NtDocutils

0.2.0

👤 Miguel Angel Rivera Notararigo

✉ ntrrgx@gmail.com

🕐 2018-03-06 19:00:00 -04:00

© MIT

NtDocutils is a Material Design Lite theme for Docutils (maybe a little more 😁). This site was built with it, so you may see what you can do, but if you want to see all the styles, you should go to the demo site.

Contents

- Features
- Install
  - From PyPI
  - From source
- Usage
  - Filtering
  - Attachments
  - Printing
  - Theme customization
  - Command line reference
- Uninstall
- Contributing
- Acknowledgment

Features

- All features from Docutils for the `rst2html5.py` writer.

- Special roles for emojis and keyboard keys (see this section).

- Filter content by OS, distributions, categories or any other filters you want (see this section).

- Attachments (see this section), print friendly (see this section) and more from the ⋮ button.

- Theme customization (see this section).

Install

> **Warning** ⚠️
>
> You will need superuser privileges if you don't use a virtualenv.

NtDocutils requires:

- Python 3.4 or above
- Docutils 0.14 (autoinstalled)
- Pygments 2.2.0 (autoinstalled)

From PyPI

```
pip install NtDocutils==0.2.0
```

From source

```
wget -c 'https://github.com/ntrrg/NtDocutils/archive/v0.2.0.tar.gz'

tar -xvf NtDocutils-0.2.0.tar.gz

cd NtDocutils-0.2.0

python3 setup.py
```

Usage

Basically, you have to do two things:

1. Create a `.rst` file:

`example.rst`:

```
 1  ==========
 2  My Article
 3  ==========
 4
 5  :Author: Vultur Gryphus
 6  :Contact: info@vultur.org.ve
 7
 8  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
 9  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
10  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
11  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
12  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
13  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
```

2. Process your file:

```
ntdocutils example.rst example.html
```

And that's it, you already have some like this:

My Article

👤 Vultur Gryphus

✉ info@vultur.org.ve

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

NtDocutils

The following sections cover the usage of some utilities to improve your article and at the end you can see the command line reference.

Filtering

You can filter content just adding the `fl fl-{{ filter name }}` classes in the elements you want filter, **NtDocutils** will create the buttons dynamically at the left bottom corner and set the first filter that it finds as default. E.g:

```
1 .. class:: fl fl-debian
2
3 **Text for Debian**
4
5 .. class:: fl fl-windows
6
7 **Text for Windows**
```

Will result in:

Text for Debian

And creates the following buttons:

Attachments

You can set a list of useful files in the button by adding the following lines in your article:

```
1 .. raw:: html
2
3     <script>
4       ATTACHMENTS = [
5         {
6           url: 'NtDocutils.rst',
7           name: 'NtDocutils 0.2.0.rst',
8           icon: 'code'
9         }
10      ];
11    </script>
```

`ATTACHMENTS` is a global array, it store the files list showed in the 📎 button, each file must be defined as an object with the following properties:

`url` **(string)**

> URL to the file, could be an external link.

`name` **(string)**

> **Optional**. Name displayed at the list, also overwrites the attachment name.

`icon` **(string)**

> **Optional**. Material icon displayed at the list, by default **NtDocutils** uses an icon related to the file extension.

```
1  .. raw:: html
2
3     <script>
4       ATTACHMENTS = [
5         {
6           "url": URL,
7           "name": DISPLAY_NAME,
8           "icon": ICON_NAME
9         },
10        ...
11        {
12          "url": URL_N,
13          "name": DISPLAY_NAME_N,
14          "icon": ICON_NAME_N
15        }
16      ];
17    </script>
```

Printing

There are some special classes that let you improve the way your article is printed when something goes wrong; for example, some content doesn't fit at the page or simply can't be showed as it should in paper. These classes are:

- `.media-screen`: shows the element just in a screen.
- `.media-print`: shows the element just in paper.

Examples:

White spaces for paper (useful for ensure printing format):

```
1  .. Page break
2
3  .. |pb| raw:: html
4
5     <div class="media-print" style="page-break-after: always"></div>
6
7  .. Line break
8
9  .. |lb| raw:: html
10
11    <br class="media-print"/>
```

Display content for specific device:

```
.. Screen

.. raw:: html

   <object data="example.html" type="text/html" height="400px" width="100%"
       class="media-screen">
   </object>

.. Paper

.. image:: images/example.png
   :class: media-print
```

Theme customization

You can use the customize tool from the MDL site to get a custom `.css` with your preferred colors, after that, you must setup some styles by creating a file with the following template:

`customize.css`

```
1  /* Ribbon */
2
3      .ribbon {
4          background-color: {{ Primary color }};
5      }
6
7  /* ... */
```

```
83  /* ... */
84
85  /* Links */
86
87      /*a {
88          color: {{ Accent color }};
89      }*/
90
91  /* ... */
```

The recommended color for the ribbon background (line 4) is the primary color from the theme, you can get this value searching the property `color` at the rule `.mdl-button.mdl-button--colored` in the file you downloaded from MDL (`material.min.css`). The links (line 88) use the accent color from the theme, but in some cases this make them a little unreadable, so you could uncomment it and use the primary color. You should feel free editing the others rules, but usually they will be fine with that values. When you are ready, you have to run **NtDocutils** with the following option:

```
ntdocutils --stylesheet=path/to/material.min.css,path/to/customize.css \
source.rst destination.html
```

Command line reference

```
ntdocutils [-h] [-V] [-S SERVER] SOURCE DESTINATION
```

`-h, --help`

  Shows the help message.

`-V, --version`

  Shows the **NtDocutils** version.

`-S SERVER, --server SERVER`

  Server from where assets will be downloaded. If `local` is passed as value, it will activate the offline mode, this will create a directory with the theme name in the `DESTINATION` parent folder and stores the necessary assets in there.

> **Note**   ▣
> ──
> All options from the `rst2html.py` writer are available.

Uninstall

> **Warning**   ⚠
> ──
> You will need superuser privileges if you didn't use a virtualenv.

Should be enough with this:

```
pip uninstall NtDocutils
```

Contributing

See the contribution guide for more information.

Acknowledgment

Working on this project I use/used:

- Debian
- XFCE
- Sublime
- Chrome
- Terminator
- Zsh
- Git
- EditorConfig
- Github
- Inkscape
- GIMP
- Material Icons
- Roboto
- RawGit

Docutils Team. *reStructuredText.* http://docutils.sourceforge.net/rst.html

Mozilla Developer Network. *JavaScript.* https://developer.mozilla.org/en-US/docs/Web/JavaScript