sphinx-versions

Release 1.1.3

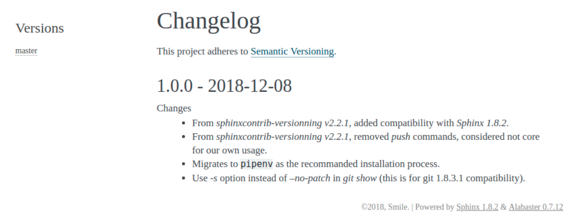
Jul 18, 2019

General

A Sphinx extension that lets you build Sphinx docs for all versions of your project without needing special hosting services.

A Few Examples

alabaster



Versions

master

Changelog

This project adheres to Semantic Versioning.

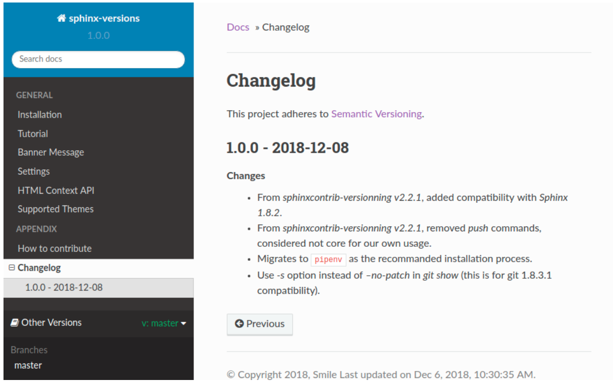
1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -s option instead of –*no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

©2018, Smile. | Powered by Sphinx 1.8.2 & Alabaster 0.7.12

sphinx_rtd_theme



Changelog

This project adheres to Semantic Versioning.

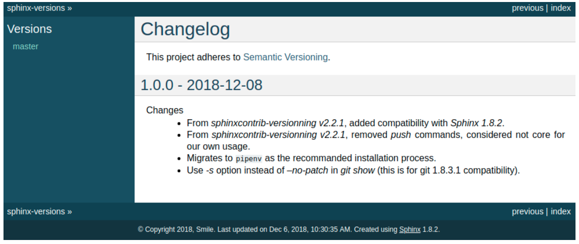
1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -s option instead of –*no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile Last updated on Dec 6, 2018, 10:30:35 AM.

classic



Changelog

This project adheres to Semantic Versioning.

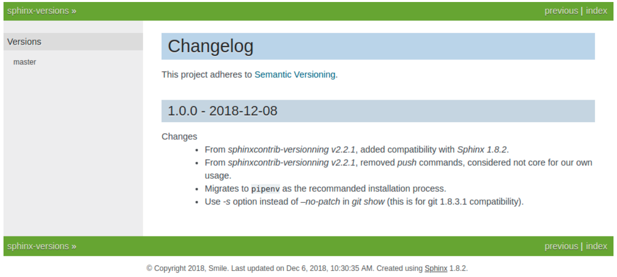
1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -s option instead of –*no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.

nature



Changelog

This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -s option instead of –*no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.

CHAPTER 1

Project Links

- Documentation: https://sphinx-versions.readthedocs.io
- Source code: https://github.com/Smile-SA/sphinx-versions

1.1 Installation

1.1.1 Requirements

Install Python & PIP & pipenv

First, you need to install Python 3 and PIP (if not already present on your system):

```
$ sudo apt-get install python3-pip
```

You might need to update PIP right away

```
$ pip3 install -U pip
```

Then, install `pipenv`

```
$ pip install --user -U pipenv
```

1.1.2 Installation

pipenv install

The suggested way to get *sphinx-versions* is to use pipenv. Simply run this command, from your current project:

```
pipenv install sphinx-versions
```

Clone and Install

Lastly you can also just clone the repo and install from it. Usually you only need to do this if you plan on *contributing* to the project.

```
git clone git@github.com:Smile-SA/sphinx-versions.git
cd sphinx-versions
python setup.py install
```

1.2 Tutorial

This guide will go over the basics of the project.

Make sure that you've already *installed* it.

