coltrane

Release 0.31.0

Adam Hill

Jan 08, 2024

CONTENTS

`coltrane` is a ***Dynamic Site Generator*** that harnesses the power of `Django` without the hassle. It can also be used to build a static HTML site or as a third-party `Django` app.

CHAPTER

ONE

FEATURES

- Render `markdown` files as HTML with automatic URL routing based on the filesystem
- Local development server with live re-rendering of markdown and data
- Use JSON files as data sources in content
- Automatic generation of `sitemap.xml` and `rss.xml` files
- Can serve non-markdown files like `robots.txt`
- Deployment best practices with `whitenoise` and `gunicorn` already configured
- Leverage custom or built-in `Django` template tags and filters
- Include any third-party `Django` app for additional functionality
- Optional building of static HTML files

CHAPTER

TWO

WHAT IS A DYNAMIC SITE GENERATOR?

`coltrane` is similar to a static site generator – it takes `markdown` content and renders it as HTML. However, it also provides an opinionated framework for building dynamic websites.

2.1 Examples in the wild

- GitEgo: An egocentric view of GitHub
- python-utils: Interactive Python playground
- unsuckjs.com: Libraries to progressively enhance HTML with minimal amounts of JavaScript

Note: Please let me know if you use `coltrane` and would like to add it to this list!

CHAPTER

THREE

WHAT'S WITH THE NAME?

`coltrane` is built on top of the `Django` web framework, which is named after Django Reinhardt. This framework is named after John Coltrane, another (more avant-garde) jazz musician.

CHAPTER

FOUR

INSPIRATION

https://twitter.com/willmcgugan/status/1477283879841157123 for the initial inspiration and my reaction https://twitter.com/adamghill/status/1477414858396164096.

CHAPTER

FIVE

DEPENDENCIES

- https://github.com/adamchainz/django-browser-reload for development server live reloads
- https://github.com/boxed/django-fastdev to ensure template variables are available
- https://github.com/lepture/mistune for doing the hard work of rendering the markdown
- https://www.djangoproject.com for doing the hard work of everything else

CHAPTER

SIX

OTHER MINIMAL DJANGO PROJECTS

- https://github.com/wsvincent/django-microframework for the `app.py` idea
- https://olifante.blogs.com/covil/2010/04/minimal-django.html
- https://simonwillison.net/2009/May/19/djng/
- https://stackoverflow.com/questions/1297873/how-do-i-write-a-single-file-django-application
- https://github.com/pauloxnet/uDjango

CHAPTER

SEVEN

RELATED PROJECTS

yamdl is another approach which lets you store instances of Django models as flat YAML files. It also supports storing markdown.

Here are a few Python static site generators:

- Pelican: Pelican is a static site generator that requires no database or server-side logic.
- Combine: Build a straightforward marketing or documentation website with the power of Jinja.
- Nikola: In goes content, out comes a website, ready to deploy.
- Lektor: A flexible and powerful static content management system for building complex and beautiful websites out of flat files.
- corvid: An opinionated simple static site generator.
- jamstack: The easiest way to create jamstack sites, as simple or as complex as you like.

7.1 Installation

1. `mkdir new-site && cd new-site` to create a new folder
2. `poetry init --no-interaction --dependency 'coltrane:<1' && poetry install` to create a new virtual environment and install the `coltrane` package
3. Optional: `brew install watchman` on MacOS for less resource-intensive local development server

7.1.1 Extras

`coltrane` has some additional functionality that is not enabled by default, but can be used if it is installed with extras.

`json5`

Adds support for using JSON5 for *data* files. This allows trailing commas and comments in JSON, so it can be useful for making JSON a little more readable.

```
poetry add coltrane -E json5
```

`deploy`

Adds support for deploying `coltrane` to a production server with `gunicorn` and `whitenoise` pre-configured. More details at *deployment.md*.

```
poetry add coltrane -E deploy
```

7.2 Local Development

`markdown` files are rendered into HTML dynamically based on the URL that is requested.

7.2.1 Create a new site

`poetry run coltrane create` creates the folder structure for a new site.

Note: More details about the `create` options and the files that are generated are in *CLI*.

7.2.2 Development server

`poetry run coltrane play` serves the content for local development.

Warning: `poetry run coltrane play` is fine for development, but should never be used in production.

7.3 Content

`coltrane` is designed around content. There are no URL routes to configure or `views` (if you are used to Django) or `controllers` (if you are used to other MVC frameworks) to create. `coltrane` automatically routes URLs to the correct content based on where the files exist on the filesystem. `markdown` files will get converted to HTML for rendering. HTML template files can also be served directly for more control.

