

# Lab # 22

## Build Your Python Flask Application



### Objective:

Getting Familiar HTTP Methods and Templates.

### Theory:

Flask supports multiple HTTP methods for client-server communication:

Method	Description
GET	Sends data without encryption via URL
POST	Sends form data securely (not cached)
HEAD	Same as GET but without response body
PUT	Replaces resource at a given URL
DELETE	Deletes resource from server

### 1. Understanding HTTP Methods in Flask

#### Steps to follow

Step 01: Open the terminal (Bash/CMD) and open the flask\_lab folder.

- **cd flask\_lab**

Step 02: Open Notepad / VS Code and write the **Program Code** and Save it.

- **notepad app.py**

```
from flask import Flask, request
app = Flask(__name__)
@app.route('/data', methods=['GET', 'POST'])
def data():
    if request.method == 'GET':
        return "This is a GET request"
    else:
        return "This is a POST request"
if __name__ == '__main__':
    app.run(debug=True)
```

## Lab # 22

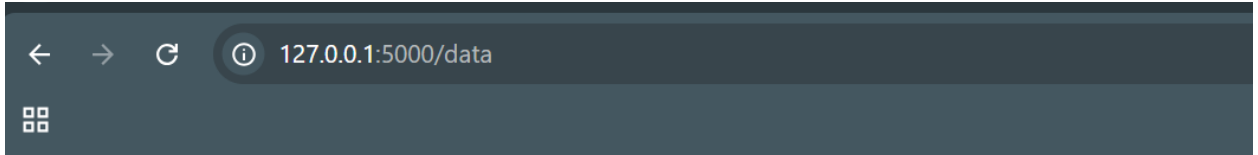
**Step 03:** Run the application

- **python app.py**

**Note:** If successful, you will see: Running on <http://127.0.0.1:5000/>

**Step 04:** Test GET Request (Browser)

- <http://127.0.0.1:5000/data>



This is a GET request

**Step 04:** Test POST Request

1. Browsers cannot directly send POST without a form.
2. Open **another** Command Prompt / Terminal window
3. Run this command: `curl -X POST http://127.0.0.1:5000/data`

```
De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ curl -X POST http://127.0.0.1:5000/data
This is a POST request
De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ |
```

## 2. Serving HTML Templates in Flask

Flask uses a **templates** directory to store HTML files, rendered using `render_template()`.

