TEMPLATES & CUSTOM HTML

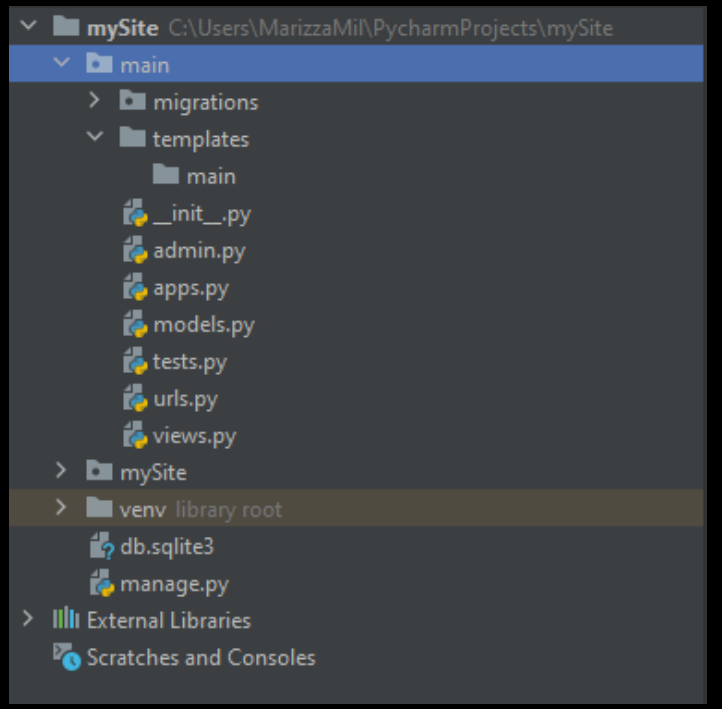
Up until this point in the series we have been returning an HttpResponse to our views which only allowed for small, simple HTML. In this tutorial I will be showing how to create complex, dynamic and scale-able HTML code by using Templates!

SETTING UP TEMPLATES

A template is essentially just an HTML file that we will render and display on our webpage. To start creating these templates we need to do the following:

- Create a new folder called "templates" inside of your app directory (my app is called "main")
- Create a folder inside of the templates folder that is the name of our app (in my case I would call it "main")

Now you can place all of your HTML templates inside that folder.



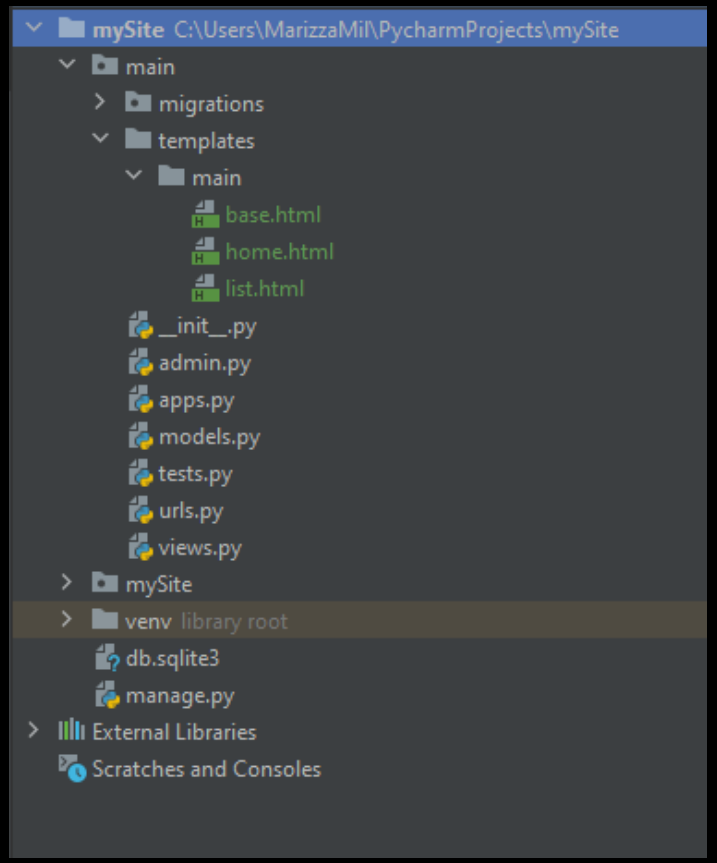
TEMPLATE INHERITANCE

One of the reasons django is so scale-able is because of template inheritance. It works on precisely the same principle as OOP inheritance where you can use all of the code from one template inside another. Obviously you can also override aspects of that code and add your own from within the child template.

