Theory: Static content

11 minutes 0 / 4 problems solved

Making your HTML pages visually appealing is really important. For that, you need to add some style to your pages using CSS or even do some programming with JavaScript. Both CSS and JavaScript run smoothly in your browser: all you need is to send these files to the user. Let's see how you can do it in a few steps with Django.

> You rarely see sites serving static content (images, JavaScript, CSS) with Django because specialized tools work with it more effectively. However, for development purposes or even for your pet project with only a few visitors, you can serve content directly with Django.

§1. Static files

HTML pages consist of three main parts:

- HTML layout
- CSS
- JavaScript code

It's convenient to define styles for pages in separate files to make our templates clear and easy to read. The same applies to JavaScript code. We know how to render templates, but how do we send other types of files to a client?

Let's refer to an example. We keep working on John Doe's blog, and this time, we want to add some style to it.

Let's define a simple CSS file with just the properties for the `h2` elements:

```
h2 {
  font-size: 24px;
  color: gray;
}
```

We save it to the file *static/css/base.css* relative to the root of the project. Also, you need to define a template in the *blog/templates/blog/index.html* file:

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/base.css' %}">

<h2>{{ blog_name }}</h2>
<div>{{ post.text }}</div>
```

We use the tag `{% load static %}` to tell Django that we want to import an additional tag for the templates named **static**. After that, we use the tag `{% static 'css/base.css' %}` as a URL to the stylesheet for the page.

> Note that we only use the *css/base.css* part for the needed stylesheet; we omit the part of the URL that matches the `STATIC_URL` variable in the *settings.py* module.

However, if you try to launch the application using this template, nothing happens to the style of the `h2` element. We forgot to tell Django that we want to use it for serving static files! Add this to the end of your *settings.py* module:

```
DEBUG = True

STATIC_URL = "/static/"
STATICFILES_DIRS = [os.path.join(BASE_DIR, "static")]
```

Current topic:

Static content Stage 5

Topic depends on:

✕ Basic syntax Stage 5

✕ Template tags Stage 3

Table of contents:

Extend your `urlpatterns` in the *urls.py* module:

```
from django.conf import settings
from django.conf.urls.static import static

# your urlpatterns

urlpatterns += static(settings.STATIC_URL)
```

Now, you can launch the application again, and serving static files will work like a charm! Your `h2` element will have gray color, and if you want to change it, you may refer to the table with other color names supported by major browsers.

> The restriction for serving static files with Django is that it only works in DEBUG mode. If your server has some sensitive data or you worry that someone will see information about your application and code in the debug trace, consider using other methods to serve static files.

§2. Media files

Besides HTML, a page may contain media files like images, videos, and audio files. To keep these files, we use the media folder on the server. It usually includes all users' media content that we keep on the server. Adding this path to your project is no harder than using static files.

First, we modify our *settings.py* module:

```
DEBUG = True

TEMPLATES[0]['OPTIONS']['context_processors'].append(
    'django.template.context_processors.media'
)

MEDIA_URL = "/media/"
MEDIA_ROOT = os.path.join(BASE_DIR, "media")
```

Then, extend `urlpatterns`:

```
from django.conf import settings
from django.conf.urls.static import static

# your urlpatterns


urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Now, create the *media* folder in the root of your project and place a picture in it with the name *media/avatar.jpg*. Finally, save the new content in your template:

```
<h2>{{ blog_name }}</h2>

<div>Here am I</div>
<img src="{{ MEDIA_URL }}/avatar.jpg" alt="avatar">

<div>{{ post.text }}</div>
```

That's all! Now the blog has a nice profile picture of John Doe on the page.

§3. Conclusion

Django provides you tools to fully cover the development process of a site, including serving static and media files. You can only use it in the debug mode, so it's only a starting point for you. To serve static files, configure your settings and URL patterns, and then add links to the content in your templates.

Report a typo

30 users liked this theory. 3 didn't like it. What about you?

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion