Holocron

Release 0.4.0

June 09, 2016

Contents

Holocron is an extendable static blog generator powered by the Force. Like others, it reads text files in various formats, renders them using templates and produces a ready-to-publish static website which could be served by Nginx or another web server. Unlike others, it tries to retrieve as many information as possible from the filesystem. Its core follows the KISS principle and any wheel reinventions were avoided during the process.

Feel the Force? Then use Holocron!

```
$ [sudo] pip3 install holocron
```

CHAPTER 1

Features

Here is an incomplete list of features:

- Supports Markdown, reStructuredText & code syntax highlighting.
- Provides simple & powerful plugin system.
- Supports third-party themes via Jinja2 templates.
- Generates both Atom feed and sitemap.xml.
- Has clear and 100% mobile responsive default theme.
- Supports Google Analytics & Yandex.Metrika counters.
- Has SEO friendly URLs.
- Provides a debug server to preview content.

CHAPTER 2

Table Of Contents

2.1 Quickstart

Eager to get started? Then follow instructions step by step and embrace the power of Holocron.

2.1.1 Installation

Holocron is distributed as a Python package and is available on PyPI. It requires Python 3.3+ and could be installed through the `pip` tool:

```
$ [sudo] pip3 install holocron
```

2.1.2 Create a New Site

Creating a new site with Holocron is a very simple task. Begin by creating a directory for your new site and switching to that directory.

```
$ mkdir -p ~/devel/mysite
$ cd ~/devel/mysite
```

Finish by initializing it with a blog skeleton.

```
$ holocron init
```

Simple, right? :)

2.1.3 Preview Your Site

Well, you have a directory with a Holocron-powered site, and it contains one post and one page. Sounds good, ha? Let's preview them in your browser.

```
$ holocron serve
```

The `serve` command starts a development server and prints to a terminal an http address where your blog is available. In other words, you should just copy-paste this address from the terminal to a browser and that's it.

But the `serve` command does even more - it watches for content changes and rebuilds blog if there're any. That means you don't need to run anything in order to preview latest changes, all you have to do is to refresh a web-page in your browser.

2.1.4 Manage Your Content

In order to understand how to manage Holocron's content, let's take a look at the blog skeleton and example content which was generated by `init` command.

```
|-- 2014
|   `-- 04
|       `-- 20
|           `-- holocron.mdown
|-- images
|   `-- young_obiWan.png
|-- about.mdown
`-- _config.yml
```

Ok, we have four files in the directory, each one represents a separate type. Here's the key:

_config.yml A Holocron's configuration file. You can change here blog title, content paths, enabled extensions, theme options, etc. This is what Holocron's looking for by default, you can pass another filename if you want.

> **Note:** The filename starts with underscore because Holocron do not process any content started with underscore (_) or dot (`.`), and we don't want to see this file in the output directory.

about.mdown A sample page. Each file that could be parsed by one of enabled converters is considered as either Page or Post. Both Page and Post documents preserve their relative paths to the content directory and will be available under almost the same path (a file extension will be stripped) in the output directory. I.e. `$content/a/b/page.mdown` will be available under the `$host/a/b/page/` url.

2014/04/20/holocron.mdown A sample post. Each file that could be parsed by one of enabled converters and has `YEAR/MONTH/DAY/FILE` path pattern is considered a post.

It's processed using almost the same rules as a page, but unlike last one it will appear in the feed and on the index page.

images/young_obiWan.png A sample static file. Each file that could not be parsed by one of active converters is considered as a static file. It doesn't mean it won't be processed at all, but will be copied into output directory "As Is".

What all that means? That means

- If you want to add new page, just drop it anywhere in the content folder in the supported format (default: Markdown & reStructuredText).
- If you want to add new post, just drop it anywhere in the content folder but use directory structure with date representation - `YEAR/MONTH/DAY`.
- If you want to add new picture, just drop it anywhere in the content folder, and it will be copied to output directory "As Is", so you can link to it without hesitation.

2.1.5 Compile Your Site

There's a short command for compiling your site into a set of HTML files, and it's called `build`.

```
$ holocron build
```

By default Holocron's build procedure includes:

- compiling content into HTML;
- generating Atom feed;

- generating sitemap.xml;
- generating index page (with all posts)
- generating index pages by tags (posts by tags)

All results will be placed in the output directory (default: `_output`), so you can share this folder with any web-server.

2.2 Settings

If you're interested in changing Holocron's defaults then you're at the right place! This section covers basic (core) settings available for you.

Holocron configuration is a very simple task. All settings are stored in one YAML file. By default, it's called `_config.yml` and should be located in current working directory. However you can pass any file via CLI (see `--conf` argument).

There are five main sections of settings:

- `site`
- `encoding`
- `paths`
- `theme`
- `ext`

Let's take a closer look!

2.2.1 Site

This section is, probably, most commonly used one. It's responsible for such base things like site 'title', its 'author' and 'URL'.

