2.2. Student Handout

Flask: Jinja2 Templating Engine and Static Files

Overview

This handout provides a concise guide to using the Jinja2 templating engine and handling static files in Flask. These concepts are crucial for creating dynamic and visually appealing web applications.

---

1. Jinja2 Templating Engine

What is a Template?

Templates allow you to create a structure with placeholders that can be dynamically replaced with data from your Flask application.

How Does Jinja2 Work?

Jinja2 uses special syntax for placeholders and logic in HTML files.

Example 1: Basic Placeholder

```
<h1>Hello, {{ name }}!</h1>
```

Example 2: Conditional Statement

```
{% if user %}
<p>Welcome, {{ user }}!</p>
{% else %}
<p>Welcome, Guest!</p>
```

```
{% endif %}
```

Example 3: Looping Through a List

```
<ul>

{% for item in items %}

<li>{{ item }}</li>

{% endfor %}

</ul>
```

---

2. Creating Dynamic Content Using Templates

Passing Data from Flask to Templates

Use the `render_template()` function to render a template and pass data to it.

Example 1: Passing a Single Variable

```
@app.route('/')

def home():

return render_template('index.html', name='Rahul')
```

Example 2: Passing Multiple Variables

```
@app.route('/profile')

def profile():
```

```
return render_template('profile.html', name='Rahul', age=25)
```

Example 3: Passing a List

```
@app.route('/items')

def items():

return render_template('items.html', items=['Apple', 'Banana', 'Cherry'])
```

---

3. Template Inheritance (Base Templates)

What is Template Inheritance?

Template inheritance allows you to define a base template with common elements and let other templates inherit from it.

How Does Template Inheritance Work?

Example 1: Base Template (`base.html`)

```
<!DOCTYPE html>

<html lang="en">

<head>

<title>{% block title %}My Website{% endblock %}</title>

</head>

<body>

<header>

<h1>Welcome to My Website</h1>

</header>
```

```
<div>
{% block content %}{% endblock %}
</div>
<footer>
<p>Footer content here</p>
</footer>
</body>
</html>
```

Example 2: Home Page Template (`home.html`)

```
{% extends 'base.html' %}

{% block title %}Home Page{% endblock %}

{% block content %}
<p>This is the home page content.</p>
{% endblock %}
```

Example 3: About Page Template (`about.html`)

```
{% extends 'base.html' %}

{% block title %}About Us{% endblock %}
```

```
{% block content %}

<p>This is the about page content.</p>

{% endblock %}
```

---

4. Handling Static Files (CSS, JavaScript, Images)

What are Static Files?

Static files include CSS, JavaScript, and images that do not change dynamically.

Organizing Static Files in Flask

Example 1: Directory Structure

```
/static

/css

style.css

/js

script.js

/images

logo.png
```

Linking Static Resources in Templates

Example 2: Linking a CSS File

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

Example 3: Linking a JavaScript File

```
<script src="{{ url_for('static', filename='js/script.js') }}"></script>
```

---

5. Creating a Basic Webpage Layout

Step-by-Step Guide

Step 1: Create the Base Template (`base.html`)

```
<!DOCTYPE html>

<html lang="en">

<head>

<title>{% block title %}My Website{% endblock %}</title>

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

</head>

<body>

<header>

<h1>Welcome to My Website</h1>

</header>

<nav>
```

```
<ul>

<li><a href="/">Home</a></li>

<li><a href="/about">About</a></li>

</ul>

</nav>

<div>

{% block content %}{% endblock %}

</div>

<footer>

<p>Footer content here</p>

</footer>

</body>

</html>
```

Step 2: Create the Home Page Template (`home.html`)

```
{% extends 'base.html' %}


{% block title %}Home Page{% endblock %}


{% block content %}

<p>This is the home page content.</p>

{% endblock %}
```

Step 3: Create the About Page Template (`about.html`)

```
{% extends 'base.html' %}


{% block title %}About Us{% endblock %}


{% block content %}
<p>This is the about page content.</p>
{% endblock %}
```

Step 4: Create the CSS File (`style.css`)

```
body {
font-family: Arial, sans-serif;
}


header {
background-color: #f8f9fa;
padding: 10px;
text-align: center;
}


nav ul {
list-style-type: none;
```

```
padding: 0;

}



nav ul li {

display: inline;

margin-right: 10px;

}



footer {

background-color: #f8f9fa;

padding: 10px;

text-align: center;

}
```

Step 5: Run the Flask Application

```
from flask import Flask, render_template



app = Flask(__name__)



@app.route('/')

def home():

return render_template('home.html')
```

```
@app.route('/about')

def about():

return render_template('about.html')



if __name__ == '__main__':

app.run(debug=True)
```

Conclusion

This guide covered:

- **Jinja2 Templating Engine**: Creating dynamic content using templates.
- **Template Inheritance**: Reusing a base template across multiple pages.
- **Static Files**: Organizing and linking CSS, JavaScript, and images.
- **Basic Webpage Layout**: Combining templates and static files to create a simple website.

For further questions or clarifications, feel free to reach out.