Flask Session 1

Mohamed Emary

April 2, 2025

1 Pre-Session

- Request/Response Lifecycle
- Request contents (headers, body, etc.)
- When we can't use request/response?
- Middleware Definition
- What does synchronous/asynchronous mean?
- Topics you need to know as a software engineer

2 Session Notes

2.1 Flask vs Django

- Flask is a micro-framework, while Django is a full-fledged framework.
- Flask is more flexible and allows you to create your own structure, while Django follows a specific structure.

2.2 Example simple Flask app

```python
from flask import Flask

app = Flask(__name__)


@app.route("/")
def hello():
    return "<h1>hello world</h1>"
```

In the code you may notice the function `hello` is not called directly, but it is called when the route `/` is accessed. This is because Flask uses a decorator to register the function as a route handler. The `@app.route("/")` decorator tells Flask to call the `hello` function when the root URL is accessed.

2.3 Running the Flask app

To run your app you can use the command:

```
flask --app app run --reload
```

This will start a development server on the file `app.py` and reload it automatically when you make changes to the code.

You can also specify the host and port by using the `--host` and `--port` options:

```
flask --app app run --reload --host 0.0.0.0 --port 5000
```

This will start the Flask application and make it accessible from any device on the local network. You can access it by navigating to `http://<your-ip-address>:5000` in your web browser.

The part `--host 0.0.0.0` allows the application to be accessible from any IP address on the local network.

2.4 Dynamic Routing & Request Methods

To create a dynamic route, you can use angle brackets in the route definition. For example:

```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/user/<name>", methods=["GET", "POST"])
def show_user_profile(name):
    """
    Purpose: Displays name page
    """
    users = {
        "mohamed": "mohamed@iti.com",
        "ahmed": "ahmed@iti.com",
        "mahmoud": "mahmoud@iti.com",
    }

    if request.method == "GET":
        return f"<h1>Hello {name}</h1>"
    elif request.method == "POST":
        if name in users:
            return f"<h1>Welcome back {name}</h1>"
        else:
            return f"<h1>Welcome Unknown user {name}</h1>"
    return f"<h1>Method not allowed</h1>"
```

In this example, any `GET` or `POST` request to `/user/<name>` will call the `show_user_profile` function, and the value of `<name>` will be passed as an argument to the function.

Templates vs APIs?

Differences between templates and APIs:

- Templates are used to render HTML pages, while APIs are used to return data in a

structured format (like JSON or XML).

- Templates are usually rendered on the server-side, while APIs can be consumed by client-side applications.
- Templates are often used for web applications, while APIs can be used for any type of application (web, mobile, etc.).
- Templates are usually tied to a specific view or page, while APIs are more flexible and can be used for multiple purposes.

When to use templates vs APIs:

- Use templates when you want to render HTML pages on the server-side and send them to the client.
- Use APIs when you want to return data in a structured format (like JSON or XML) that can be consumed by client-side applications or other services.
- Use templates when you want to create a web application with a specific structure and design.
- Use APIs when you want to create a service that can be consumed by multiple clients (web, mobile, etc.).
- Use templates when you want to create a user interface that is tightly coupled with the server-side logic.
- Use APIs when you want to create a decoupled architecture where the client and server can evolve independently.

2.5 Using templates

To use templates in Flask, you need to create a folder named `templates` in the same directory as your Flask app. Inside this folder, you can create HTML files that will be used as templates.

Then in your Flask app, you can use the `render_template` function to render the HTML files. For example:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/")
def hello():
    return render_template("index.html")
```

In this example, when you access the root URL, Flask will look for a file named `index.html` in the `templates` folder and render it.

And to pass variables to the template, you can do it like this:

```
@app.route("/template/<name>")
def template(name):
    return render_template("index.html", name=name, current_app=current_app)
```

- `name=name` is the variable you want to pass to the template `index.html`.
- `current_app=current_app` is used to access the current application context.

Inside `index.html` file, you can access the variable `name` like this:

```
<p>{{ name }}</p>
```

This will display the value of the `name` variable in the HTML page.