Example:

```
site:
  title:    Kenobi's Thoughts
  author:   Obi-Wan Kenobi
  url:      http://obi-wan.jedi
```

2.2.2 Encoding

It's a common practice to use Unicode everywhere in general, and UTF-8 as file encoding in particular. Holocron isn't exception here, so UTF-8 is used as default encoding for source content and produced HTMLs. However, if by some reason it doesn't suite you, see the following example how you can change it according to your needs.

Example:

```
encoding:
  content:  utf-8
  output:   utf-8
```

2.2.3 Paths

The `paths` section allows to change default paths used by Holocron for different purposes.

Example:

```
paths:
  content:  {here}/
  output:   {here}/_build
```

where

- `content` – a path where to search for posts, pages, etc
- `output` – a path where to put produced HTMls

The section supports the `{here}` macro which would be resolved into a path to directory with your `_config.yml`.

2.2.4 Theme

Note: Settings listed bellow are **only applied** for default Holocron theme. If you use some third-party one, please check out its documentation.

Well, the section contains theme specific settings. They are passed to templates "As Is".

Example:

```
theme:
  navigation: !!pairs
    - about: /about
    - feed:  /feed.xml

  copyright: >
    &copy; 19 BBY, Obi-Wan Kenobi

  ribbon:
    text:  Star On GitHub
    link:  https://github.com/ikalnitsky/holocron

  twitter_cards:
    username: twitter

  counters:
    google_analytics: XX-XXXXXXXX-X
    yandex_metrika: XXXXXXX
```

where

- `navigation` – a list to be shown on theme's navigation bar; it's usually used for putting top-level pages, or some other useful links such as 'feed' or 'twitter'.
- `copyright` – an HTML text that will be shown in footer section on each web page.
- `ribbon` – a ribbon label that appears on top right corner, and that leads on some page you want to promote (e.g. twitter, github, etc).
- `twitter_cards` – Twitter Cards is a technology for showing rich snippets in tweets if someone posts a link to your site.

- `counters` – setup your counters, and watch the stats about visitors.

2.2.5 Ext

Note: See *Extensions* page for extensions' settings.

The `ext` section is used to enable and configure Holocron's extensions. This documentation page will cover only how to enable them. If you're interested in extensions' setting, please consider the notice above.

In order to enable extension you have to only put its name to the `enabled` subsection.

Example:

```
ext:
  enabled:
    - markdown
    - restructuredtext
    - index
    - feed
    - sitemap
    - tags
    - my-super-puper-extension      # <- inserted by us
```

Warning: You must list explicitly all extensions you want to be enabled. There's no inheritance for them. I.e. you can't do

```
ext:
  enabled:
    - my-super-puper-extension      # <- inserted by us
```

and expect that `markdown`, `feed` and others are enabled.

2.3 Extensions

2.3.1 Markdown

Integrated	yes
Description	Converts Markdown documents into HTML.
File Extensions	`.md`, `.mkd`, `.mdown`, `.markdown`
Syntax	https://daringfireball.net/projects/markdown/syntax

Holocron uses extended version of Markdown syntax, so you'll have fenced code blocks, tables and much more out of the box. Markdown converter could be configured through the `_config.yml`,

```
ext:
  markdown:
    extensions: [ ... ]       # default: [codehilite, extra]
```

where `ext.markdown.extensions` is passed directly to Markdown library. Please check out the full list of supported extensions here:

https://pythonhosted.org/Markdown/extensions/index.html

2.3.2 reStructuredText

Integrated	yes
Description	Converts reStructuredText documents into HTML.
File Extensions	`.rst`, `.rest`
Syntax	http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html

reStructuredText converter doesn't support any options, except ones passed to docutils library. They are tricky, worthless and for nerds. So only a simple example is provided.

```
ext:
  restructuredtext:
    docutils:
      initial_header_level: 2       # start sections from <h2>
```

2.3.3 Creole (Wiki)

Integrated	no, requires holocron-creole to be installed
Description	Converts Creole (Wiki) documents into HTML.
File Extensions	`.creole`
Syntax	http://www.wikicreole.org/wiki/AllMarkup

Creole converter isn't distributed within Holocron due to its license. So you need to install it first before using.

```
$ [sudo] pip3 install holocron-creole
```

Then, just put `creole` to the list of enabled extensions and voilà!

The converter supports only one option:

```
ext:
  creole:
    syntax_highlight: true
```

When it's `true` (default), the `code` macro is provided for sharing code snippets with syntax highlighting.

```
<<code ext=".py">>
    def add(x, y):
        return x + y
<</code>>
```

2.3.4 Feed

Integrated	yes
Description	Generates an Atom feed from posts.

Feed extension generates only an Atom feed, and supports the following options:

```
ext:
  feed:
    save_as: feed.xml
    posts_number: 5
```

where

- `save_as` – an output filename (path relative to output directory)
- `posts_number` – a number of recent posts which appear in the feed

2.3.5 Sitemap

Integrated	yes
Description	Generates a sitemap.xml from your posts and pages.

Produced `sitemap.xml` contains only links to posts and pages; neither tags nor other garbage are in there.

No configuration is allowed, since `sitemap.xml` must be located in the root of output directory.

2.3.6 Index

Integrated	yes
Description	Generates an `index.html` in the output folder.