7.3.1 Markdown

Add `markdown` files (or sub-directories with `markdown` files) to the `content` directory and rendered HTML will be accessible via auto-generated routes. `index.md` would be used similarly to `index.html`.

- `/` would convert the `markdown` in `content/index.md` and render is as HTML
- `/about/` would convert the `markdown` in `content/about.md` and render it as HTML
- `/articles/` would convert the `markdown` in `/content/articles/index.md` and render it as HTML
- `/articles/this-is-the-first-article/` would convert the `markdown` in `/content/articles/this-is-the-first-article.md` and render it as HTML
- `/not-there/` would 404

Frontmatter

YAML before the actual markdown content is supported. Any keys and their values will be added to the context variable that is used when rendering the HTML. The default base.html template will use lang (to specify the HTML language; defaults to "en"), and title variables if they are specified in the frontmatter.

template

Used to specify a custom template that Django will use to render the markdown.

content/index.md

```
---
lang: en
title: This is a good title
template: another_app/new-template.html
adjective: perfect
---

This is sample text
```

another_app/new-template.html

```
<title>{{ title }}</title>

{{ content }} and it's {{ adjective }}
```

Generated HTML

```
<title>This is a good title</title>

<p>This is sample text and it's perfect</p>
```

7.3.2 HTML

If a markdown file can not be found for the based on the URL's slug, but there is an HTML file with the same slug in the templates directory the HTML template will be rendered.

- /app/ would render the HTML in /templates/app.html or /templates/app/index.html

Wildcards

A filename with an asterisk can be used as a "wildcard" and will be served for any slug that does not have a matching markdown or specific HTML template file.

- /app/some-user would render the HTML from /templates/app/*.html

Directories can also be a wildcard to handle a specific part of a slug.

- /app/some-user would render the HTML from (in priority order) /templates/app/some-user.html or /templates/app/*.html or /templates/*/some-user.html or /templates/*/*.html
- /app/another-user would render the HTML from (in priority order) /templates/app/another-user.html or /templates/app/*.html or /templates/*/another-user.html or /templates/*/*.html

7.4 Templates

`coltrane` comes with two minimal templates that get used by default: `coltrane/base.html` and `coltrane/content.html`. Overriding those templates works just like in Django.

7.4.1 Override included templates

coltrane/base.html

Create a file named `templates/coltrane/base.html` in your app to override the base template. By default, it needs to include a `content` block.

```
{% block content %}{% endblock content %}
```

coltrane/content.html

Create a file named `templates/coltrane/content.html` in your app to override the content template. By default, it needs to include a `content` block for the base template and `{{ content }}` to render the markdown.

Note: The `content` template variable is already marked "safe" so you do not need to use the `safe` filter.

```
{% block content %}{{ content }}{% endblock content %}
```

7.4.2 Custom template

Specify a custom template with a `template` variable in the `markdown` frontmatter.

content/index.md

```
---
title: This is good content
template: sample_app/new-template.html
---

# Heading 1

This will use sample_app/new-template.html to render content.
```

sample_app/new-template.html

```
<title>{{ title }}</title>

{{ content }}
```

Generated HTML

```
<title>This is good content</title>

<h1 id="heading-1">Heading 1</h1>
```

(continues on next page)

(continued from previous page)

```
<p>This will use sample_app/new-template.html to render content.</p>
```

7.5 Template tags

Template tags are the way for Django templates to use Python code. Django has a large list of built-in template tags for everything from looping over objects, date formatting, boolean logic with `if/else` blocks, or getting the length of an object. By default, all template tags in Django are available in markdown content files.

7.5.1 Humanize template tags

`django.contrib.humanize` includes a useful template tags to format numbers and dates in human-friendly ways. Normally it needs to be enabled and loaded in templates manually, but `coltrane` enables it by default so it is available to use in markdown content files automatically.

7.5.2 Coltrane template tags

`directory_contents`

A list of the content at a particular directory.

List markdown files based on the request path

If the request url is https://localhost:8000/ and there are these files:

- content/test1.md
- content/test2.md

```
# Contents

{% directory_contents as directory_contents %}

{% for content in directory_contents %}

- {{ content.slug }}

{% endfor %}
```

```
<h1 id="contents">Contents</h1>

<ul>
  <li>test1</li>
  <li>test2</li>
</ul>
```

List markdown files based on a particular directory

If the request url is https://localhost:8000/ and there are these files:

- content/articles/article1.md

- content/articles/article2.md

```
# Articles

{% directory_contents 'articles' as directory_contents %}

{% for content in directory_contents %}

- {{ content.slug }}

{% endfor %}
```

```
<h1 id='articles'>Articles</h1>

<ul>
  <li>article1</li>
  <li>article2</li>
</ul>
```

Exclude a slug from being included

If the request url is https://localhost:8000/ and there are these files:

- content/articles/article1.md
- content/articles/article2.md

```
# Articles

{% directory_contents 'articles' exclude='article1' as directory_contents %}

{% for content in directory_contents %}

- {{ content.slug }}

{% endfor %}
```

```
<h1 id="articles">Articles</h1>

<ul>
  <li>article2</li>
</ul>
```

Sort the results of the directory

The `order_by` kwarg will sort the results by a particular `key`. Available `keys` are `slug`, `now`, and anything in the YAML frontmatter. All keys will be coerced to strings and if a `key` is missing an empty string will be used by default.

