server, receiving which, the Google server, returns relevant suggestions. The commonly used HTTP methods, for this interconnection, are – **GET** and **POST**.4

GET Method in Flask

The request we type, in the browser address bar, say: 'http://google.com' is called the Uniform Resource Locator. It mentions the address we are looking for, in this case, the Google landing(starting) page. The browser, sends a GET request, to the Google server, which returns the starting webpage, in response. The GET request is simply used, to fetch data from the server. It should not be used, to apply changes, to the server data.

File Structure



```
templates
    answer.html
    squarenum.html
main.py
```

File Structure

Example of HTTP GET in Flask

Let us discuss, the execution of the GET method, using the Flask library. In this example, we will consider, a landing page, that gives us facts, about Math calculations, and, allows us to enter a number, and, return its square. Let us see the example:

Step 1: The '**squarenum.html**' file, has a form tag, that allows the user to enter a number. The form data is sent, to the page mentioned, in the 'action' attribute. Here, the data is sent, to the same page, as indicated by '#'. Since we are using the GET method, the data will be appended, to the URL, in the name-value pair format. So, if we enter number 12, and, click on Submit button, then data will be appended, as 'http://localhost:5000/square?num=12&btnnum=Submit#'

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Square Of Number!</title>
</head>
<body>
<h1><i> Welcome to the Maths page!</i></h1>
    <p>Logic shapes every choice of our daily lives.<br>
    Logical thinking enables someone to learn and
    make decisions that affect their way of life. !</p>
    <form method="GET" action ="#">
        Enter a number :
        <input type="text" name="num" id="num"></input>
        <input type="submit" name="btnnum" id="btnnum">
</input>
  </form>
</body>
</html>
```

Step 2: The **answer.html** code file looks as follows:

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Answer Page!</title>
</head>
<body>
    <h1>Keep Learning Maths!</h1>
    <h2>Square of number {{num}} is :{{squareofnum}}</h2>
</body>
</html>
```

Step 3: Here, we have written a view function, called ‘squarenumber’. The view function returns an HTTP response. The function is decorated with 'app.route('/square')'. This decorator, matches the incoming request URLs, to the view functions. Thus, incoming requests like 'localhost:5000/' will be mapped to the view squarenumber() function.

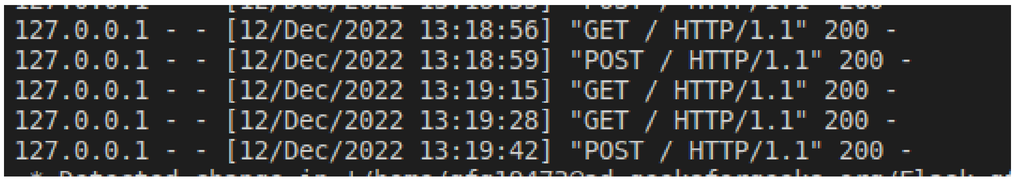
In this case, we have used a GET, hence we mention 'methods=['GET']' in the app.route decorator. The HTML form will append the number, entered by the user, in the URL. Hence, we check if data is present. To do so, we have to use the if-elif-else logic. When data is not present, the value of argument "num" will be None. Here, we will display, the webpage to the user. If the user has not entered, any number, and right away clicked the Submit button, then, we have displayed the error message. We have to use the Flask Jinja2 template, to pass the result, and, the number entered into the HTML file.

Python3

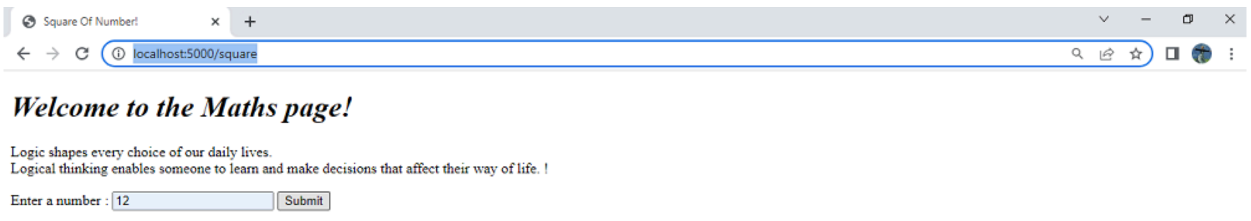
```python
# import the Flask library
from flask import Flask, render_template, request


# Create the Flask instance and pass the Flask
# constructor the path of the correct module
app = Flask(__name__)

# The URL  'localhost:5000/square' is mapped to
# view function 'squarenumber'
# The GET request will display the user to enter
# a number (coming from squarenum.html page)


@app.route('/square', methods=['GET'])
def squarenumber():
    # If method is GET, check if  number is entered
    # or user has just requested the page.
    # Calculate the square of number and pass it to
    # answermaths method
    if request.method == 'GET':
   # If 'num' is None, the user has requested page the first time
        if(request.args.get('num') == None):
            return render_template('squarenum.html')
          # If user clicks on Submit button without
```

```
     # entering number display error
    elif(request.args.get('num') == ''):
      return "<html><body> <h1>Invalid number</h1></body></html>"
    else:
     # User has entered a number
     # Fetch the number from args attribute of
     # request accessing its 'id' from HTML
      number = request.args.get('num')
      sq = int(number) * int(number)
      # pass the result to the answer HTML
      # page using Jinja2 template
      return render_template('answer.html',
                  squareofnum=sq, num=number)


# Start with flask web app with debug as
# True only if this is the starting page
if(__name__ == "__main__"):
  app.run(debug=True)
```