Note: If you have installed *sphinx-versions* with *pipenv* (which you should), you need to prefix your `sphinx-versioning` commands with `pipenv run ...` or execute them in your virtualenv (see pipenv documentation for more information on this matter).

1.2.1 Building Docs Locally

Before we begin make sure you have some Sphinx docs already in your project. If not, read First Steps with Sphinx first. If you just want something quick and dirty you can do the following:

```
git checkout -b feature_branch master  # Create new branch from master.
mkdir docs  # All documentation will go here (optional, can be anywhere).
echo "master_doc = 'index'" > docs/conf.py  # Create Sphinx config file.
echo -e "Test\n====\n\nSample Documentation" > docs/index.rst  # Create one doc.
git add docs
git commit
git push origin feature_branch  # Required.
```

Note: It is **required** to push doc files to origin. sphinx-versions only works with remote branches/tags and ignores any local changes (committed, staged, unstaged, etc). If you don't push to origin sphinx-versions won't see them. This eliminates race conditions when multiple CI jobs are building docs at the same time.

1.2.2 Build All Versions

Now that you've got docs pushed to origin and they build fine with `sphinx-build` let's try building them with sphinx-versions:

```
sphinx-versioning build -r feature_branch docs docs/_build/html
open docs/_build/html/index.html
```

More information about all of the options can be found at *Settings* or by running with `--help` but just for convenience:

- `-r feature_branch` tells the program to build our newly created/pushed branch at the root of the "html" directory. We do this assuming there are no docs in master yet. Otherwise you can omit this argument.
- `docs/_build/html` is the destination directory that holds generated HTML files.

- The final `docs` argument is the directory where we put our RST files in, relative to the git root (e.g. if you clone your repo to another directory, that would be the git root directory). You can add more relative paths if you've moved the location of your RST files between different branches/tags.

The command should have worked and your docs should be available in *docs/_build/html/index.html* with a "Versions" section in the sidebar.

Note: You can add a *-P pdf-file-name.pdf* option to also generate a pdf of all versions of your documentation

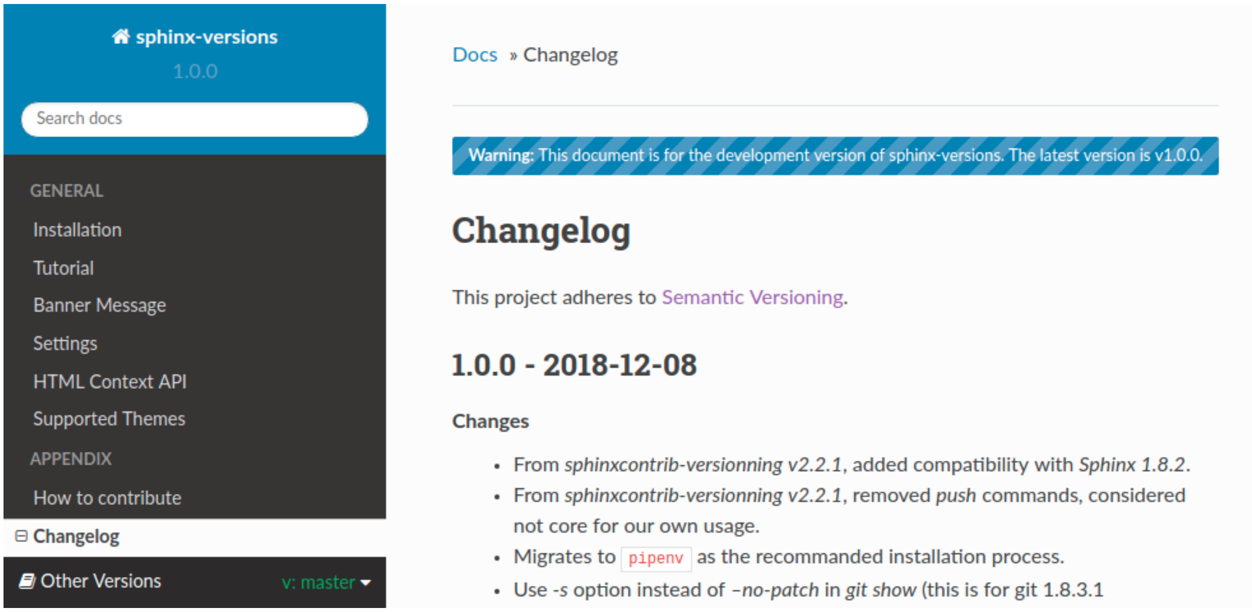
1.3 Banner Message

Banner messages can be displayed at the top of every document informing users that they are currently viewing either old or the development version of the project's documentation, with the exception of the `--banner-main-ref`. This feature is inspired by banner on the Jinja2 documentation.

The banner feature is disabled by default. It can be enabled with the `--show-banner` setting (see the *settings* for more configuration options).

1.3.1 From branch to *main-ref*

The message displayed when users are viewing docs from a branch and the `--banner-main-ref` is a tag. The entire banner is a link that sends users to the latest version of the current page if it exists there.



Warning: This document is for the development version of sphinx-versions. The latest version is v1.0.0.

Changelog

This project adheres to Semantic Versioning.

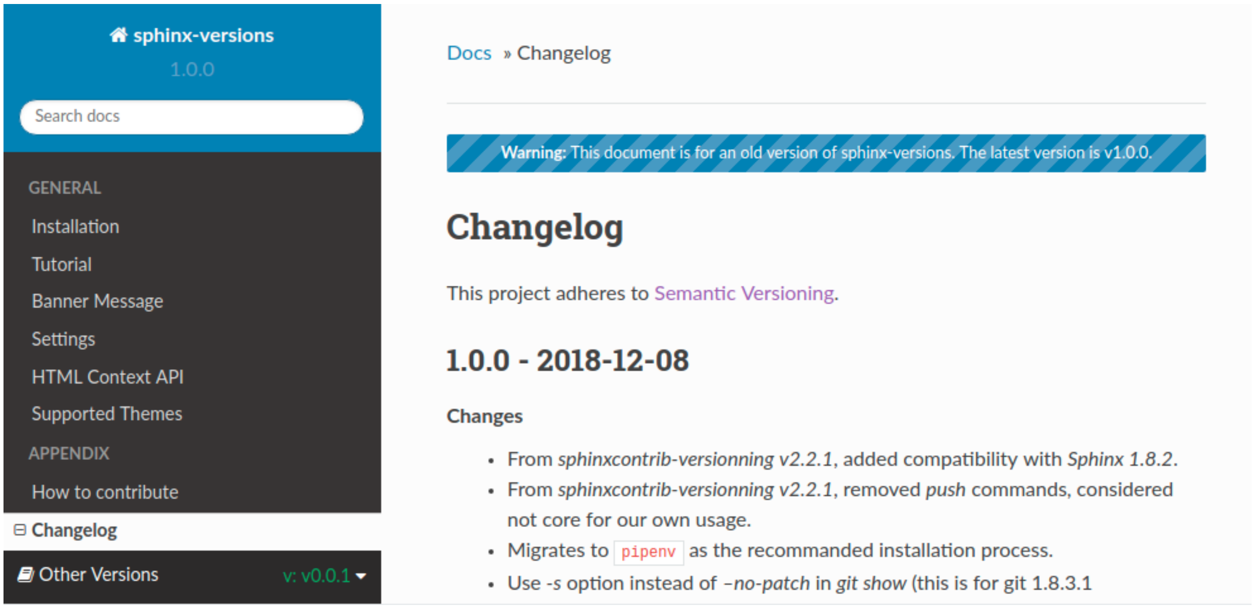
1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1

1.3.2 From old tag to *main-ref*

The message displayed when users are viewing docs from a tag and the `--banner-main-ref` is a tag. Like the message above this one links users to the latest version of the current page.



Warning: This document is for an old version of sphinx-versions. The latest version is v1.0.0.

Changelog

This project adheres to Semantic Versioning.