If the request url is https://localhost:8000/ and these files are present in the `content` directory:

- content/article1.md
- content/article2.md

```
# Sorted Articles

{% directory_contents order_by='slug' as directory_contents %}
```

(continues on next page)

(continued from previous page)

```
{% for content in directory_contents %}

- {{ content.slug }}

{% endfor %}
```

```
<h1 id="sorted-articles">Sorted Articles</h1>

<ul>
  <li>article1</li>
  <li>article2</li>
</ul>
```

```
# Reverse Sorted Articles

{% directory_contents order_by='-slug' as directory_contents %}

{% for content in directory_contents %}

- {{ content.slug }}

{% endfor %}
```

```
<h1 id="reverse-sorted-articles">Reverse Sorted Articles</h1>

<ul>
  <li>article2</li>
  <li>article1</li>
</ul>
```

include_md

Similar to the `include` template tag, but can be used to include a markdown file and have it render correctly into HTML. It can be used in markdown files or in HTML templates.

```
# include_md

{% include_md '_partial.md' %}
```

```
<h1>include_md</h1>

{% include_md '_partial.md' %}
```

parent

A `filter` that returns the parent directory for a particular path. Can be passed a `request` or a `string`.

```
<!-- request of http://localhost/articles/some-article -->
{{ request|parent }} == '/articles'
```

```
{{ 'http://localhost/articles/some-article'|parent|parent }} == ''
```

to_html

Convert raw markdown text to html. This is probably the most useful when using `coltrane` as a `Django app`.

views.py

```
def my_view(request):
    markdown_text = """---
title: Article 1
---

# {{ title }}
"""
    ...
```

my_template.html

```
<main>
    {{ markdown_text|to_html }}
</main>
```

Rendered html content

```
<main>
   <h1>Article 1</h1>
</main>
```

raise_404

Raises a 404 from template. Can be useful when using wildcard HTML templates.

last_path

Gets the last portion the URL path, e.g. the last path of `/app/user/123` would be `"123"`.

`paths`

Gets all parts of the path as a list of strings, e.g. the paths of `/app/user/123` would be `["app", "user", "123"]`.

7.5.3 Custom template tags

`coltrane` will automatically enable any template tags it finds in the `templatetags` directory to be used in `markdown` or HTML templates.

`templatetags/custom_tags.py`

```
from django import template

register = template.Library()

@register.filter(name="test")
def test(value, arg):
    return value + " is a test"
```

`content/index.md`

```
{{ 'This'|test }}
```

Generated `index.html`

```
This is a test
```

7.6 Context

The template context for each `markdown` file includes:

- all key/value pairs in the `markdown` frontmatter
- rendered `markdown` HTML in `content`
- JSON data from the `data` directory
- `now` which provides the current `datetime` (would be the time of HTML rendering for when generating a static site)
- `request` which provides the current request
- `debug` which contains the `DEBUG` setting (or if `INTERNAL_IPS` has the current request's IP)
- `slug` which contains the current file's "slug" (e.g. `articles/some-new-article` if there was a markdown file at `content/articles/some-new-article.md`)
- `toc` which is an automatically generated table of contents rendered as HTML
- if `publish_date` is found, it is converted to a Python `datetime` instance using the excellent dateparser library

7.6.1 Example context

data/index.json

```
{ "test": "Great" }
```

content/index.md

```
---
this_is_a_variable: This is a good test
template: some_app/custom-template.html
publish_date: 2022-02-26 10:26:02
---

{{ this_is_a_variable }}

Data from JSON files: {{ data.index.test }}

Current datetime: {{ now }}

Publish date: {{ publish_date|naturalday }}
```

some_app/templates/some_app/custom-template.html

```
{{ content }}
```

Generated index.html

```
<p>This is a good test</p>

<p>Data from JSON files: Great</p>

<p>Current datetime: 8 Jan. 11, 2022, 10:02 p.m.</p>

<p>Publish date: Feb. 26, 2022</p>
```

7.7 Data

`coltrane` is designed to be used without a database, however, sometimes it's useful to have access to data inside your templates.

