django-static-delivery

unknown

Feb 25, 2023

CONTENTS

CHAPTER

ONE

ALL CONTENTS

1.1 Installation

django-static-delivery supports Python 3 only and requires at least Django 1.11. No other dependencies are required.

To start, simply install the latest stable package using the command

```
$ pip install django-static-delivery
```

In addition, you have to add `'static_delivery.StaticDeliveryMiddleware'` to the `MIDDLEWARE` setting in your `settings.py`. Make sure to add the middleware to the top of the list.

```
MIDDLEWARE = [
    'static_delivery.StaticDeliveryMiddleware',
    # ... all other middlewares
]
```

Please make sure that your `staticfiles` related settings are configured properly. Besides having `STATIC_ROOT` and `STATIC_URL` set, you have to use a staticfile storage with hashed file names, for example `ManifestStaticFilesStorage`.

```
# Filesystem path there collected staticfiles are stored
STATIC_ROOT = '/var/www/static'

# Public base path to access files in STATIC_ROOT
STATIC_URL = '/static/'

STATICFILES_STORAGE = 'django.contrib.staticfiles.storage.ManifestStaticFilesStorage'
```

Thats it, now continue to the *Advanced topics section* to learn how to optimize your reverse proxy for a good performance - serving static files via Django is never a fast way.

1.2 Advanced topics

1.2.1 Nginx configuration

To improve the performance of static file delivery, you might use the cache options Nginx provide.

Here is an example configuration of Nginx together with uwsgi:

```
# Prepare a cache with 100 MB storage capacity
uwsgi_cache_path
    /path/to/cache
    levels=1:2
    keys_zone=static_cache:2m
    max_size=100m
    inactive=1w
    use_temp_path=off;

# You django backend / uwsgi process
upstream app {
    server django:8000;
}

server {
    listen 80 default_server;
    root /var/www;

    location / {
        try_files $uri @proxy_to_app;
    }

    # Important section, tell nginx to use the cache for all requests to /static/*
    location /static/ {
        uwsgi_cache static_cache;
        uwsgi_cache_use_stale updating;
        uwsgi_cache_lock on;
        uwsgi_cache_valid any 1w;
        uwsgi_cache_key $host$request_uri;

        include uwsgi_params;
        uwsgi_pass app;
    }

    location @proxy_to_app {
        include uwsgi_params;
        uwsgi_pass app;
    }
}
```

Please note, this is just an example. Please test the configration before putting this to production.

Note: You are not bound to use uWSGI. Gunicorn will work fine too - you just need to find the Nginx equivalents to the uwsgi_* settings (most of them will be prefixed proxy_*)

1.3 API Reference

1.3.1 Middleware

class static_delivery.middleware.**StaticDeliveryMiddleware**(*get_response=None*)

Bases: object

Middleware to serve static files from within Django.

In some setups it is a good idea to serve static files from Django and have them cached in a reverse proxy like Nginx or something similar.

By doing this, we can easily serve static files from our - for example - our Docker image without putting them in a shared volume.

It is important to know that serving files from Django directly won't perform very well. Always have a cache in front of it.

Additionally, the middleware is able to recover from invalid hashes in static file names when you use a staticfiles storage with name hashing in place. If a file with a certain hash is unavailable, the middleware will try to look up the correct hash for the file.

HASHED_PATH_RE = re.compile('(.+)(\\.[0-9a-f]{12})(\\.?)(\\w+)?$')

path_re = None

The middleware instance has a regex ready to match paths against STATIC_URL.

manifest = None

the staicfiles manifest is loaded once when the middleware is initialized.

serve_response(*request*, *path*)

This method takes the request and a path to deliver static content for.

The method tries to deliver content for the requested path, if this fails the code will try to recover the currently valid path and try to serve that file instead. If nothing works, None is returned.

get_staticfile_response(*request*, *path*)

This method takes a path and tries to serve the content for the given path. Will return None if serving fails.

recover_staticfile_path(*path*)

This method strips the hash from the requested path and tries to look up the unhashed file name in the manifest dataset.

load_staticfiles_manifest()

Supported staticfiles storages map original static file names to hashed ones. This method loads the manifest file for lookups later.

In addition, this method might raise an exception if the configured staticfiles storage doesn't support manifest files/data.

1.4 Changelog

1.4.1 0.0.2 - 2021-02-11

- For not found static files, don't raise 404 but pass to next middleware.

1.4.2 0.0.1 - 2018-04-26

- Initial release of *django-static-delivery*

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

G

H

L

M

P

R

S