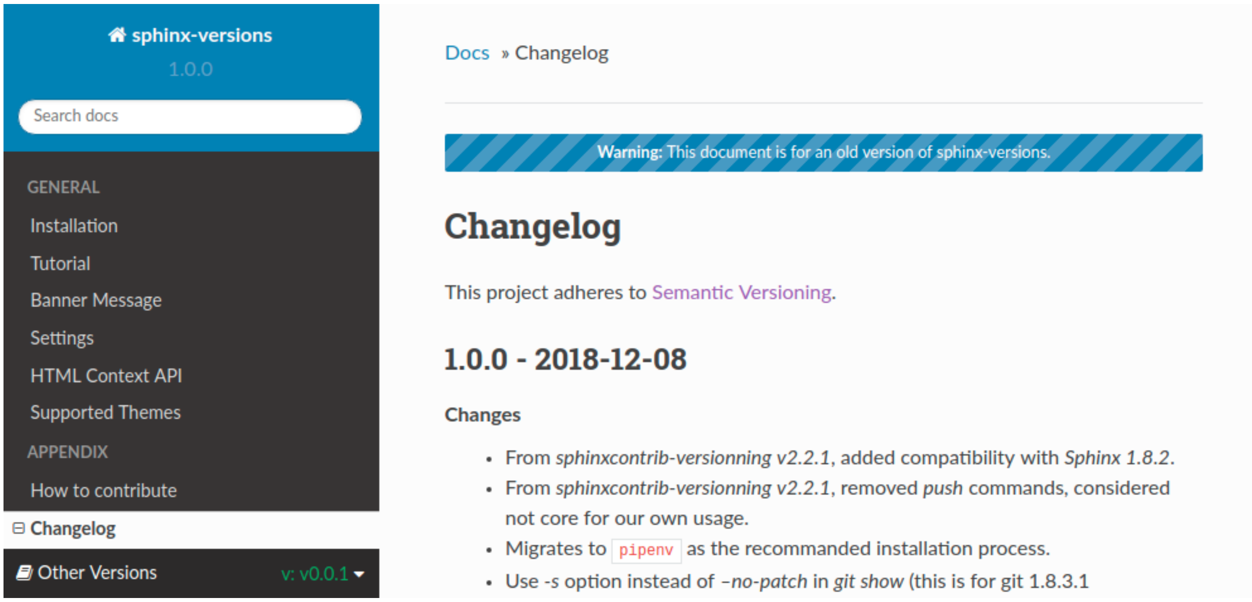
1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1

1.3.3 From a page not existing in the *main-ref*

An example of a banner message from a page that does not exist in the `--banner-main-ref` version. Since there is no page to link to this is just text informing the user that they're viewing the development version of the docs.



Warning: This document is for an old version of sphinx-versions.

Changelog

This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1

1.4 Settings

```
sphinx-versioning [GLOBAL_OPTIONS] build [OPTIONS] REL_SOURCE... DESTINATION
```

sphinx-versions reads settings from two sources:

- Your Sphinx **conf.py** file.

- Command line arguments.

Command line arguments always override anything set in conf.py. You can specify the path to conf.py with the *--local-conf* argument or sphinx-versions will look at the first conf.py it finds in your *REL_SOURCE* directories. To completely disable using a conf.py file specify the *--no-local-conf* command line argument.

Below are both the command line arguments available as well as the conf.py variable names sphinx-versions looks for. All conf.py variable names are prefixed with `scv_`.

An example:

```
# conf.py
author = 'Your Name'
project = 'My Project'
scv_greatest_tag = True
```

1.4.1 Global Options

These options apply to to *build* sub commands. They must be specified before the build command or else you'll get an error.

-c <directory>, **--chdir** <directory>

Change the current working directory of the program to this path.

-g <directory>, **--git-root** <directory>

Path to directory in the local repo. Default is the current working directory.

-l <file>, **--local-conf** <file>

Path to conf.py for sphinx-versions to read its config from. Does not affect conf.py loaded by sphinx-build.

If not specified the default behavior is to have sphinx-versions look for a conf.py file in any *REL_SOURCE* directory within the current working directory. Stops at the first conf.py found if any.

-L, --no-local-conf

Disables searching for or loading a local conf.py for sphinx-versions settings. Does not affect conf.py loaded by sphinx-build.

-N, --no-colors

By default INFO, WARNING, and ERROR log/print statements use console colors. Use this argument to disable colors and log/print plain text.