7.7.1 JSON data directory

Create a directory named `data` in your project folder (if it doesn't already exist) and add JSON files. The name of the file (without the `json` extension) will be used as the key in the context data.

If there are JSON files in sub-directories, the directory names will be included in the dictionary hierarchy.

data/author.json

```
{
    {"name": "Douglas Adams"}
}
```

data/books/book.json

```
{
    {"title": "The Hitchhiker's Guide to the Galaxy"}
}
```

content/index.md

```
# index

{{ data.author.name }} is the author.

{{ data.books.book.title }} is the book title.
```

Generated index.html

```
<h1 id="index">index</h1>

<p>Douglas Adams is the author.</p>

<p>The Hitchhiker's Guide to the Galaxy is the book title.</p>
```

7.7.2 JSON5 support

JSON5 data files are supported if the *`json5` extra* is installed and the *`COLTRANE_JSON5_DATA` environment setting* is set to `True`.

7.8 Static Files

`Django` handles static files (e.g. CSS, JavaScript, and images) already which `coltrane` leverages as part of the *record command*. The `collectstatic` management command is used to copy all static files to the `output/static` directory.

7.8.1 Referring to static assets

Instead of hardcoding the URL path to static assets, the `static` template tag should be used in either `markdown` or HTML templates.

Note: Using the `static` template tag might feel unnecessary for simpler sites, but it will automatically use hashed file names that `whitenoise` provides for efficient serving and caching of static files.

---

content/index.md

```
![music note]({% static 'images/music-note.svg' %})
```

Generated index.html

```
<img src="/static/images/music-note.abcd123.svg" />
```

templates/custom/custom-template.html

```
<link src="{% static 'css/styles.css' %}" />
```

Generated HTML

```
<link src="/static/css/styles.wxyz789.css" />
```

7.9 Sitemap

`sitemap.xml` is a standard for search engines to find content on your site. `coltrane` automatically provides a URL route for `sitemap.xml` and will create the file when building a static site.

7.9.1 Django app configuration

When using `coltrane` as a `Django` app, the sitemap will *need to be configured*.

7.10 RSS

`coltrane` automatically creates an `rss.xml` file containing all `markdown` content. It will be served from the `/rss.xml` URL or output into the `output` directory for static sites.

7.10.1 Required setting

RSS requires an absolute URL so `coltrane` needs to know the domain for the site.

`COLTRANE_SITE_URL` needs to be set in the `.env` file.

7.10.2 Django app configuration

When using `coltrane` as a `Django` app, RSS will *need to be configured*.

7.11 Deployment

`coltrane` can be installed with deployment features for production by installing the `deploy` extras.

```
poetry add coltrane -E deploy
```

Note: If using `pip` you can do something like: `pip install coltrane[deploy]`.

7.11.1 Required settings

- *DEBUG* should be `False` (more details in Django docs).
- *ALLOWED_HOSTS* must be set to the acceptable host or domain names (more details in Django docs).

```
DEBUG=False
ALLOWED_HOSTS=coltrane.com,www.coltrane.com
```

7.11.2 Gunicorn

`gunicorn` is a production `WSGI` server and is perfect for serving `coltrane` apps.

An example command for using `gunicorn` in production: `poetry run gunicorn -b localhost:8000 app:wsgi`.

7.11.3 Whitenoise

`whitenoise` allows regular `WSGI` servers to serve static files without needing to move assets to S3 or another hosted file platform. It will be configured automatically when `DEBUG` is set to `False`.

7.11.4 Hosting

Docker

A sample `Dockerfile` is created for new `Coltrane` projects. It can be used along with `gunicorn.conf.py` for any hosting platform that supports `Docker`.

Heroku

Heroku will run the `collectstatic` management command by default for Django projects, but this should be disabled by setting the `DISABLE_COLLECTSTATIC` environment variable to `1`. Add the nginx buildpack from https://buildpack-registry.s3.amazonaws.com/buildpacks/heroku-community/nginx.tgz.

Then, add the following files so that `nginx` will serve the static files efficiently.

