Introducing DJ-Static

2013

1 min read • 211 words

Django doesn't recommend the production use of its static file server for a number of reasons. There exists, however, a lovely WSGI application aptly named Static.

Thus, DJ-Static was born.

> Finally, a super-simple way of serving assets in Django that'll actually perform well — @jacobian

Installation and Configuration

```
$ pip install dj-static
```

Configure your static assets in `settings.py`:

```
STATIC_ROOT = 'staticfiles'
STATIC_URL = '/static/'
```

Then, update your `wsgi.py` file to use DJ-Static:

```
from django.core.wsgi import get_wsgi_application
from dj_static import Cling

application = Cling(get_wsgi_application())
```

That's it! Django deployment has never been simpler.

Benefits

Serving static files from Python greatly simplifies the deployment process. The fewer moving parts your application has, the fewer parts there are to break unexpectedly.

Most importantly, this facilitates Dev/prod parity, which should be a goal of all developers.

What about a CDN?

If you have to ask that question, there's actually quite a good chance you don't. Static responses aren't very different than dynamic ones, especially when using the HTTP Cache headers that DJ-Static provides.

If you're running a top-tier application, optimizing for delivery and reducing frontend load, you will want to explore using a CDN with Django-Storages.

Related Links

- DJ-Static on PyPi
- DJ-Static on GitHub
- Django and Static Assets on Heroku
- The 12 Factor App: Dev/prod Parity