-v, --verbose

Enable verbose/debug logging with timestamps and git command outputs. Implies *--no-colors*. If specified more than once excess options (number used - 1) will be passed to sphinx-build.

1.4.2 Common Positional Arguments

The *build* sub commands use these arguments.

REL_SOURCE

The path to the docs directory relative to the git root. If the source directory has moved around between git tags you can specify additional directories.

This cannot be an absolute path, it must be relative to the root of your git repository. Sometimes projects move files around so documentation might not always have been in one place. To mitigate this you can specify additional relative paths and the first one that has a **conf.py** will be selected for each branch/tag. Any branch/tag that doesn't have a conf.py file in one of these REL_SOURCEs will be ignored.

--, scv_overflow

It is possible to give the underlying `sphinx-build` program command line options. sphinx-versions passes everything after `--` to it. For example if you changed the theme for your docs between versions and want docs for all versions to have the same theme, you can run:

```
sphinx-versioning build docs docs/_build/html -- -A html_theme=sphinx_rtd_theme
```

This setting may also be specified in your conf.py file. It must be a tuple of strings:

```
scv_overflow = ("-A", "html_theme=sphinx_rtd_theme")
```

1.4.3 Build Arguments

The `build` sub command builds all versions locally. It always gets the latest branches and tags from origin and builds those doc files.

Positional Arguments

In addition to the *common arguments*:

DESTINATION

The path to the directory that will hold all generated docs for all versions.

This is the local path on the file sytem that will hold HTML files. It can be relative to the current working directory or an absolute directory path.

Options

These options are available for the build sub command:

-a, --banner-greatest-tag, scv_banner_greatest_tag

Override banner-main-ref to be the tag with the highest version number. If no tags have docs then this option is ignored and *`--banner-main-ref`* is used.

This setting may also be specified in your conf.py file. It must be a boolean:

```
scv_banner_greatest_tag = True
```

-A, --banner-recent-tag, scv_banner_recent_tag

Override banner-main-ref to be the most recent committed tag. If no tags have docs then this option is ignored and *`--banner-main-ref`* is used.

This setting may also be specified in your conf.py file. It must be a boolean:

```
scv_banner_recent_tag = True
```

-b, --show-banner, scv_show_banner

Show a warning banner. Enables the *Banner Message* feature.

This setting may also be specified in your conf.py file. It must be a boolean:

```
scv_show_banner = True
```

-B <ref>, **--banner-main-ref** <ref>, **scv_banner_main_ref**

The branch/tag considered to be the latest/current version. The banner will not be displayed in this ref, only in all others. Default is **master**.

If the banner-main-ref does not exist or does not have docs the banner will be disabled completely in all versions. Docs will continue to be built.

This setting may also be specified in your conf.py file. It must be a string:

```
scv_banner_main_ref = 'feature_branch'
```

-i, --invert, scv_invert

Invert the order of branches/tags displayed in the sidebars in generated HTML documents. The default order is whatever git prints when running "**git ls-remote –heads –tags**".

This setting may also be specified in your conf.py file. It must be a boolean:

```
scv_invert = True
```

-p <kind>, **--priority** <kind>, **scv_priority**

`kind` may be either **branches** or **tags**. This argument is for themes that don't split up branches and tags in the generated HTML (e.g. sphinx_rtd_theme). This argument groups branches and tags together and whichever is selected for `kind` will be displayed first.

This setting may also be specified in your conf.py file. It must be a string:

```
scv_priority = 'branches'
```

-r <ref>, **--root-ref** <ref>, **scv_root_ref**

The branch/tag at the root of *`DESTINATION`*. Will also be in subdirectories like the others. Default is **master**.

If the root-ref does not exist or does not have docs, `sphinx-versioning` will fail and exit. The root-ref must have docs.

This setting may also be specified in your conf.py file. It must be a string:

```
scv_root_ref = 'feature_branch'
```

-s <value>, **--sort** <value>, **scv_sort**

Sort versions by one or more certain kinds of values. Valid values are `semver`, `alpha`, and `time`.