gunicorn.conf.py

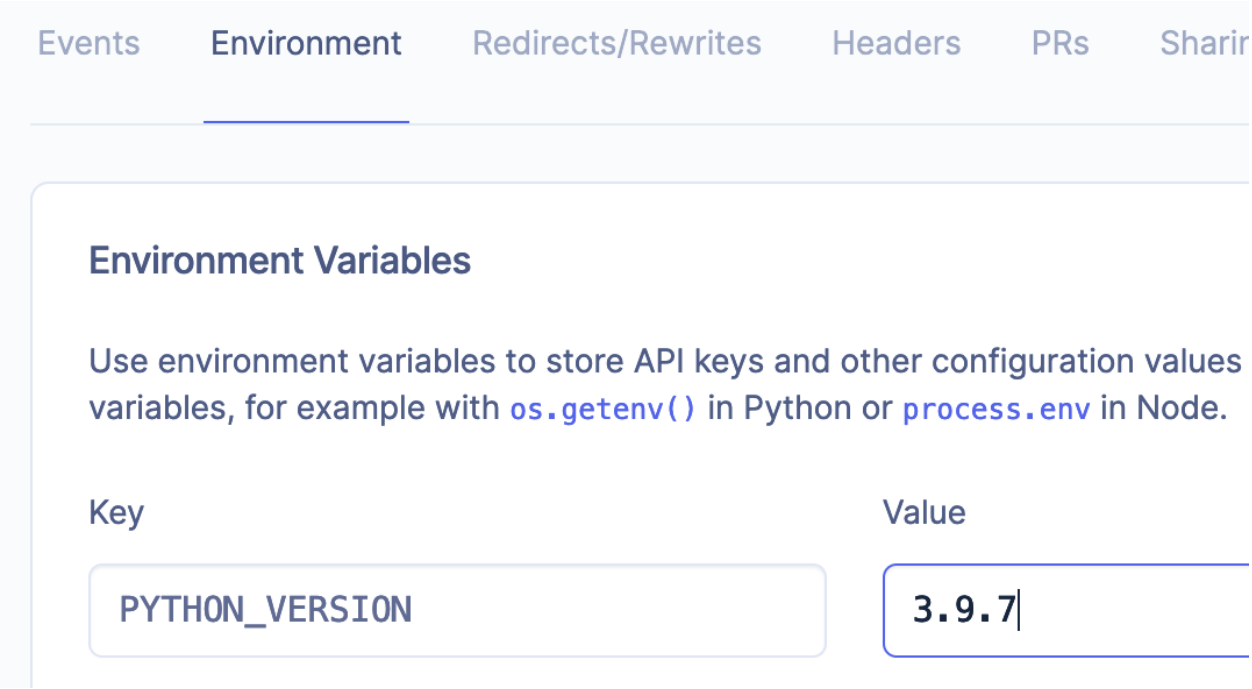
```
def when_ready(server):
    # touch app-initialized when ready
    open("/tmp/app-initialized", "w").close()


bind = "unix:///tmp/nginx.socket"
workers = 3
```

Procfile

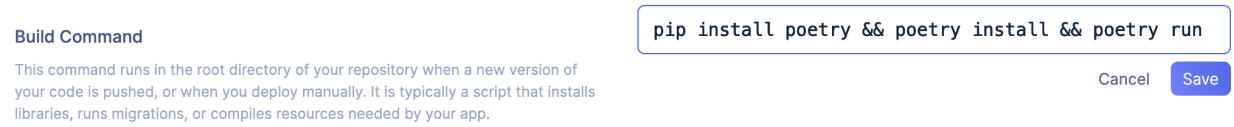
```
web: python app.py collectstatic --noinput && bin/start-nginx gunicorn -c gunicorn.conf.
↪py app:wsgi
```

render.com

- Set the `PYTHON_VERSION` environment variable to the desired Python version (must be at least 3.8)



- Go to `settings` and use `pip install poetry && poetry install && poetry run coltrane build` for the `Build Command`



7.12 CLI

7.12.1 Create

`poetry run coltrane create` sets up a default `coltrane` project.

Generated files

```
.
├── .env
├── .gitignore
├── .watchmanconfig
├── __init__.py
├── app.py
├── content
│   └── index.md
```

(continues on next page)

(continued from previous page)

```
├── data
├── Dockerfile
├── gunicorn.conf.py
├── templates
├── poetry.lock
└── pyproject.toml
```

.env

Example environment variables.

.gitignore

Prevent committing certain files.

.watchmanconfig

Prevent `node_modules` directory from triggering excessive restarts of the development server.

__init__.py

Denote the folder is a Python module.

app.py

The entry point for `coltrane` apps. Similar to a standard `manage.py` file in `Django`.

content

Standard directory for `markdown` files.

data

Standard directory for `JSON` files.

Dockerfile

Example `Dockerfile` for deployment.

gunicorn.conf.py

Example `gunicorn.conf.py` for deployment.

templates

Standard directory for `HTML` template files.

poetry.lock

Lock file for dependencies.

pyproject.toml

Lists dependencies. More details in the Poetry documentation.

Force creation

```
poetry run coltrane create --force
```

Force the creation of a new `coltrane` site even if there is an existing one.

7.12.2 Play

```
poetry run coltrane play
```

Starts a development webserver to render the markdown files into HTML. Defaults to `127.0.0.1:8000`.

Port

The port to use rather than the default `8000`.

`poetry run coltrane play --port 8001` would start the development server at `127.0.0.1:8001`.