Output:



```
127.0.0.1 - - [12/Dec/2022 13:18:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2022 13:18:59] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2022 13:19:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2022 13:19:28] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2022 13:19:42] "POST / HTTP/1.1" 200 -
```



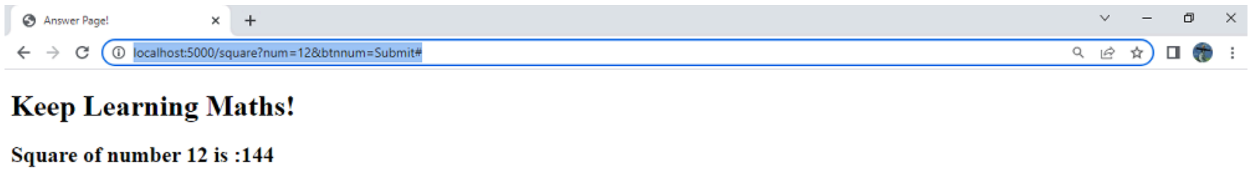
Welcome to the Maths page!

Logic shapes every choice of our daily lives.
Logical thinking enables someone to learn and make decisions that affect their way of life. !

Enter a number : 12 Submit

The user requests 'localhost:5000/square' and enters a number

On clicking the Submit button, one can notice, the value entered, appended, to the URL, and, the output displayed.



Keep Learning Maths!

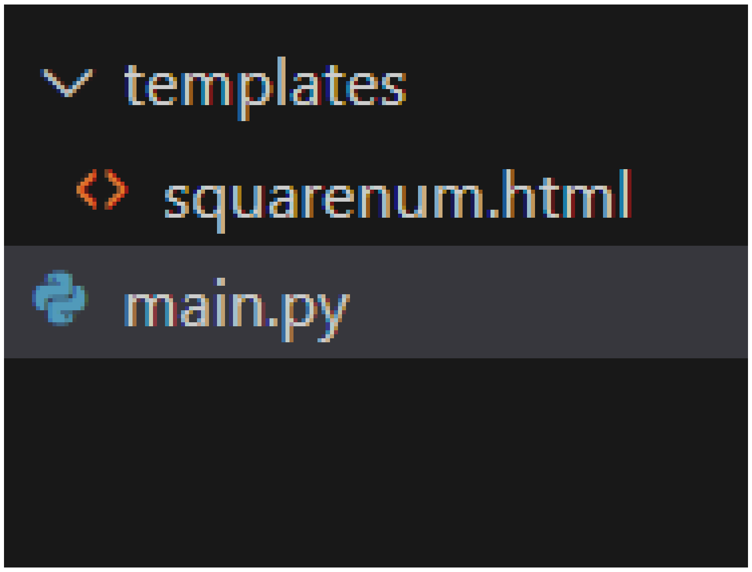
Square of number 12 is :144

Data entered is appended in the URL and output is displayed.

POST Method in Flask

Suppose, we need to register our details, to a website, OR, upload our files, we will send data, from our browser(the client) to the desired server. The HTTP method, preferred here, is POST. The data sent, from HTML, is then saved, on the server side, post validation. The POST method should be used, when we need to change/add data, on the server side.

File Structure



File Structure

Example of HTTP POST in Flask

In this example, we will consider, the same landing page, giving us facts, about Math calculations, and, allowing us to enter a number, and, return its square. Let us see the example:

Step 1: The same HTML page, called '**squarenum.html**', is in the templates folder. At the backend side, we will write, appropriate logic, in a view function, to get the number, entered by the user, and, return the same, in the 'answer.html' template. The frontend code file is as shown below:

HTML

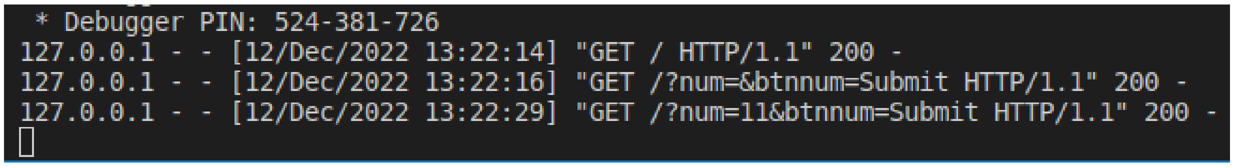
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Square Of Number!</title>
</head>
<body>
<h1><i> Welcome to the Maths page!</i></h1>
    <p>Logic shapes every choice of our daily lives.<br>
    Logical thinking enables someone to learn and
    make decisions that affect their way of life. !</p>
    <form method="POST" action ="#">
        Enter a number :
        <input type="text" name="num" id="num"></input>
        <input type="submit" name="btnnum" id="btnnum">
</input>
  </form>
</body>
</html>
```

Step 2: The view function squarenumber(), now also contains value POST in the 'methods' attribute, in the decorator. Thus, when the user requests the page, the first time, by calling "http://localhost:5000/square", a GET request will be made. Here, the server will render, the corresponding "squarenum.html", webpage. After clicking on Submit, on entering a number, the data is posted back, to the same webpage. Here, the POST method, sends data, in the message body, unlike GET, which appends data in the URL. In the view function, the If-Else condition block, retrieves the number, by accessing the 'form' attribute, of the request. The square of the number is calculated, and, the value is passed to the same "answer.html" webpage, using the Jinja2 template.

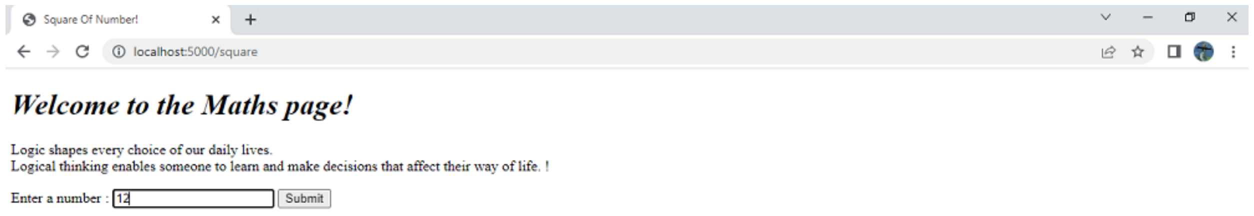
Python3

```python
# import the Flask library
from flask import Flask, render_template, request


# Create the Flask instance and pass the Flask constructor the path of the correct module
app = Flask(__name__)


@app.route('/', methods=['GET', 'POST'])
def squarenumber():
 # If method is POST, get the number entered by user
 # Calculate the square of number and pass it to answermaths
    if request.method == 'POST':
        if(request.form['num'] == ''):
            return "<html><body> <h1>Invalid number</h1></body></html>"
        else:
            number = request.form['num']
            sq = int(number) * int(number)
            return render_template('answer.html',
                            squareofnum=sq, num=number)
    # If the method is GET,render the HTML page to the user
    if request.method == 'GET':
        return render_template("squarenum.html")


# Start with flask web app with debug as True only
# if this is the starting page
if(__name__ == "__main__"):
    app.run(debug=True)
```

Output:



```
 * Debugger PIN: 524-381-726
127.0.0.1 - - [12/Dec/2022 13:22:14] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2022 13:22:16] "GET /?num=&btnnum=Submit HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2022 13:22:29] "GET /?num=11&btnnum=Submit HTTP/1.1" 200 -
```

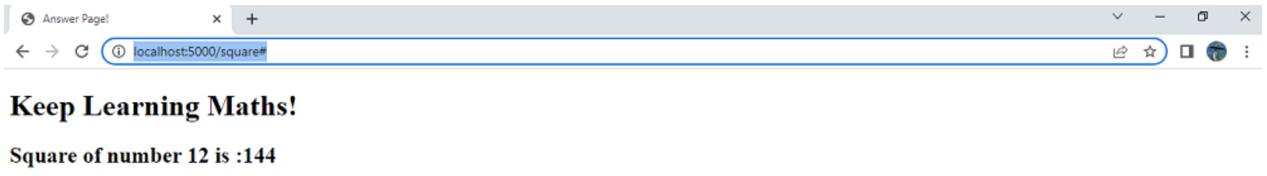

Welcome to the Maths page!

Logic shapes every choice of our daily lives.
Logical thinking enables someone to learn and make decisions that affect their way of life. !

Enter a number : 12 Submit

The user enters a number when the page is rendered for URL 'localhost:5000/square'

On clicking the Submit button, the data is posted back, to the server, and, the result page is rendered. Please note, the value entered, is not visible in the URL now, as the method used, is POST.


Keep Learning Maths!

Square of number 12 is :144

Related Articles: Difference between GET and POST

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!

P phad... 6

Previous Article

Flask App Routing

Next Article

Flask - Variable Rule

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

Flask HTTP methods, handle GET & POST requests

In this article, we are going to learn about how to handle GET and POST requests of the flask HTTP methods in Python. HTTP Protocol is necessary for data...

6 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

Handling 404 Error in Flask

Prerequisite: Creating simple application in Flask A 404 Error is showed whenever a page is not found. Maybe the owner changed its URL and forgot to...

4 min read

Python | Using for loop in Flask

Prerequisite: HTML Basics, Python Basics, Flask It is not possible to write front-end course every time user make changes in his/her profile. We use a template...

3 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

Article Tags : Python Technical Scripter python Technical Scripter 2022

Practice Tags : python python