You can specify just one (e.g. "semver"), or more. The "semver" value sorts versions by Semantic Versioning, with the highest version being first (e.g. 3.0.0, 2.10.0, 1.0.0). Non-semver branches/tags will be sorted after all valid semver formats. This is where the multiple sort values come in. You can specify "alpha" to sort the remainder alphabetically or "time" to sort chronologically (most recent commit first).

This setting may also be specified in your conf.py file. It must be a tuple of strings:

```
scv_sort = ('semver',)
```

-t, --greatest-tag, scv_greatest_tag

Override root-ref to be the tag with the highest version number. If no tags have docs then this option is ignored and *`--root-ref`* is used.

This setting may also be specified in your conf.py file. It must be a boolean:

```
scv_greatest_tag = True
```

-T, --recent-tag, scv_recent_tag

Override root-ref to be the most recent committed tag. If no tags have docs then this option is ignored and *`--root-ref`* is used.

This setting may also be specified in your conf.py file. It must be a boolean:

```
scv_recent_tag = True
```

-w <pattern>, **--whitelist-branches** <pattern>, **scv_whitelist_branches**

Filter out branches not matching the pattern. Can be a simple string or a regex pattern. Specify multiple times to include more patterns in the whitelist.

This setting may also be specified in your conf.py file. It must be a tuple of either strings or `re.compile()` objects:

```
scv_whitelist_branches = ('master', 'latest')
```

-W <pattern>, **--whitelist-tags** <pattern>, **scv_whitelist_tags**

Same as *--whitelist-branches* but for git tags instead.

This setting may also be specified in your conf.py file. It must be a tuple of either strings or `re.compile()` objects:

```
scv_whitelist_tags = (re.compile(r'^v\d+\.\d+\.\d+$'),)
```

1.5 HTML Context API

The following Jinja2 context variables are exposed in the Sphinx HTML builder context in all versions.

1.5.1 Versions Iterable

`versions` is the main variable of interest. It yields names of other (and the current) versions and relative URLs to them. You can iterate on it to get all branches and tags, or use special properties attached to it to yield just branches or just tags.

versions

An iterable that yields 2-item tuples of strings. The first item is the version (branch/tag) name while the second item is the relative path to the documentation for that version. The path is URL safe and takes into account HTML pages in sub directories.

```
{%- for name, url in versions %}
    <li><a href="{{ url }}">{{ name }}</a></li>
{%- endfor %}
```

versions.**branches**

The `versions` iterable has a **branches** property that itself yields versions in branches (filtering out git tags). The order is the same and it yields the same tuples.

```
<dl>
    <dt>Branches</dt>
    {%- for name, url in versions.branches %}
    <dd><a href="{{ url }}">{{ name }}</a></dd>
    {%- endfor %}
</dl>
```

versions.**tags**

The `versions` iterable also has a **tags** property that itself yields versions in tags (filtering out git branches). Just as the **branches** property the order is maintained and the yielded tuples are the same.

```
<dl>
    <dt>Tags</dt>
    {%- for name, url in versions.tags %}
    <dd><a href="{{ url }}">{{ name }}</a></dd>
    {%- endfor %}
</dl>
```

1.5.2 Functions

vhasdoc(*other_version*)

Similar to Sphinx's hasdoc() function. Returns True if the current document exists in another version.

```
{% if vhasdoc('master') %}
    This doc is available in <a href="../master/index.html">master</a>.
{% endif %}
```

vpathto(*other_version*)

Similar to Sphinx's pathto() function. Has two behaviors:

1. If the current document exists in the specified other version pathto() returns the relative URL to that document.

2. If the current document does not exist in the other version the relative URL to that version's master_doc is returned instead.

```
{% if vhasdoc('master') %}
    This doc is available in <a href="{{ vpathto('master') }}">master</a>.
{% else %}
    Go to <a href="{{ vpathto('master') }}">master</a> for the latest docs.
{% endif %}
```

1.5.3 Banner Variables

These variables are exposed in the Jinja2 context to facilitate displaying the banner message and deciding which message to display.

scv_banner_greatest_tag

A boolean set to True if *--banner-greatest-tag* is used.

scv_banner_main_ref_is_branch

A boolean set to True if the banner main ref is a branch.

scv_banner_main_ref_is_tag

A boolean set to True if the banner main ref is a tag.

scv_banner_main_version

A string, the value of *--banner-main-ref*.

scv_banner_recent_tag

A boolean set to True if *--banner-recent-tag* is used.

scv_show_banner

A boolean set to True if *--show-banner* is used.