7.12.3 Record

```
poetry run coltrane record
```

Builds the static site from `markdown` content and stores the HTML in the `output` directory. Stores static files in the `output/static/` directory.

Incremental builds

By default, `coltrane` will only build markdown files that have changed since the last build. To force re-building all files use `--force`.

```
poetry run coltrane record --force
```

Output directory

By default `coltrane` will write all files to a directory named `output`. But, that can be overriden with `--output`.

```
poetry run coltrane record --output public
```

Multithreaded

By default `coltrane` tries to use the optimal number of threads. But, the number of threads to use can be overriden with `--threads`.

```
poetry run coltrane record --threads 2
```

Ignore errors

By default `coltrane` will exit with a status code of 1 if there is an error while rendering the markdown into HTML. Those errors can be ignore with `--ignore`.

```
poetry run coltrane record --ignore
```

7.13 Environment

For local web development `coltrane` uses an `.env` file in the base directory for potentially sensitive settings. When deployed to production, those settings would be retrieved from environment variables (following the 12-factor app method).

7.13.1 Example `.env` file

```
DEBUG=True
INTERNAL_IPS=127.0.0.1
ALLOWED_HOSTS=example.com
COLTRANE_SITE_URL=https://example.com
SECRET_KEY=this-should-be-lots-of-random-characters
```

7.13.2 Keys

DEBUG

Whether the server is in debug mode or not. Error tracebacks, context, and sensitive information is displayed on the error page when this is set to `True`, so it should always be set to `False` when the app is deployed to production. Defaults to `True` for local development purposes.

INTERNAL_IPS

Used to determine if the current request is internal or not. Must be set for the `debug` template variable to be populated (more information in the Django documentation). Defaults to `127.0.0.1`. If more than one IP is required, separate them by commas.

```
INTERNAL_IPS=127.0.0.1,localhost,192.168.0.1
```

SECRET_KEY

A random string of letters, numbers, and characters. (More information in the Django documentation. Generated automatically when the `.env` file is created. Required.

ALLOWED_HOSTS

The acceptable host or domain names when the site is deployed to production. Required when `DEBUG` is set to `False`. Defaults to `""`. If more than one host name is required, separate them by commas.

```
ALLOWED_HOSTS=coltrane.com
```

COLTRANE_SITE_URL

The hosting domain's scheme and domain. Required.

```
COLTRANE_SITE_URL=https://coltrane.com
```

COLTRANE_TITLE

The title of the website. Required for generating `rss.xml`.

```
COLTRANE_TITLE=Coltrane
```

COLTRANE_DESCRIPTION

The description of the website. Required for generating `rss.xml`.

```
COLTRANE_DESCRIPTION=A simple content site framework that harnesses the power of Django␣
↪without the hassle.
```

COLTRANE_IS_SECURE

Informs `coltrane` that it is served securely, i.e. with SSL with an `https` protocol. This needs to be set to `True` if SSL is provided by a proxy server (for example, Cloudflare). If the site is only served by `https` and you see errors like "403 forbidden CSRF origin didn't match" set this to `True`. Defaults to `False`.

COLTRANE_CONTENT_DIRECTORY

The directory that should be used for `markdown` content. Relative to the base directory. Defaults to "content".

COLTRANE_DATA_DIRECTORY

The directory that should be used for data. Relative to the base directory. Defaults to "data".

COLTRANE_DATA_JSON5

Whether or not data files should be parsed as JSON5. Also requires installing with the `json5` extras (e.g. `poetry add coltrane -E json5` or `pip install coltrane[json5]`). Defaults to `False`.

COLTRANE_DISABLE_WILDCARD_TEMPLATES

To prevent *wildcard templates* from being served, set this to `True`. Defaults to `False`.

CACHE

The type of cache to use for `coltrane`. Acceptable options are: `dummy`, `memory`, `filesystem`, `memcache`, or `redis`. The default is `dummy`.

Note: `filesystem`, `memcache`, and `redis` options require `CACHE_LOCATION` to also be set.

CACHE_LOCATION

The location of the cache. Required for `filesystem`, `memcache`, and `redis` cache options. The `filesystem` cache requires an absolute path. The `memcache` and `redis` cache options include multiple cache servers in a commma-delimited list.

