**Flask Basics: Routing, Requests, and Templates**

**1. Introduction to Flask**

Flask is a lightweight **Python web framework** used to build web applications. It is simple and easy to learn, making it a great choice for beginners. Flask follows the **WSGI (Web Server Gateway Interface)** standard and is built on **Werkzeug** and **Jinja2**.

**Why Flask?**

* Minimal and lightweight
* Easy to set up and use
* Supports routing, templates, and request handling
* Extensible with Flask extensions (like SQLAlchemy, Flask-Login, etc.)

**2. Setting Up Flask**

Before starting, install Flask using:

pip install flask

Create a simple **Flask app**:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return "Hello, Flask!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

* @app.route('/') → Defines the route (URL path)
* app.run(debug=True) → Runs the server with debug mode enabled

Run the script, and visit http://127.0.0.1:5000/ in your browser to see **"Hello, Flask!"**.

**3. Routing in Flask**

**Routing** maps URLs to Python functions. Flask provides decorators (@app.route) to define routes.

**Static Routes**

@app.route('/about')

def about():

return "This is the About page"

Visit http://127.0.0.1:5000/about to access it.

**Dynamic Routes**

Dynamic routes accept parameters.

@app.route('/user/<name>')

def greet(name):

return f"Hello, {name}!"

Example:

* http://127.0.0.1:5000/user/Marco → **"Hello, Marco!"**

**Specifying Data Types in Routes**

@app.route('/post/<int:post\_id>')

def show\_post(post\_id):

return f"Post ID: {post\_id}"

This ensures post\_id is always an **integer**.

**4. Handling Requests in Flask**

Flask supports **GET** and **POST** requests.

**GET Requests (Default)**

from flask import request

@app.route('/search', methods=['GET'])

def search():

query = request.args.get('q', 'Default Query')

return f"Search results for: {query}"

Example:

* http://127.0.0.1:5000/search?q=flask → **"Search results for: flask"**

**POST Requests**

@app.route('/submit', methods=['POST'])

def submit():

data = request.form['username']

return f"Form submitted by {data}"

Use a **form** in HTML to send a POST request:

<form action="/submit" method="post">

<input type="text" name="username">

<input type="submit" value="Submit">

</form>

**5. Using Templates (Jinja2 in Flask)**

Flask uses **Jinja2** as its templating engine to render HTML dynamically.

**Creating a Template**

1. Create a folder named templates
2. Inside templates, create index.html:

<!DOCTYPE html>

<html>

<head>

<title>Flask Template</title>

</head>

<body>

<h1>Welcome, {{ name }}!</h1>

</body>

</html>

1. Modify app.py to render it:

from flask import render\_template

@app.route('/welcome/<name>')

def welcome(name):

return render\_template('index.html', name=name)

Visit http://127.0.0.1:5000/welcome/Marco → Displays "Welcome, Marco!"

**Using Jinja2 Variables & Loops**

**Passing multiple values:**

@app.route('/users')

def users():

user\_list = ['Alice', 'Bob', 'Charlie']

return render\_template('users.html', users=user\_list)

**Modify ``:**

<ul>

{% for user in users %}

<li>{{ user }}</li>

{% endfor %}

</ul>

Output:

* Alice
* Bob
* Charlie

**6. Connecting Flask to SQLAlchemy**

SQLAlchemy is an ORM (Object-Relational Mapper) for working with databases in Flask.

**Installing SQLAlchemy**

pip install flask-sqlalchemy

**Configuring the Database**

Modify app.py:

from flask\_sqlalchemy import SQLAlchemy

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///users.db'

db = SQLAlchemy(app)

**Defining a Model**

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(80), nullable=False)

def \_\_repr\_\_(self):

return f"<User {self.name}>"

**Creating the Database**

db.create\_all()

**Adding Data**

new\_user = User(name='Marco')

db.session.add(new\_user)

db.session.commit()

**Retrieving Data**

users = User.query.all()

for user in users:

print(user.name)

**7. Summary**

By now, you've learned: ✔ Flask app setup & routing (static, dynamic)  
✔ Handling requests (GET, POST)  
✔ Rendering HTML templates (Jinja2)  
✔ Using Flask-SQLAlchemy for databases

**8. Next Steps**

👉 **Practice!** Try building a small **login form** or **guestbook app**.  
👉 **Coming Up Next:** Handling User Authentication (Sessions, Login, Logout) 🚀