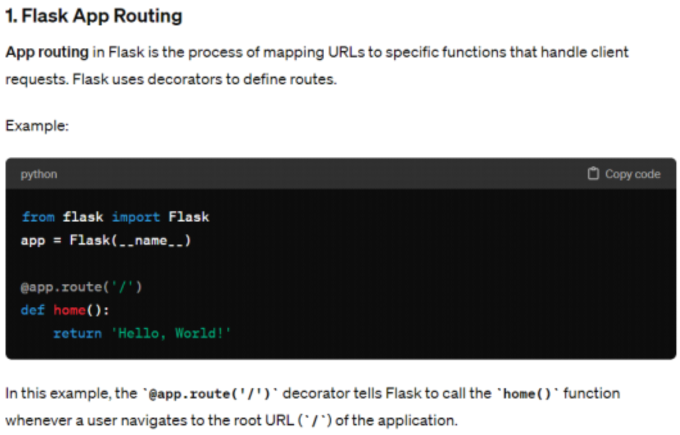
# Flask for Data Science

09 March 2024 12:04

Can quickly develop a web application



### 1. Flask App Routing

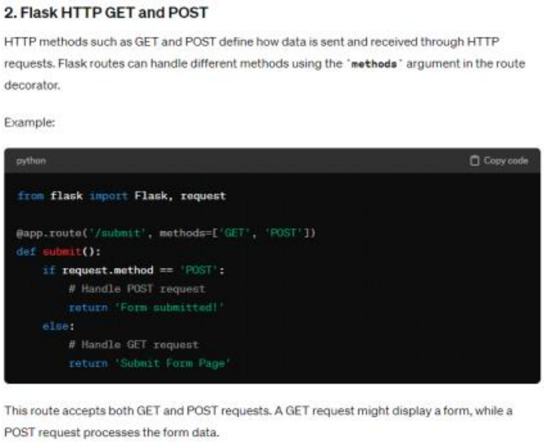
**App routing** in Flask is the process of mapping URLs to specific functions that handle client requests. Flask uses decorators to define routes.

Example:

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return 'Hello, World!'
```

In this example, the `` `@app.route('/')` `` decorator tells Flask to call the `` `home()` `` function whenever a user navigates to the root URL (`` `/` ``) of the application.



### 2. Flask HTTP GET and POST

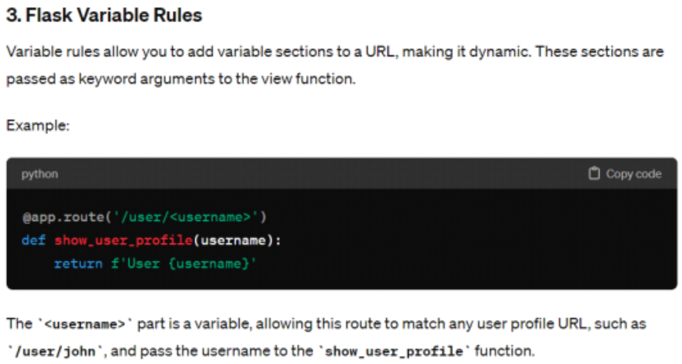
HTTP methods such as GET and POST define how data is sent and received through HTTP requests. Flask routes can handle different methods using the `` `methods` `` argument in the route decorator.

Example:

```python
from flask import Flask, request

@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        # Handle POST request
        return 'Form submitted!'
    else:
        # Handle GET request
        return 'Submit Form Page'
```

This route accepts both GET and POST requests. A GET request might display a form, while a POST request processes the form data.



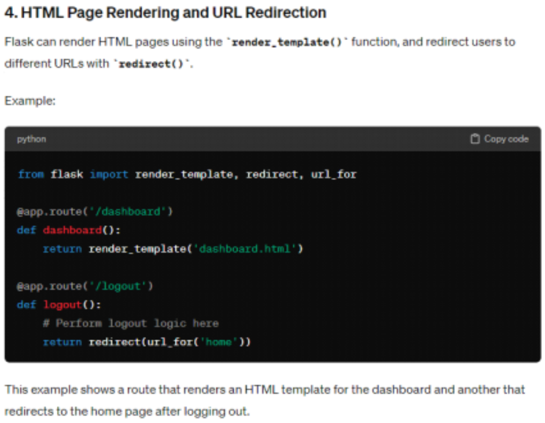
### 3. Flask Variable Rules

Variable rules allow you to add variable sections to a URL, making it dynamic. These sections are passed as keyword arguments to the view function.

Example:

```python
@app.route('/user/<username>')
def show_user_profile(username):
    return f'User {username}'
```

The `` `<username>` `` part is a variable, allowing this route to match any user profile URL, such as `` `/user/john` ``, and pass the username to the `` `show_user_profile` `` function.



### 4. HTML Page Rendering and URL Redirection

Flask can render HTML pages using the `` `render_template()` `` function, and redirect users to different URLs with `` `redirect()` ``.

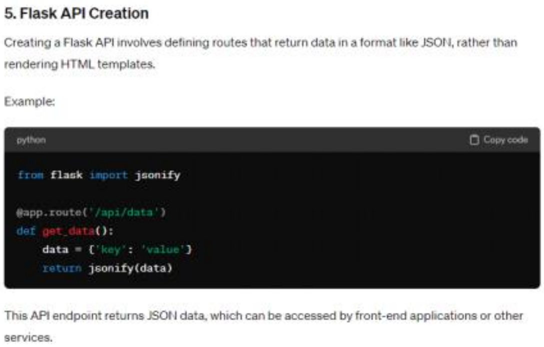
Example:

```python
from flask import render_template, redirect, url_for

@app.route('/dashboard')
def dashboard():
    return render_template('dashboard.html')

@app.route('/logout')
def logout():
    # Perform logout logic here
    return redirect(url_for('home'))
```

This example shows a route that renders an HTML template for the dashboard and another that redirects to the home page after logging out.



### 5. Flask API Creation

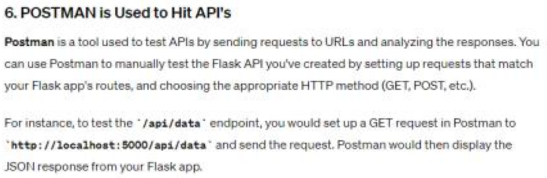
Creating a Flask API involves defining routes that return data in a format like JSON, rather than rendering HTML templates.

Example:

```python
from flask import jsonify

@app.route('/api/data')
def get_data():
    data = {'key': 'value'}
    return jsonify(data)
```

This API endpoint returns JSON data, which can be accessed by front-end applications or other services.



### 6. POSTMAN is Used to Hit API's

**Postman** is a tool used to test APIs by sending requests to URLs and analyzing the responses. You can use Postman to manually test the Flask API you've created by setting up requests that match your Flask app's routes, and choosing the appropriate HTTP method (GET, POST, etc.).

For instance, to test the `` `/api/data` `` endpoint, you would set up a GET request in Postman to `` `http://localhost:5000/api/data` `` and send the request. Postman would then display the JSON response from your Flask app.

POSTMAN is used to hit API's

Through this flow, Flask allows for the creation of dynamic web applications and APIs, capable of handling web pages, user input, and data interchange with front-end technologies or other web services.