7.14 Settings

Settings for `coltrane` are specified in a `COLTRANE` dictionary in the `settings.py` file. All *env settings* are available to be set directly. Just remove the leading "COLTRANE_" from the environment name if applicable.

```
# settings.py

...

# Sample `coltrane` settings
COLTRANE = {
    "MISTUNE_PLUGINS": [
        "strikethrough",
        "footnotes",
        "table",
        "task_lists",
        "def_list",
        "abbr",
        "mark",
        "insert",
        "superscript",
        "subscript",
    ],
}

...
```

Note: `coltrane` settings can be passed into the `initialize()` method in `app.py` as `kwargs`.

```
# existing app.py file
wsgi = initialize(MARKDOWN_EXTRAS=["metadata",], MISTUNE_PLUGINS=["table",])
# rest of the app.py file
```

7.14.1 Keys

The keys below are specific to the `COLTRANE` dictionary `settings.py`. But, all *env settings* can be used.

MISTUNE_PLUGINS

The features that should be enabled when rendering markdown with `mistune`. A list of all available features: https://mistune.lepture.com/en/latest/plugins.html. The default extras are:

```
[
    "strikethrough",
    "footnotes",
    "table",
    "task_lists",
    "def_list",
    "abbr",
```

(continues on next page)

(continued from previous page)

```
    "mark",
    "insert",
    "superscript",
    "subscript",
]
```

VIEW_CACHE

Caches the rendered HTML when dynamically rendering. Enabled by adding the `SECONDS` key to a `VIEW_CACHE` dictionary. Not used for static sites.

SECONDS

Specifies how long the markdown should be cached when Django is dynamically serving the markdown.

```
COLTRANE = {
    # other settings
    "VIEW_CACHE": {"SECONDS": 60 * 15},
}
```

CACHE_NAME

Specifies a name for the cache to use. Defaults to "default".

```
COLTRANE = {
    # other settings
    "VIEW_CACHE": {"SECONDS": 60 * 15, "CACHE_NAME": "coltrane-view-cache"},
}
```

7.15 Build

`coltrane` will render the `markdown` files into HTML and stored the generated file in an `output` folder. This is similar to other static site generators like `Jekyll` or `Hugo`.

7.15.1 Record

The static site is built with the *`record` command*.

7.16 Installation

1. `poetry add coltrane` or `pip install coltrane`
2. Add `coltrane` to the list of `INSTALLED_APPS` in Django settings file
3. Add `path("", include("coltrane.urls"))`, to the bottom of the `urlpatterns` in the root `urls.py` (i.e. the `urls.py` specified by `ROOT_URLCONF`)

```
# urls.py
from django.urls import include, path

urlpatterns = [
    ...
    path("", include("coltrane.urls")),
]
```

7.17 Integration

7.17.1 Linking

Django templates can link to `coltrane` markdown content with the `url` template tag and the slug of the markdown file.

```
<!-- this will link to a route which renders the /content/about.md markdown file -->
<a href="{% url 'coltrane:content' 'about' %}">About</a>
```

7.17.2 Sitemap

1. Add `"django.contrib.sitemaps"`, to `INSTALLED_APPS` in the settings file
2. Make sure your `TEMPLATES` setting contains a `DjangoTemplates` backend whose `APP_DIRS` options is set to `True`.
3. Add the following to `urls.py`

```
from django.contrib.sitemaps.views import sitemap
from coltrane.sitemaps import ContentSitemap

# Make sure that the protocol is https
ContentSitemap.protocol = "https"

sitemaps = {
    "content": ContentSitemap,
}

urlpatterns = [
    # other URL paths here
    path(
        "sitemap.xml",
        sitemap,
```

(continues on next page)

(continued from previous page)

```
        {"sitemaps": sitemaps},
        name="django.contrib.sitemaps.views.sitemap",
    ),
    # other URL paths here
]
```

More details are in the Django documentation for sitemaps.

7.17.3 RSS

RSS requires an absolute URL so `coltrane` needs to know the domain for the site. The settings file needs to include something similar to the following.

```
COLTRANE = {
    "SITE_URL": "https://example.com",
}
```

URL Routing

1. Add the following to `urls.py`

```
from django.urls import path
from coltrane.feeds import ContentFeed

urlpatterns = [
    path("rss.xml", ContentFeed()),
]
```

7.18 Changelog

7.18.1 0.31.0

- Create example `Dockerfile` and `gunicorn.conf.py` files for easier deployments of `coltrane` apps.
- *Add the ability* to use JSON5 for data files.

Breaking changes

- Remove loading `data.json`. All data should be in JSON files in the `data` directory.
- The default markdown renderer is now `mistune` instead of `markdown2`. The next version of `coltrane` will remove the option to use `markdown2`.

7.18.2 0.30.0

- Add `COLTRANE_IS_SECURE` *env variable*.
- Add `django.middleware.gzip.GZipMiddleware`, `django.middleware.http.ConditionalGetMiddleware`, `django.middleware.csrf.CsrfViewMiddleware` middlewares.

7.18.3 0.29.0

- `django-unicorn` integration.
- Fix: Passing `INSTALLED_APPS` into `init` now does not override the default apps.