Index extension is intended to generate so called front page, since it's what users see when they open the site. By default, it generates a list of posts, but one may add a new template to render using the following option:

```
ext:
  index:
    template: document-list.html
```

2.3.7 Tags

Integrated	yes
Description	Generates 'tags' pages with posts.

Tags extension generates an index page for each tag used in blog, which contains a list of posts marked with the tag. It also makes tags clickable in posts. The extension supports the following settings:

```
ext:
  tags:
    template: document-list.html
    output: tags/{tag}
```

where

- `template` – a template to be used to render tags index pages
- `output` – an output directory for tags index page

2.3.8 User Theme

Integrated	yes
Description	Allows to setup external theme for content rendering.

When enabled, the `_theme` folder in current working directory will be consumed as user theme. The path could be changed using the following option:

```
ext:
  user-theme:
    path: {here}/_themes/my-awesome-theme
```

where

- `{here}` – a macros that will be replaced by the path to directory with configuration file (i.e. `_config.yml`)

2.4 Contributing

Important: Thank you for considering contribution to Holocron. Any contribution is greatly appreciated!

There are many ways to contribute to the project. Here are some of them:

- Tell the world about Holocron using social networks and/or blogs.
- Report bugs and propose features.
- Submit bugfixes and enhancements.
- Develop amazing extensions or beautiful themes.
- Improve documentation quality.

Any feedback, any contribution is appreciated. Let's awaken the Force together!

2.4.1 Reporting Bugs

Warning: Bug reports are hugely important! Before you raise one, though, please check Issue Tracker, both open and closed, to confirm that a bug hasn't been reported before. Duplicate bug reports are a huge drain on the time of other contributors, and should be avoided as much as possible.

Each time you are about to report a bug please provide as much information as possible, but keep focusing on things that matters. Usually it'd be enough to specify the following things:

- Steps to reproduce.
- Expected behaviour.
- Actual behaviour.
- Holocron version.
- Python version.
- Operating System.

Since Issue Tracker is used not only to track bugs, please do not forget to attach `bug` tag to your report.

2.4.2 Proposing Features

The best way to propose a feature is to file an issue at Issue Tracker with `enhancement` tag. If you want to have a quick feedback on proposal write it as clear as possible.

2.4.3 Submitting Patches

There are two ways to submit a patch. The first one is by sending a pull request. It's the preferred way and is recommended for general usage. If by some reason you don't have a GitHub account, there's another option - to send a patch via email to project maintainers.

Before you send a patch, please make sure that:

- Your changes are covered by tests.

- `tox` run reports success.

2.4.4 Developing Extensions

Holocron provides a flexible extension system for developers. If you're a frontend web developer and you have a beautiful site template, please do not pass by and create a Holocron theme.

On the other hand, if you miss some converter or generator - you can develop an additional one for Holocron, and share it with others.

Please visit project documentation for details.

2.5 FAQ

2.5.1 Why The Name Holocron?

Holocron (*short for holographic chronicle*) is a device in which Jedi stored different data. In most cases, they used it as diary.

2.5.2 How To Create Stylized Post / Page?

Holocron supports a YAML front matter for posts and pages where you can specify the `template` attribute. The example below renders a blog post using the `yoda.html` template.

```
---
template: yoda.html
---

some content
```

Note: Templates must be available in runtime, and the only way to make it happen is to use either a third party theme or *User Theme*.

2.6 Changes

Here you can see the list of changes between each Holocron release.

2.6.1 0.4.0 (unreleased)

2.6.2 0.3.0 (2016-03-12)

- Added API for registering external themes. That means from now on themes could be distributed as Holocron extensions via Python packaged.
- Added User Theme extension that allows to setup external theme.
- Fixed posts ordering on index and tags pages.
- Fixed broken ribbon, so it's clickable again.

- **DEPRECATED**: `paths.theme` option will be removed in favor of `ext.user-theme.path`
- **DEPRECATED**: user theme won't be enabled by default, so please enable it explicitly

2.6.3 0.2.0 (2015-12-16)

- Added reStructuredText converter.
- Added `holocron.ext.abc.Extension` interface.
- Added Python 3.5 support.
- Added Twitter Cards support.
- Fixed Markdown title parser for documents with multiple <h1> titles..
- Fixed security issue when content author may steal private data through content's meta header.
- Fixed YAML header parser for documents with multiple `---` signs.
- Fixed rebuilding of HTML produced by Tags and Feed extensions.
- Default theme is more responsive for smartphones & tablets now.

2.6.4 0.1.1 (2015-08-22)

- `tests` package is no longer installed.
- `holocron.theme` data is now installed by setuptools.

2.6.5 0.1.0 (2015-08-22)

- First public release.

2.7 Authors

Holocron is written and maintained by these fine people:

2.7.1 Holocron Team

- Igor Kalnitsky <igor@kalnitsky.org>
- Andrii Gamaiunov <andrii.gamaiunov@gmail.com>

2.7.2 Contributors

- Olha Kurkaiedova <olya.kurkaedova@gmail.com>

2.8 License

Holocron is licensed under a three clause BSD License. That means do whatever you want with it as long as the copyright in Holocron sticks around.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Holocron nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.