Templates

Learning Objectives

After this lesson, you will be able to:

- Create a template HTML document.
- Pass variables to a template HTML document via a Flask app.

A Million Copies

- Don’t you hate it when you have to repeat yourself?
- What if you had a website with 10 pages that were almost the same?
- Would you code them all from scratch?

We Do: Your `index.html` Page

- Any route can use an `html` page.

- Try it!
  - In your `my_website.py`, set the `/randnum/<inte>` and `/` routes to both `render_template("index.html")`.

- What if we want them to display a different heading?

- Do we need to rewrite the whole file?

No! That’s What Templates Are For!

We use templates to:

- Write one HTML file.
- Pass it variables.
- Transfer info from Flask to HTML.

As well as for one important design reason:

- We can separate data from how we present data to users.

Jinja2

- One of the most widely used template engines for Python.
- Used in places that you might have visited already, like Instagram or NPR.

Why Jinja2?

Jinja2 has some really powerful features that web design folks want to take advantage of:

- Template inheritance
    - Like class inheritance!
- HTML escaping
    - Stops some hacking attacks (XSS and SQL injection).
- Speed and efficiency
    - Optimized Python code.
- Flexibility and extensibility
    - We can add our own filters and functions.

Expanding on Our `index.html`

- We'll send a `greeting` variable into our `index.html` from both routes.
- The routes will display different things!

Adding Templates

- Remember `import render_template`?
- This is the function that Flask uses to… you guessed it: Render our template(s)!

Edit `index.html`

- Change the `<h1>` to be `{{ greeting }}`.

```
...
   <body>
      <h1>Hello {{ name }}!</h1>
            <p>If music be the food of love, play on!</p>
    ....
```

Templating Syntax With Jinja

- Recognize the `{{}}`?
- In Jinja, **templates** are rendered with double curly brackets (`{{ }}`).
- **Statements** are rendered with curly brackets and percent signs (`{% %}`).
    - A use case here is passing in logic like:

```
{% if name == 'kevin' %}
# Do the thing
{% else %}
# Do all the other things.
```

Rendering a Template in Flask

Let's change our `my_website.py` accordingly:

```
@app.route('/')
def home():
  return render_template("index.html", greeting="Hello World!")

...

@app.route('/shownum/<inte>')
def shownum(inte):
  my_greeting = "Your number is " + str(inte)
  return render_template("index.html", greeting=my_greeting)
```

Try it!

- Check out: `http://localhost:5000`.
- Then: `http://localhost:5000/shownum/26`.

Do your other routes still work?

Knowledge Check: Discussion

What two arguments did we pass into the `render_template` function?

What’s one reason we use templates?

Your Turn!

- Create a new Flask app, `shakespeare.py`.
- Create a new template HTML file, `hello.html`.
  - It will display a paragraph with a parameter `poem` in it.
- Render it from the index endpoint.
- Remember calling in variables from the last lesson?
  - Have your Flask app read in the poem saved in `hi.txt`, then pass that to the `hello.html` template to display.
- Launch your Flask app and check the results!

Template Solution

```
<!doctype html>

<html>

<head>

    <meta charset="utf-8">

    <title>Shakespeare</title>

</head>

<body>

    <p>{{text}}</p>

</body>

</html>
```

Python Solution

```
from flask import Flask, render_template
import os # Note the new import — to be in the file system.


app = Flask(__name__)


file_path = '.'


with open(os.path.join(file_path, 'hi.txt')) as f:
    the_text = f.read()


@app.route('/') # When someone goes here...
```

Summary

- Jinja:
  - A popular templating engine.
  - Templates save us time and abstract presentation from data.
- Template fun:
  - We can pass variables into the template from the Flask app and the URL.