django-webp-converter Documentation

Release 0.2.1

Tom Miller

Apr 21, 2020

Contents

CHAPTER 1

Introduction

django-webp-converter is a Django app which straightforwardly converts static images to WebP images, falling back to the original static image for unsupported browsers.

WebP lossless images are 26% smaller in size compared to PNGs. WebP lossy images are 25-34% smaller in size compared to JPEG images at equivalent SSIM index (https://developers.google.com/speed/webp/?hl=en). By using WebP images, you can increase page load speeds with no noticeable change in image quality.

CHAPTER 2

Contents

2.1 Installation

2.1.1 Prerequisites

Note: django-webp-converter requires that Pillow and its appropriate libraries are installed. Please see https://pillow.readthedocs.org/ for instructions on installing Pillow and its dependencies.

2.1.2 Installation

Install django-webp-converter with `pip`:

```
$ pip install django-webp-converter
```

Add `webp_converter` to your INSTALLED_APPS in settings.py:

```
INSTALLED_APPS = (
    ...,
    'webp_converter',
)
```

Add the `webp_support` context processor to your list of context processors:

```
'context_processors': [
    ...,
    'webp_converter.context_processors.webp_support',
]
```

Run `./manage.py migrate` to add the required tables to the database.

You will also need to configure django to serve locally stored files by configuring the `MEDIA_URL` and `MEDIA_ROOT` settings in your project's settings.py file.

For example:

```
MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

You'll probably also want to ensure that these files will be served during development.

This project also makes use of Django's cache framework to avoid making repeated requests to the database. You should ensure that you have set up caching via the `CACHES` setting to achieve optimal performance.

2.2 Usage

2.2.1 Showing WebP images

Load the webp_converter template tag in your template:

```
{% load webp_converter %}
```

Replace usage of the `static` tag with the `static_webp` tag. For example:

```
<img src="{% static_webp 'img/hello.jpg' %}">
```

The user will be shown a WebP version of the image if their browser supports it (currently Chrome & Opera). Otherwise, they will be shown the original static file.

You can also specify the quality of the WebP image. For example:

```
<img src="{% static_webp 'img/hello.jpg' 70 %}">
```

2.2.2 Clearing the cache

Clear the cache and delete the WebP images folder by running the `manage.py` command:

```
./manage.py clear_webp_cache
```

By default this will display a confirmation prompt. If you would like to clear the cache without receiving any prompts, you can run:

```
./manage.py clear_webp_cache --no-input
```

2.3 Settings

2.3.1 WEBP_DIRECTORY

Default: `WEBP`

The name of the directory where WebP images are stored.

2.3.2 WEBP_CACHE_PREFIX

Default: `webp_converter`

The key prefix prepended to all cache keys.

2.4 Changelog

2.4.1 0.3.0 (2020-04-21)

- Use the default_storage.save and default_storage.delete methods when saving and deleting images.
- Add check for whether a WebP image already exists before attempting to save the WebP image again following a cache miss.

2.4.2 0.2.1 (2019-12-07)

- Remove duplicate deletion of WebP folder in the *clear_webp_cache* command.

2.4.3 0.2.0 (2019-12-04)

- Switch to MIT license
- Fix bug where duplicate WebPImage models could be created due to the non-uniqueness of nullable fields (https://github.com/tmiller02/django-webp-converter/issues/1).
- Change the default behaviour of the 'clear_webp_cache' command to prompt for user confirmation before deleting the WebP image folder.
- Restructure project to split the webp_converter app out of the sample project
- Regenerate sample project using Django 3.0.