

Django

Documentation

Writing your first Django app, part 6

This tutorial begins where Tutorial 5 left off. We've built a tested Web-poll application, and we'll now add a stylesheet and an image.

Aside from the HTML generated by the server, web applications generally need to serve additional files — such as images, JavaScript, or CSS — necessary to render the complete web page. In Django, we refer to these files as “static files”.

For small projects, this isn't a big deal, because you can just keep the static files somewhere your web server can find it. However, in bigger projects — especially those comprised of multiple apps — dealing with the multiple sets of static files provided by each application starts to get tricky.

That's what **django.contrib.staticfiles** is for: it collects static files from each of your applications (and any other places you specify) into a single location that can easily be served in production.

Customize your app's look and feel

First, create a directory called **static** in your **polls** directory. Django will look for static files there, similarly to how Django finds templates inside **polls/templates/**.

Django's **STATICFILES_FINDERS** setting contains a list of finders that know how to discover static files from various sources. One of the defaults is **AppDirectoriesFinder** which looks for a “static” subdirectory in each of the **INSTALLED_APPS**, like the one in **polls** we just created. The admin site uses the same directory structure for its static files.

Within the **static** directory you have just created, create another directory called **polls** and within that create a file called **style.css**. In other words, your stylesheet should be at **polls/static/polls/style.css**. Because of how the **AppDirectoriesFinder** staticfile finder works, you can refer to this static file in Django simply as **polls/style.css**, similar to how you reference the path for templates.



Static file namespacing

Just like templates, we *might* be able to get away with putting our static files directly in **polls/static** (rather than creating another **polls** subdirectory), but it would actually be a bad idea. Django will choose the first static file it finds whose name matches, and if you had a static file with the same name in a *different* application, Django would be unable to distinguish between them. We need to be able to point Django at the right one, and the easiest way to ensure this is by *namespacing* them. That is, by putting those static files inside *another* directory named for the application itself.

Put the following code in that stylesheet (**polls/static/polls/style.css**):

polls/static/polls/style.css

```
li a {
    color: green;
}
```

Next, add the following at the top of **polls/templates/polls/index.html**:

polls/templates/polls/index.html

```
{% load staticfiles %}

<link rel="stylesheet" type="text/css" href="{% static 'polls/style.css' %}" />
```

{% load staticfiles %} loads the **{% static %}** template tag from the **staticfiles** template library. The **{% static %}** template tag generates the absolute URL of the static file.

That's all you need to do for development. Reload **http://localhost:8000/polls/** and you should see that the question links are green (Django style!) which means that your stylesheet was properly loaded.

Adding a background-image

1 of 3

Next, we'll create a subdirectory for images. Create an **images** subdirectory in the **polls/static/polls/** directory. Inside this directory, put an image called **background.gif**. In other words, put your image in **polls/static/polls/images/background.gif**.

Language: en

05/06/2016 10:03 PM

`polls/static/polls/style.css`

```
body {
    background: white url("images/background.gif") no-repeat right bottom;
}
```

Reload `http://localhost:8000/polls/` and you should see the background loaded in the bottom right of the screen.



Warning

Of course the `{% static %}` template tag is not available for use in static files like your stylesheet which aren't generated by Django. You should always use **relative paths** to link your static files between each other, because then you can change `STATIC_URL` (used by the `static` template tag to generate its URLs) without having to modify a bunch of paths in your static files as well.

These are the **basics**. For more details on settings and other bits included with the framework see [the static files howto](#) and [the staticfiles reference](#). Deploying static files discusses how to use static files on a real server.

When you're comfortable with the static files, read [part 7](#) of this tutorial to learn how to customize Django's automatically-generated admin site.

[◀ Writing your first Django app, part 5](#)

[Writing your first Django app, part 7 ▶](#)

Learn More

[About Django](#)

[Getting Started with Django](#)

[Team Organization](#)

[Django Software Foundation](#)

[Code of Conduct](#)

[Diversity statement](#)

Get Involved

[Join a Group](#)

[Contribute to Django](#)

[Submit a Bug](#)

[Report a Security Issue](#)

Language: **en**

