Tuesday: Introduction, Template Inheritance; Adding Bootstrap

Introduction

We have seen how useful templating is and how we can add dynamic information and logic to our templates. Now we will consider how to customize our templates.

We will learn how to use Bootstrap framework, How to customize our the look of our templates using CSS and How to add Links to our templates.

Template Inheritance

Notice how every time we create a new template we have to define the HTML structure. In a small application like ours it does not seem like a lot of work. But in a large application with dozens of pages it can be very tedious . Jinja allows for **template inheritance**. That is we can inherit some properties of a template in other child (or derived) templates.

Lets see what that looks like. We start by creating a new template file *base.html*

base.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        {% if title %}
        <title> {{ title }}</title>
        {% else %}
        <title> Welcome to the best Movie Review website </title>
        {% endif %}
    </head>
    <body>
        <!-- Content block -->

        {% block content %}

        {% endblock%}
    </body>
</html>
```

We create a base file with the basic HTML structure. We then Inside the `body` tag we have the `block` control block that defines elements that can be changed by the derived template.

We can create as many blocks as we want to replace in our content. The `block content` is what we will be replacing in our other templates.

index.html

```
{% extends 'base.html'%}
{% import 'macros.html' as macro%}
```

```
...
```

We first extend our *base.html* file to the top of our index page.

index.html

```
<!-- Content block -->
{% block content %}

<!-- Popular movie -->
<h3> Popular Movies</h3>
<ul>
    {{ macro.displayMovieList(popular)}}
</ul>
<!-- Upcoming movie  -->
<h3> Upcoming Movies </h3>
<ul>
    {{ macro.displayMovieList(upcoming)}}
</ul>
<!-- Now showing movie -->
<h3>Now Showing</h3>
<ul>
    {{ macro.displayMovieList(now_showing)}}
</ul>


{% endblock%}
...
```

We then replace our HTML structure with our `block content` and place our body content inside the blocks.

Task

Replace the HTML in the *movie.html* file and extend the *base.html*

Adding Bootstrap

Our application looks a bit stale we can add some styling using twitter bootstrap framework. We use the Flask-Bootstrap extension.

Let us download the extension.

```
(virtual)$ pip install flask-bootstrap
```

We then need to initialize the extension. Most flask extensions need to be initialized before we can start using them. We do that in our *__init__.py* file .

__init__.py

```
...
from flask_bootstrap import Bootstrap
...
```

We first import the `Bootstrap` class from the `flask_bootstrap` extension.

__init__.py

```
...
# Initializing application
app = Flask(__name__,instance_relative_config = True)

# Setting up configuration
app.config.from_object(DevConfig)
app.config.from_pyfile("config.py")

# Initializing Flask Extensions
bootstrap = Bootstrap(app)
...
```

We then initialize the Bootstrap class by passing in the app instance. This is how most extensions are initialized. We can now use bootstrap in our application.

We can now replace **all** the content in our *base.html* file with the following code.

base.html

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

Flask-Bootstrap contains its own *base.html* which has defined blocks that we can refer to and add our own data. **Full List of Flask-Bootsrap available blocks (https://pythonhosted.org/Flask-Bootstrap/basic-usage.html#available-blocks)** We can run our application and see it still runs as normal.

Let us add a navigation bar to our application. Create a new template file *navbar.html*.

navbar.html

```
    <div class="navbar navbar-inverse" role="navigation">
        <div class="container-fluid">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-targe
t=".navbar-collapse">
                  <span class="sr-only">Toggle navigation</span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="/"> Watchlist </a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="/">Home</a></li>
            </div>

        </div>
    </div>
```

We create a bootstrap navbar and we can now include it in our *base.html* file.

base.html

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

<!-- Navbar block  -->
{% block navbar %}
    {% include 'navbar.html' %}
{% endblock %}
```

When we have sections of our template that are reused in multiple parts or in multiple templates we can separate it into its own template file and include it wherever we need it.

We use the `bootstrap/base.html` navbar block and we use the **includes** control block to include our _navbar.html_ file to that section of the *base.html* template.

We can now run our application and see our new navbar.

You can find the application at this point from here https://github.com/mbuthiya/watchlist/tree/09-Adding-Bootstrap