Master Template

The idea of having separate templates for each django app is good but far from completely functional. That's because there are a ton of alike parts in a webpage that can be reused instead of re-implemented!

You must already be familiar with the concept of Abstraction and Inheritance in OOP, good news is that we have something faily similar to those when it comes to django templates.

You can have a set of HTML docs as the Master Template of your project. To do this, we only need to make a folder called `templates` at the project root. The hierarchy would look like this after the work is done.

```
/MyProject
    /MyProject
        <files.py>

    /app1
        /templates
            /app1
                index.html
        <files.py>

    /app2
        /templates
            /app2
                index.html
        <files.py>

    /templates                       # Master Template!
        master.html
```

You'll notice the file called `master.html` in the `templates` directory that includes anything your website has among different pages (most of the pages). Like the intersection for most of html files in the project hierarchy. Not only that, you must also add a Django Template Tag called `block` in any place that might be overrided in project apps. Take a look:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>
        {% block page_title %} <!--page_title is like a variable name
for this block-->
            Default Page Title
        {% endblock page_title %}
    </title>
</head>
    <body>
        {% block content %}

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

Make sure you have `BASE_DIR / "templates` added to `TEMPLATES` in Project Settings.

```
from MyProject.MyProject.settings import *

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            BASE_DIR / "templates",
        ],
        'APP_DIRS': True,

        # The rest of the dictionary.
    },
]
```

When master template is set up, it's time to move to the `/MyProject/app1/templates/app1/index.html` or any other HTML file you desire.`

```
{% extends "master.html" %}
<!--
    The file will automatically find master.html
    if the master template directory is added to project
-->

{% block page_title %} App1 {% endblock page_title %}

{% block content %}
    <h1>Extends Master Template!</h1>
{% endblock content %}
```

Includes!

You can store a slice of a HTML file to later reuse it in other parts of your project. By making a new folder in `/MyProject/app1/templates/app1/includes` in which you'll be having html files like this:

```
<!--/MyProject/app1/templates/app1/includes/header.html-->
<header>
    <nav>
        <a href="/">click me!</a>
    </nav>
</header>
```

You are now able to stick this slice to any part of the app1 pages using a Django Template Tag called `{% includ %}`

```
<!--/MyProject/app1/templates/app1/index.html-->
{% extends "master.html" %}
<!--
    The file will automatically find master.html
    if the master template directory is added to project
-->

{% block page_title %} App1 {% endblock page_title %}

<!-- ----- -->

{% block content %}
    <h1>Extends Master Template!</h1>
    {% include "app1/includes/header.html" %}
{% endblock content %}
```

The include block supports sending data back to the reference HTML files, something like the `context` parameter that was sent to rendered HTML when using `render()` method.

```
    {% include "app1/includes/header.html" with context="Sent Data"%}
```

And back in the refrence HTML file `header.html`:

```
<header>
    <nav>
        <h3>{{ context }}</h3>
        <a href="/">click me!</a>
    </nav>
</header>
```

Note: The syntax of Django Template Language is not highlited here, so it maked it pretty difficult to understand which words are keywords and whichones are values. So below you will file a list of so-far-used keywords in DTL. Nth beyond this list is a Keyword in DTL examples used so far.

- for / in
- if / elif / else
- block

- include / with
- endfor / endif / endblock ...

404 Not Found!

The not found page is one of the most common parts where you need to work with master templates. Look at this part as a practice for other things we learned so far. Let's dive right in.

First off, we make the HTML file for our not found page somewhat like below, where do we make this file? of course in the master templates directory.

```
<!-- /MyProject/templates/404.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body style="text-align: center; font-size: 100px;">
    <h1 style="line-height: 50px;">404</h1>
    <h3 style="font-size: 60px;">Not Found!</h3>

    <h5>
        {% block url %}
            /
        {% endblock url %}
    </h5>
</body>
</html>
```

Then, it's time to make a view that renders this page.

```
from django.template.loader import render_to_string
from django.http import HttpResponseNotFound

def not_found(request, dynamic):
    response = render_to_string('404.html') # Another way was to
extend the HTML file and user render().
    return HttpResponseNotFound(response)
```

Finally, add a Dynamic Address URL to the `urlpatterns`.

```
from django.urls import path
from MyProject.app1 import views

urlpatterns = [
    path("app1", views.index, name="app1"),
    path("<dynamic>", views.not_found, name="not_found")
]
```

One cool thing here is that `django.http` has a built in exception called `Http404()` which could be raised in the `not_found()` view we made. This exception looks at mentioned URLs in project settings for a file named exactly `404.html`. and returns that file.

You won't see the result of `Http404()` exception in your project when you're in debug mode!

Literally everything about this exception shouts it's a best practice!

Final Joke!

If you deploy your website in `Debug = True`, start packing your stuff right away and say goodbye tothe company!