7.18.4 0.28.0

- Add `DISABLE_WILDCARD_TEMPLATES` setting.
- Add `data`, `slug`, `template`, and `now` to direct HTML template for as much parity to `markdown` content as possible.

7.18.5 0.27.0

- Support directory wildcards.
- Add `paths` template tag.

7.18.6 0.26.0

- Ability to *configure cache*.
- Allow content or data directory to be specified #48.
- Fix: Handle invalid JSON data #48.

7.18.7 0.25.0

- If a markdown file with a slug cannot be found, look for a template with the same slug. Special case for `*.html` which can be a fall-back option to render for any slug.
- Add `raise_404` template tag.
- Add `last_path` template tag.

7.18.8 0.24.0

- Support Django template tags with the `mistune` markdown renderer.

7.18.9 0.23.1

- Include extra files when building the static site.

7.18.10 0.23.0

- Add `EXTRA_FILE_NAMES` setting to support serving static files like `robots.txt`.

7.18.11 0.22.0

- Add support for rendering markdown with `mistune`. See *MARKDOWN_RENDERED* for how to enable. `mistune` will be the default renderer after 0.22.0 because it is 1) faster rendering markdown than `markdown2`, 2) enables new functionality like `abbr`, 3) fixed a bug in the generation of the tables of contents HTML, and 4) has a plugin architecture to add new features.
- Improve table of contents rendering for `mistune`.

7.18.12 0.21.0

- Add `order_by` to `directory_contents` templatetag.
- Fix `TOC` outputting 'None' when it should be `None`.

7.18.13 0.20.0

- Add `to_html` template tag. #37 by Tobi-De
- Breaking change: change `date` to `publish_date` in metadata. #39 by Tobi-De
- Breaking change: change `SITE` setting to `SITE_URL`.
- Automatically add `verbatim` templatetag around code fences.

7.18.14 0.19.0

- Update project name to `coltrane`.

7.18.15 0.18.3

- Fix bug where templatetags could not be loaded when the base directory was ".".

7.18.16 0.18.2

- Add `request` to the template context when building static sites.

7.18.17 0.18.1

- Fix bug where static site path was incorrect.

7.18.18 0.18.0

- Add `toc` to the template context which provides a table of contents for the markdown.

7.18.19 0.17.0

- Fix bug with relative URLs when generating `sitemap.xml`
- Automatic generation of `rss.xml` file

7.18.20 0.16.1

- Create `COLTRANE_SITE` setting in `.env` file during `create` command

7.18.21 0.16.0

- Output an error if rendering fails during `record` command
- `include_md` template tag
- `parent` filter
- Serving of `/sitemap.xml` for standalone
- Automatic creation of `sitemap.xml` during `record` command

Breaking changes

- `COLTRANE_SITE` is required in .env file

7.18.22 0.15.1

- Include all frontmatter metadata in `directory_contents` template tag output
- Parse `date` frontmatter into `datetime`
- Parse `draft` frontmatter into `boolean`

7.18.23 0.15.0

- *directory_contents* template tag
- Add django-fastdev for more immediate feedback when an invalid template variable is used
- Show error message if a markdown file cannot be output to HTML
- Fix bug where index.md files in a sub-directory were not output correctly

7.18.24 0.14.0

- Add --output option to record command #19 by stlk
- Nicer help output for CLI

7.18.25 0.13.1

- Add --threads option to record command

7.18.26 0.13.0

- Multithread record command
- Better console output for record command

7.18.27 0.12.0

- Fix elapsed time for record command
- More performant collection of markdown content files
- Don't include markdown or data when collecting static files while running record

7.18.28 0.11.0

- Add --force option to create command
- Automatically refresh the browser when markdown content or data is saved

7.18.29 0.10.0

- Fix generating root index.md

7.18.30 0.9.0

- Add support for static files
- Add `watchman` support
- Add `whitenoise` for static handling
- Add `--force` option to `record` command

7.18.31 0.8.0

- Read `INTERNAL_IPS` from .env file
- Add `now` to template variables
- Include found template tags in built-ins
- Include `humanize` template tags in built-ins

7.18.32 0.7.0

- Support nested directories for content and data
- Update default markdown extras

7.18.33 0.6.0

- Add support for markdown frontmatter
- Support custom templates specified in markdown frontmatter

7.18.34 0.5.0

- Add `build` maangement command
- Store build manifest so that HTML doesn't re-render if possible
- Loosen dependencies

7.18.35 0.4.0

- Unit tests, coverage, and fixes for mypy

7.18.36 0.3.0

- Bug fixes

7.18.37 0.2.0

- Bug fixes

7.18.38 0.1.0

- Basic Django app which renders markdown files at a URL
- Basic script