Coil CMS Documentation

Release 1.3.12

Chris Warrick and the Coil contributors

Oct 29, 2018

Contents

The Coil project is retired due to low interest and lack of resources. Instead, we recommend using Dato CMS or Netlify CMS.

We cannot guarantee full compatibility with modern versions of Nikola, or that anything still works with Coil.

CHAPTER 1

Coil CMS

1.1 User documentation for Coil CMS

Welcome to Coil CMS!

Coil CMS is a Content Management System. A Content Management System is used to administer websites. Coil is one of many of them on the market. Coil offers many benefits over the traditional CMSes to both users and administrators. Coil CMS strives to be user-friendly and easy to use.

1.1.1 Getting Started

When you access the Coil CMS app, you first need to log in by using your username and password, which were provided by the administrator. The first thing you should do is visiting the *Account* page and change your password to secure your account.

When you are done securing your account, you can start working.

On the *Index* page, there are two sections: Posts and Pages. You can create new posts or edit existing ones. Depending on the permissions you were given by the administrator, you may or may not be able to edit posts created by other users — this may cause an empty list to be displayed. If you can see posts by others, but would like to hide them, use the buttons on the top of the page to switch between display modes.

Each post displays *Edit* and *Delete* buttons, which can be used to alter contents.

After performing changes, you need to *Rebuild* the page in order to make them visible to the world. Sometimes, you may not be able to perform a rebuild yourself and may need the assistance of someone else. Please check with your organization for more information.

1.1.2 Index Page

On the top of the index page, you can switch between displaying only your own posts and displaying posts of all authors.

Permissions

In order to view posts of other authors, you must have the "can edit others' posts" permission.

In addition, you can set the default view in the *Account* section.

Below, Posts and Pages are displayed in separate columns. On the top of each column, you can create posts by typing their title in the field.

Warning: The post URL will be set based on the post title. It cannot be changed later!

All the posts are displayed under the creation field. You can *edit* or *delete* them.

1.1.3 Editing posts

Permissions

You need the "can edit others' posts" permission to edit posts of other users. You can always edit your own posts.

The *Edit* view lets you make modifications to posts.

The topmost field is the title of the post. You can always change it. Remember that the post URL will **not** change, as it is set in stone during post creation.

The next line features four fields: date, author, tags and category.

Permissions

Post author may be changed only if you have the "transfer post authorship" permission.

Tags are comma-separated. A post can have only **one** category. Many date formats are accepted. The default format used is `yyyy-mm-dd hh:mm:ss UTC±hh:mm`, (the last element being the timezone) but if you fill in your own format, it will usually work.

The next line is the editor toolbar (which also features the Save button). Underneath it, the post content is displayed in a box. You can edit it in any way you like. The editor handles formatting for you, but you can click the </> button to edit the HTML source.

1.1.4 Deleting posts

Permissions

You need the "can edit others' posts" permission to delete posts of other users. You can always delete your own posts.

Clicking on the red *Delete* button under a post will delete the post after confirming with the user.

Warning: Deleting a post is **permanent**.

1.1.5 Rebuilding the site

To have the changes show up on the page, you must rebuild the site.

When the site is in need of rebuilding, the menu bar displays an exclamation mark next to the Rebuild option, the word **Rebuild** also appears in bold.

Clicking on the Rebuild button will initiate a rebuild. The progress will be displayed.

If an error occurs, you will see the output. The error will be displayed in the Advanced information box. If you cannot understand and solve the problem yourself, contact your administrator.

Permissions

In order to rebuild the site, you need the "can rebuild site" permission.

1.1.6 Account

You can edit your account in the Account page. You can change your real name, e-mail and password. You can also set the preferences.

Some sites require you contact an administrator to change your details. In this case, the Account page allows you to generate a hash to let your password be changed securely (your administrator cannot recover the password from the hash).

Warning: The real name is not changed on the posts already created.

Note: If you want to change your username, you must contact an administrator.

Permissions

In order to enable the "Show me posts by other users by default" preference, you must first have the "can edit others' posts" permission.

1.2 Administrator documentation for Coil CMS

1.2.1 Setup

Contents

How Coil works alongside Nikola

Coil requires Nikola to work. Nikola is a static site generator, written in Python. Coil manages the files that are then used by Nikola to build the site.