Directory Structure:

```
project/
|— app.py
|— templates/
|   └— index.html
```

**Step 1:** Created Project Folder

- **mkdir flask\_template\_lab**

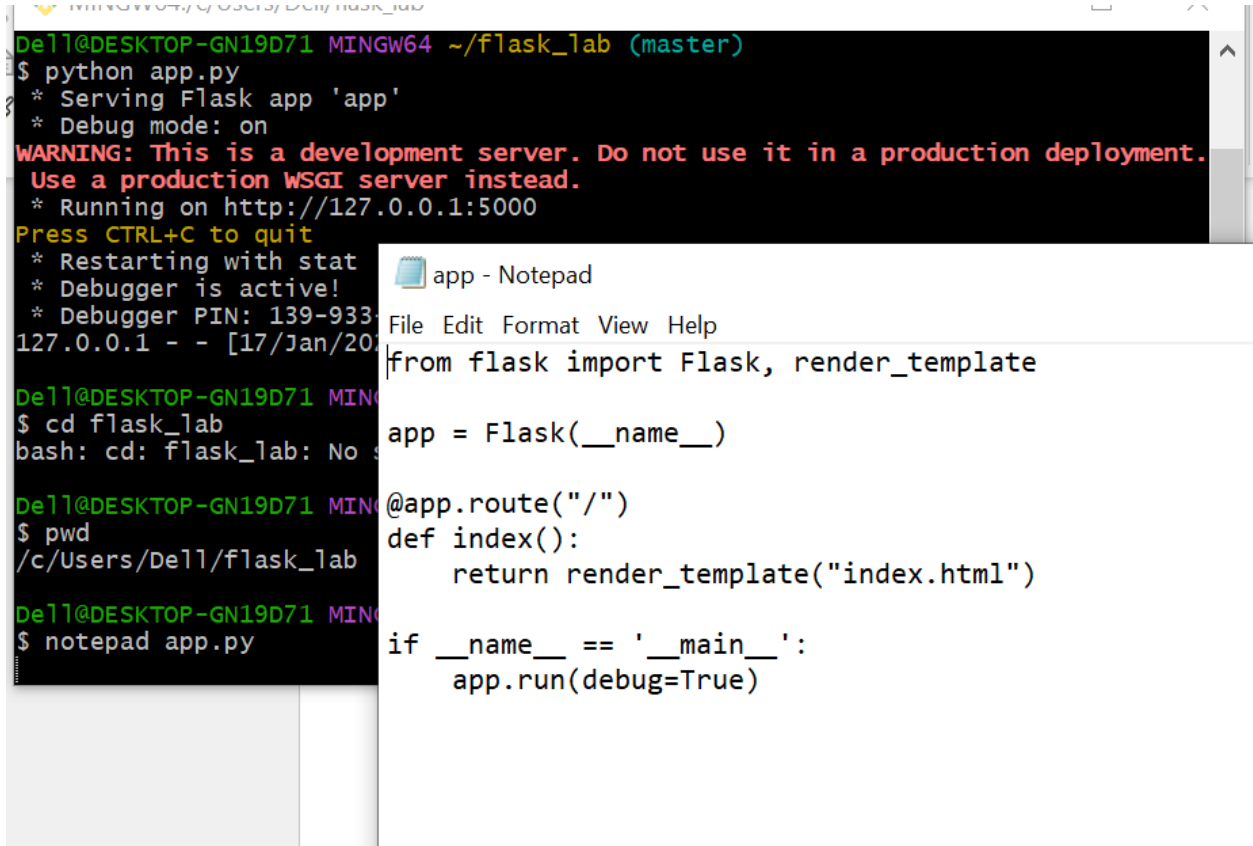
## Lab # 22

**Step 02:** Open the Folder

- **cd flask\_template\_lab**

**Step 03:** Open VS Code / notepad /any editor and write a Program Code, save it.

- **notepad app.py**



The screenshot shows a terminal window on the left and a Notepad editor on the right. The terminal window displays the following commands and output:

```
De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 139-933
127.0.0.1 - - [17/Jan/2020:12:00:00] "GET / HTTP/1.1" 200 127.0.0.1

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab
$ cd flask_lab
bash: cd: flask_lab: No such file or directory

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab
$ pwd
/c/Users/De11/flask_lab

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab
$ notepad app.py
```

The Notepad editor shows the following code:

```
app - Notepad
File Edit Format View Help
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

if __name__ == '__main__':
    app.run(debug=True)
```

### Program Code

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/")
def index():
    return render_template("index.html")
if __name__ == '__main__':
    app.run(debug=True)
```

**Step 04:** Create **Templates Folder** inside the **flask\_template\_lab**

- **mkdir templates**

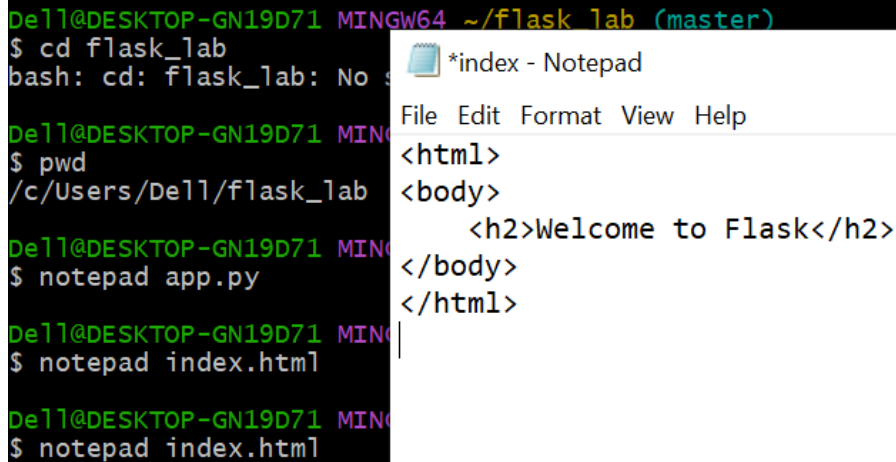
## Lab # 22

Step 05: Open the **templates** folder

- **cd templates**

Step 06: Create a new file **index.html** and write a **Program Code** and save it.

- **notepad index.html**



The screenshot shows a terminal window on the left and a Notepad window on the right. The terminal window shows the following commands and output:

```
De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ cd flask_lab
bash: cd: flask_lab: No such file or directory

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ pwd
/c/Users/De11/flask_lab

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ notepad app.py

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ notepad index.html

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ notepad index.html
```

The Notepad window shows the following HTML code:

```
*index - Notepad

File Edit Format View Help

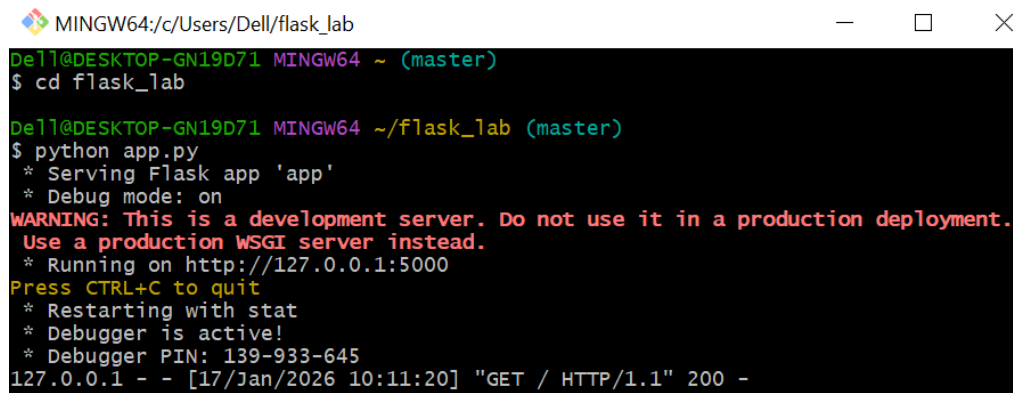
<html>
<body>
    <h2>Welcome to Flask</h2>
</body>
</html>
```

### Program Code

```
<html>
<body>
    <h2>Welcome to Flask Students!</h2>
</body>
</html>
```

Step 07: Run the app

- **python app.py**



The screenshot shows a terminal window with the following output:

```
MINGW64:/c/Users/Dell/flask_lab

De11@DESKTOP-GN19D71 MINGW64 ~ (master)
$ cd flask_lab

De11@DESKTOP-GN19D71 MINGW64 ~/flask_lab (master)
$ python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 139-933-645
127.0.0.1 - - [17/Jan/2026 10:11:20] "GET / HTTP/1.1" 200 -
```

## Lab # 22

**Step 08: Open the Browser and visit the link:**

- **<http://127.0.0.1:5000/>**

**Output**



**Welcome to Flask Students!**

### **Task:**

**Create your first Python Flask application that demonstrates the use of HTTP methods and HTML templates.**

Your application should:

1. Use **Flask** to create a web server.
2. Create one route (/) that uses the **GET** method to display an HTML page using the **templates** folder.
3. Create one route (/data) that handles both **GET** and **POST** requests.
4. Display a different message for GET and POST requests.
5. Run the application on the default Flask server (localhost:5000).