For this tutorial we will create three templates:

- base.html
- home.html
- list.html

Our templates "home.html" and "list.html" will inherit from "base.html"



CREATING THE BASE TEMPLATE

The base template is where we will put all of the code that we wan't to apply to both of our other HTML files. It should contain things like a website sidebar or logo, essentially anything that will always be shown no matter what the page. For our purposes we will create a very basic base template and add to it later.

 # base.html file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}My Site{% endblock %}</title>
</head>
<body>
    <div id="content" name="content">
    {% block content %}
    {% endblock %}
    </div>
</body>
</html>
```

The {% %} symbol is the way that we can add code into our html files. When we are going to be creating a new block or writing something like an if statement or for loop we put it inside of this tag.

When we put {% block content %} inside our html we are defining that this block can be overridden from a child class and that what's inside of it should be replaced by whatever appears inside the corresponding tags of the child class. As you can see we have defined two blocks here: content and title.

It is important than whenever we create a block we also end it by using {% endblock %}.

CHILD TEMPLATES

The next template that we will code will be the home.html template. This template will inherit all of the code from our base template and replace the blocks set in the base template.

To inherit from the base template we need to include the following line at the beginning of our code:

{% extends 'main/base.html' %}

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

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

{% block content %}
<h1>Home Page</h1>
{% endblock %}
```

 # home.html file

By redefining the blocks inside of our child template whatever is inside those tags will replace the block with a corresponding name from our base template.

PYTHON CODE INSIDE TEMPLATES

Now it is time to move onto our template list.html. This template will be responsible for displaying the contents of any To Do List we pass it. This means we now need to setup our template in a way where it can accept a variable and change the HTML code accordingly.

The next section will show how to pass variables to our templates but let's start by showing how we can use them.

To use a variable inside of a template you simply embed it in: {{ }}
Anything inside those tags will be treated as a variable. In the example below we use a variable called "ls" that will need to passed into the template from our views.py file.

We can also use code like if statements and for loops inside or templates like seen below.

 # list.html file

```
{% extends "main/base.html" %}
{% block title %}View List{% endblock %}

{% block content %}
    <h1>{{ls.name}}</h1>
    <ul>
        {% for item in ls.item_set.all %}
            {% if item.complete == True %}
                <li>{{item.text}} - COMPLETE</li>
            {% else %}
                <li>{{item.text}} - INCOMPLETE</li>
            {% endif %}
        {% endfor %}
    </ul>
{% endblock %}
```

This code will display the name of the To Do List we pass it as well as all of the items in that list below it.

RENDERING OUR VIEWS

Now that we have created our templates it is time to show them on the screen. To do this we need to render them from our views.py file.

```python
# views.py file
from django.shortcuts import render
from django.http import HttpResponse
from .models import ToDoList, Item

def index(response, id):
    ls = ToDoList.objects.get(id=id)
    return render(response, "main/list.html", {"ls":ls})

def home(response):
    return render(response, "main/home.html", {})
```

Modify code

Add code

The {} inside the render function is where we are able to pass a dictionary of variables to our templates. The key for the dictionary should be the name of the variable from the template and the value should be the value to pass.

EDITING THE URLS

Almost done! Now the last thing to do is edit our URLS and we are finished!

```python
# urls.py file

from django.urls import path
from . import views

urlpatterns = [
    path('<int:id>', views.index, name='index'),
    path("", views.home, name="home")
]
```

Modify code

Add code