As such, you must configure Nikola first before you start Coil. You can also use an existing site.

Virtualenv

Create a virtualenv in `/srv/coil` and install Coil, Nikola, uWSGI and rq in it.

```
# virtualenv-2.7 /srv/coil
# cd /srv/coil
# source bin/activate
# pip install nikola coil uwsgi
# pip install 'git+https://github.com/nvie/rq.git#egg=rq'
```

Nikola and `conf.py`

Start by setting up Nikola. This can be done using `nikola init`.

```
# mkdir /srv/coil
# cd /srv/coil
# nikola init my_coil_site
Creating Nikola Site
====================

```

(continues on next page)

(continued from previous page)

```
[a wizard will guide you through configuration]

[2015-01-10T18:16:35Z] INFO: init: Created empty site at my_coil_site.
# cd my_coil_site
```

Then, you must make some changes to the config:

- `COIL_SECRET_KEY` — a bunch of random characters, needed for sessions. **Store it in a safe place** — git is not one! You can use `os.urandom(24)` to generate something good.
- `COIL_URL` — the URL under which Coil can be accessed.
- `_MAKO_DISABLE_CACHING = True`
- Modify `POSTS` and `PAGES`, replacing `.txt` with `.html`.
- You must set the mode (Limited vs Full) and configure it accordingly — see next section for details.

CSS for the site

You must add some CSS for wysihtml5. The easiest way to do this is by downloading the raw `.css` file and saving it as `files/assets/css/custom.css`.

Special config for demo sites

The former demo site used the following two settings, which might also be useful for some environments:

- **`COIL_LOGIN_CAPTCHA` — if you want reCAPTCHA to appear on the login page** (aimed at plugic environments, eg. the demo site), set this to a dict of `{'enabled': True, 'site_key': '', 'secret_key': ''}` and fill in your data. If you don't want a CAPTCHA, don't set this setting.
- `COIL_USERS_PREVENT_EDITING` — list of user IDs (integers) that cannot edit their profiles.

Limited Mode vs. Full Mode

Coil can run in two modes: Limited and Full.

Limited Mode:

- does not require a database, is easier to setup
- stores its user data in `conf.py` (no ability to modify users on-the-fly)
- MUST run as a single process (`processes=1` in uWSGI config)

Full Mode:

- requires Redis and RQ installed and running
- stores its user data in the Redis database (you can modify users on-the-fly)
- may run as multiple processes

Configuring Limited Mode

You need to add the following to your config file:

```
COIL_LIMITED = True
COIL_USERS = {
    '1': {
        'username': 'admin',
        'realname': 'Website Administrator',
        'password': '$bcrypt-sha256$2a,12$St3N7xoStL7Doxpvz78Jve
↪$3vKfveUNhMNhvaFEfJllWEarb5oNgNu',
        'must_change_password': False,
        'email': 'info@getnikola.com',
        'active': True,
        'is_admin': True,
        'can_edit_all_posts': True,
        'wants_all_posts': True,
        'can_upload_attachments': True,
        'can_rebuild_site': True,
        'can_transfer_post_authorship': True,
    },
}
```

The default user is `admin` with the password `admin`. New users can be created by creating a similar dict. Password hashes can be calculated on the *Account* page. Note that you are responsible for changing user passwords (users should provide you with hashes and you must add them manually and restart Coil) — consider not setting `must_change_password` in Limited mode.

Continue with *First build*.

Configuring Full Mode

Full Mode requires much more extra configuration.

Redis

You need to set up a Redis server. Make sure it starts at boot.

RQ

You need to set up a RQ worker. Make sure it starts at boot, after Redis. Here is a sample `.service` file for systemd:

```
[Unit]
Description=RQWorker Service
After=redis.service

[Service]
Type=simple
ExecStart=/srv/coil/bin/rqworker coil
User=nobody
Group=nobody

[Install]
WantedBy=multi-user.target
```

Users

Run `coil write_users`:

```
# coil write_users
Redis URL [redis://]:
Username: admin
Password: admin
```

You will be able to add more users and change the admin credentials (which you should do!) later. See also: *Users*.

conf.py additions

You must add `COIL_LIMITED = False` and `COIL_REDIS_URL`, which is an URL to your Redis database. The accepted formats are:

- `redis://[:password]@localhost:6379/0` (TCP)
- `rediss://[:password]@localhost:6379/0` (TCP over SSL)
- `unix://[:password]@/path/to/socket.sock?db=0` (Unix socket)

The default URL is `redis://localhost:6379/0`.

