SECTION-13: Building Web Applications Using Flask

Lec-65

Creating our first Flask App :-

Always try to create main file as → main.py

Folder → static → It holds all the static files
- It will contain CSS and JavaScript File
- You can store those files which you want to be publicaly available.

name should be as it is

templates → It will hold HTML templates.

Boilerplate of flask template :-

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p> Hello, World! </p>"

app.run(debug = True)
```

, render_template

render_template("index.html")

file name

What did that code do?

1. First we imported the Flask class. An instance of this class will be our WSGI application.

2. Next we create an instance of this class. The first argument is the name of the application's module or package. `__name__` is a convenient shortcut for this that is appropriate for most cases. This is needed so that Flask knows where to look for resources such as templates and static files.

3. We then use the route() decorator to tell Flask what URL should trigger our function.

4. The function returns the message we want to display in the user's browser. The default content type is HTML, so HTML in the string will be rendered by the browser

---

Lec-66

Jo websites pe scan hote unko tap karne par agar hum chhate har usko highlight karna (change) ⟶ toh <li> ke andar anchor tag m class="nav link" Jaha likha hai waha aage active likh do class="nav link active".

Jaise har heading jo top par or woh upar ke navigation bar se bhaut kharib ho, toh agar usko dur karna ho toh hum starting tag

ke andar jo class hoti hai usme "container" ke
uage my-4 likh skte hai ⇒ class = "container my-4"

Lec-61

For Serving Static Files (Downloading files through an website)

→ Step ① Create Boilerplate of flask template by importing render_template.

② Create static and templates Folder

③ Inside static upload all the files which you want to download through website.

④ Inside templates folder create index.html

⑤ In index.html inside <body> tag, we can add files name which we want to download

⑥ Video 1 Download

↳ Use this, when we want to open an new page in the current open page. [means it will not help in direct downloading]

⑦ Video 1 Download

↳ This will instantly download the file.

Lec-68

Handling Forms in Flask

We have mainly two types of main request when we are working with web development i.e., **Get requests** and **Post Requests**.

Get requests :→ It is a very simple request where you simply request something from the server and it gives you the web page.

Post requests :→ In post requests you are going to request something to the server along with some payload.

```
from flask import Flask, render_template, request

app= Flask(__name__)

@app.route("/", methods= ["GET", "POST"])
def hello_world():
    print(request.method) # It will tell which type of method
                          is used (GET or POST)
    print(request.form) # It will return an immutable dict
    if(request.method == "POST"):
        with open("File.txt", "w") as f:
            f.write(f"Name of the user: {request.form['Name']}
                    Email of the user: {request.form['Email']}
                    ")
    return render_template("contact.html")
```

contact.html

yeh use karne par Dynamic data jo ke andar show hoga, otherwise nahi hoga

```
<body>
    <div>
        <form action="/" method="post">
            <input name="Name" type="text" placeholder="Enter Your Name">
            <input name="Email" type="text" placeholder="Enter Your Email">
            <input type="submit" value="submit">
        </form>
    </div>
</body>
```

Lec-69

Jinja 2 Template → It is used for passing variables to the index.html

We can still pass the variable to index.html without using Jinja 2 template

→ main.py

```
def hello_world():
    marks = {
        "Saurabh": 89,
        "Raghav": 56,
        "Shaurya": 77,
        "Shashank": 89
    }
    return render_template
        ("index.html", marks=marks)
```

→ index.html

```
<body>
  <ul>
    <li> {{marks['Saurabh']}}
                    </li>
  <li>{{marks['Raghav']}}</li>
  </ul>
</body>
```

Lec-70

Comments, Loops & Conditional in Jinja 2

Comments in Jinja 2 → {# This is a comment #}

For loop → {% for key in marks.key() %}
(for accessing key-value pair) (name of the dictionary)
 The marks of the {{key}} is {{marks.get(key)}}
{% endfor %}

For getting the syntax of anything in Jinja 2 Template → j. (name of that thing)

Lec-71

Template Inheritance

→ Create an base.html as a file inside the template folder and write all your code there.

→ For starting the process of inheritance create blocks like this in the base.html
{% block title %} {% endblock %}
(any name you want to give)

→ For implementing inheritance using "base.html" inside that file write {% extends "base.html" %} . then use it by writing those same blocks with your desired content
{% block title %} Home {% endblock %}

→ This is not a part of template inheritance.

For Creating clickable links of the .html files from the main page (suppose, we have about.html and contact.html