First build

When you are done configuring Nikola, Coil and any other dependencies, run `nikola build`. This will build an empty Nikola site that can now be hosted outside. You need to do this, because Coil itself uses some assets from this site.

```
# nikola build
```

Permissions

```
# chown -Rf nobody:nobody .
```

Chown `my_coil_site` *recursively* to `nobody`, or whatever user Coil will run as. Coil must be able to write to this directory.

Make sure to fix permissions if you fool around the site directory!

Server

Built-in development server

For testing purposes, or for ad-hoc usage (especially in Limited mode), you can just run `coil devserver`. However, it should **NOT** be used in production. In a public environment, especially in Full mode, you should use uWSGI Emperor and nginx instead.

If you are on Windows and the server crashes, try `python -m coil devserver`.

uWSGI

Sample uWSGI configuration:

Note: `python2` may also be `python`, depending on your environment.

Warning: `processes` **MUST** be set to 1 if running in Limited Mode.

```
[uwsgi]
emperor = true
socket = 127.0.0.1:3031
chdir = /srv/coil/my_coil_site
master = true
threads = 5
binary-path = /srv/coil/bin/uwsgi
virtualenv = /srv/coil
module = coil.web
callable = app
plugins = python2,logfile
uid = nobody
gid = nobody
processes = 3
logger = file:/srv/coil/my_coil_site/uwsgi.log
```

nginx

Sample nginx configuration:

Note: This configuration block assumes you followed the guide. You may need to change the location aliases to match your system.

You should change `server_name` to something you own and can run the server on.

```
server {
    listen 80;
    server_name coil.example.com;
    root /srv/coil/my_coil_site;

    location / {
        include uwsgi_params;
        uwsgi_pass 127.0.0.1:3031;
    }

    location /favicon.ico {
        alias /srv/coil/my_coil_site/output/favicon.ico;
    }

    location /assets {
        alias /srv/coil/my_coil_site/output/assets;
    }
```

(continues on next page)

(continued from previous page)

```
    location /coil_assets {
        alias /srv/coil/lib/python2.7/site-packages/coil/data/coil_assets;
    }

    location /bower_components {
        alias /srv/coil/lib/python2.7/site-packages/coil/data/bower_components;
    }
}
```

Other web servers

You can also use any other web or WSGI server. You must take care of:

- location aliases for `/favicon.ico`, `/assets`, `/coil_assets`, `/bower_components` — see above for sample destinations
- correct process count (must be 1 in Limited mode)

1.2.2 Users

Contents

Users of Coil are stored in the Redis database.

Information stored

The following data about users is stored:

Profile

Name	In Account	Notes
`username`	Username	Used to log in
`realname`	Real name	Prominently displayed on posts
`email`	E-mail address	Used by administrators to contact users
`password`	Password	Hashed and salted using bcrypt

Preferences

Name	In Account	In Permissions
`want_all_posts`	Show me posts of other users by default	Want all posts

Permissions

Coil uses a very granular permission system. Each user can have a different set of permissions, depending on the needs of the organization.

Name	In Account	In Permissions
`active`	n/a	Active
`must_change_password`	Must change password on next login	Must change password
`is_admin`	User is an administrator	Admin
`can_edit_all_posts`	Can edit posts of other users	Can all posts
`can_upload_attachments`	Can upload attachments	Attachments
`can_rebuild_site`	Can rebuild the site	Rebuild
`can_transfer_post_authorship`	Can transfer post authorship	Transfer authorship

Managing users and permissions

All administrators (people with the `is_admin` permission) get access to user management views, accessible from the user menu. They are:

Manage users

This is a table of all users. You can add new users at the bottom by typing in a name and clicking *Create*. You can also *Edit*, *Delete* or *Undelete*.

Bulk import

The last row lets you import a file with user data. TSV (Tab-Separated Values) files are accepted.