```
<body>
    <nav>
        <a href="/"><span>Home</span></a>
        <a href="/about"><span>About</span></a>
        <a href="/contact"><span>Contact</span></a>
    </nav>
</body>
```

→ It's example in main.py

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

Lec=72

Changing static url path:

Create the static folder, in that create any file (suppose we are creating style.css)

Now for accessing this file, we can simply type this in the browser 127.0.0.1:5000/static/style.css

But, now if we want to change the url, we can do it by changing this in the main.py

```
app = Flask(__name__, static_url_path="/anyname")
```

For changing static folder location

Suppose you have created a new folder (assets) and you want to access the file (anotherstyle.css) inside it, you can't do it directly by typing 127.0.0.1:5000/assets/anotherstyle.css, it will throough an error

But you can do it by, changing static folder location.

app = Flask(__name__, static_folder = "_____")

↑ name of that folder

We can also use both static folder and static url path together.

app = Flask(__name__, static_folder = "_____", static_url_path = "/_____")

Now, suppose we have created an index.html, inside it

<head>

static_url_path, static_folder = "assets"

<link rel="stylesheet" href="/static/anotherstyle.css">

↳ This will work, but we have to change it manually as we change static folder or url_path inside main.py

To fix this problem, we have: function called url_for( )

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

</head>

So url_for is a function in flask that we can use in our templates, which will automatically adjust your static_folder and static_url_path.

Lec-73 (Query Parameter in Flask)

→ main.py

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/")
def hello_world():
    name = "Gautam"
    token = "1000"
    return render_template("index.html", name=name, token=token)
app.run(debug=True)
```

→ index.html

```
<body>
    <h1> Hey Welcome Back {{name}} </h1>
    <p>
        You have left only {{token}} tokens !
    </p>
</body>
```

Doing this till now will simply print the respecting things, so now what does we mean by Query parameter

By Default (URL) ⟹ 127.0.0.3:3000

If we pass Query Parameter inside the URL ⟹ 127.0.0.3:3000?name=Harry&token=500

→ So now, if we want to process this, we have to make changes inside index.html, First we import request from flask and we have to use request.args

```
from flask import Flask, render_template, request
def hello_world():
    name = "Gautam"
    token = "1000
    if "name" in request.args.keys():
        name = request.args["name"]
    if "token" in request.args.keys():
        token = request.args["token"]
    return render_template("index.html", name=name, token=token)
```

* Never use confidential information in the query parameter.
* The best use case is to pass access token in the query parameter, because they are valid for very limited time

LCC—74 (Creating API's in Flask using jsonify)

If we want to expose json as an API. Then we have to import jsonify.

```
from flask import Flask, jsonify
app= Flask(__name__)
@app.route("/")
def json():
    marks = {
        "Harry": 56,
        "Rohan": 67,
        "Aakash": 78,
        "Shubham": 100,
        "Reena": 67
            }
    return jsonify(marks)
app.run(debug=True)
```

Lot of developers use it to serve their data through API's

Lec-75 (Message Flashing in Flask)

For message Flashing You first have to ⟹ import flash message flashing is used to show temporary messages to the user in flask.

→ main.py

```
from flask import Flask, reder_template, flash
app= Flask(__name__)
app.secret_key = "ghdlfguptm12098"  # message flashing works only if you have a secret key set in your flask application
#Flashing in flask works on sessions.

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

@app.route('/logout')
def logout():
        flash("You have been logged out!", "sucess")
        return render_template("index.html")
app.run(debug=True)
```

→ index.html

```
<body>
    {% with messages=get_fashed_messages(with_categories=true)%}
        {% if messages %}
            {% for category, message in messages %}
                <li class="{{category}}">"{{message}}"</li>
            {% endfor %}
        {% end if %}
    {% endwith %}
    Hey you Just logged in
    <a= href="/logout">Logout </a>
</body>
```

With the code of index1.html, you will not be able to automatically go back on the intial page. For that, you have to do following changes:-

→ main2.py → add this in the code of main1.py

```
@.app.route('/logout')
def logout():
    flash("You have been logged out!", "sucess")
    return render_template("logout.html")

app.run(debug=True)
```

→ create index2.html

Shift the things from <body> tag to <script> tag

```
<script>
    {% with ---- ---- ---- ---- %}
        {% if ---- %}
            {% for ---- ---- ---- %}
                alert('{{message}}')
            {% ---- ---- ---- %}
        {% ---- ---- %}
    {% ---- ---- %}
</script>
```

Now, create an seperate logout.html → Created for redirecting ourselve to the home default page

Copy whole index2.html, then add location.href = "/" before the ending of closing tag of </script>

↑ write your default page

LEC-78

Using Flask for uploading Files

```
import os
from flask import Flask
from werkzeug.utils import secure_filename

UPLOAD_FOLDER = '/path/to/the/uploads'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```