The first row MUST contain all the column names. They are:

1. `username`
2. `realname`
3. `email`

4. `password`
5. `active`
6. `is_admin`
7. `must_change_password`
8. `can_edit_all_posts`

9. `want_all_posts` 10 `can_upload_attachments` 11. `can_rebuild_site` 12. `can_transfer_post_authorship`

The following rows should contain data for the users. Passwords should be in plain-text. All the boolean fields (`active` and everything after it) accept `0` or `1` as their value.

Deleting and undeleting users

Even when you press the *Delete* button, the user stays in the database. You can then *Undelete* them if you change your mind.

You could delete the user straight from Redis, but this is **not recommended** and can have unexpected side effects.

Permissions

This is a table of all permissions in the system. It can be used to quickly modify the permission list for groups of users. The teal buttons can be used to select the permission for all users.

1.2.3 Internals

Database schema

User storage

Name	Type	Contents
`users`	hash	Hash mapping usernames to UIDs
`user:uid`	hash	All the data we have on the user (see *Users documentation*)

Caching site

Name	Type	Contents
`site:timeline`	list	list of JSON lists of data needed to initialize a Post
`site:all_posts`	list	list of indexes for `timeline` matching `all_posts`
`site:posts`	list	list of indexes for `timeline` matching `posts`
`site:pages`	list	list of indexes for `timeline` matching `pages`
`site:rev`	string	revision (incremented at each scan; used to determine if updates are needed)
`site:lock`	string	lock on site DB

coil.utils

coil.utils.**ask**(*query*, *default=None*)
Ask a question.

coil.utils.**ask_yesno**(*query*, *default=None*)
Ask a yes/no question.

class coil.utils.**SiteProxy**(*db*, *site*, *logger*)
A proxy for accessing the site in a multiprocessing-safe manner.

all_posts
Get all_posts, reloading the site if needed.

pages
Get pages, reloading the site if needed.

posts
Get posts, reloading the site if needed.

reload_site()
Reload the site from the database.

scan_posts(*really=True*, *ignore_quit=False*, *quiet=True*)
Rescan the site.

timeline
Get timeline, reloading the site if needed.

coil.web

1.3 Appendices

1.3.1 Appendix A. Changelog

Version 1.3.12

1.3.12

- Really fix #49 (forgot to press Merge pull request...)

1.3.11

- Fix bootstrap3 theme detection (Issue #49)

1.3.10

- Add *coil unlock* command
- Fix links to docs

1.3.9

- Work on non-English sites (fix #45)

1.3.8

- Clean up requirements (issue #40)

1.3.7

- Nikola v7.6.4 compatibility

1.3.6

- Patch URLs for HTTPS sites
- Really add descriptions for icons (got lost between branches)

1.3.5

- Add icon descriptions for the navigation bar

1.3.4

- Link to demo site in documentation
- Support reCAPTCHA for logins (for demo site)
- Support preventing some users from editing the site (for demo site)

1.3.3 Remove yesterday's new options. Please do not use v1.3.2.

1.3.2

- Added two options that should not be used, EVER. Please ignore them.

1.3.1

- Use rq from PyPI instead of GitHub

1.3.0

- Python 3 support

1.2.2

- Don't repeat Nikola dependencies to decrease maintenance burden

1.2.1

- Specify deps in `setup.py`

1.2.0

- Added support for a Limited mode, which does not require Redis and rq
- Nikola v7.4.0 compatibility

1.1.0

- Changed hashing mechanism to sha256 + bcrypt. Hashes will be fixed automatically on first login of each user.
- Added `passlib` dependency.
- rqworker queue is now named `coil` (was `default`)
- add trailing slashes to all URLs
- use `url_for()`
- add `/rebuild/force/` (== `nikola build -a`)

1.0.0

- RENAME TO *Coil CMS*
- User documentation
- Form validation
- Redis for storing data

- Rebuilding the site, using RQ as a task queue

0.6.0

- Permission management
- `setup.py` and Python packaging
- `comet` management command
- more modern pages
- multiple issues fixed

0.5.0

- Switch to Flask
- Cookie-based authentication
- User management
- Style wysihtml properly
- Rename to *Comet CMS*

0.0.1

- Initial version
- Called `nikola webapp`
- using bottle
- HTTP Basic "Authentication" (sorry)
- wysihtml

1.3.2 Appendix B. License

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index