1.5.4 Other Variables

current_version
A string of the current version being built. This will be the git ref name (e.g. a branch name or tag name).

```
<h3>Current Version: {{ current_version }}</h3>
```

scv_is_branch
A boolean set to True if the current version being built is from a git branch.

scv_is_greatest_tag
A boolean set to True if the current version being built is:

- From a git tag.
- A valid semver-formatted name (e.g. v1.2.3).
- The highest version number.

scv_is_recent_branch
A boolean set to True if the current version being built is a git branch and is the most recent commit out of just git branches.

scv_is_recent_ref
A boolean set to True if the current version being built is the most recent git commit (branch or tag).

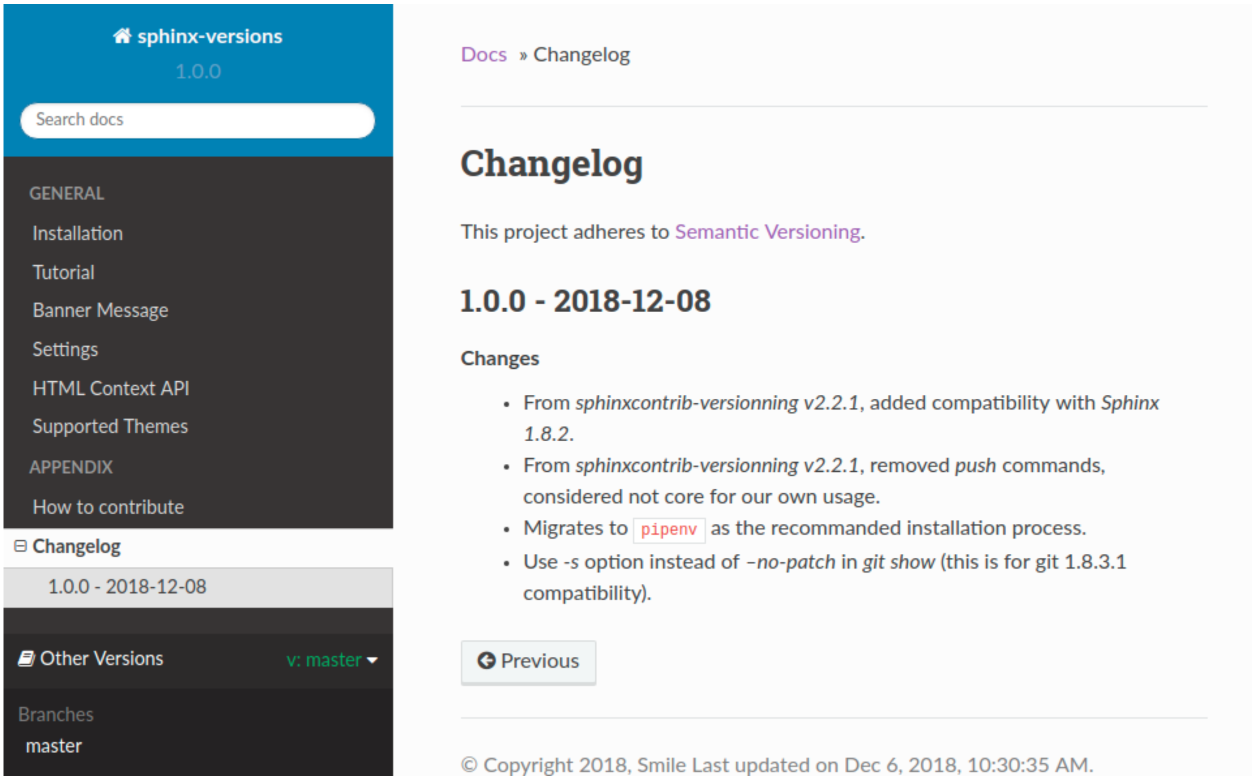
scv_is_recent_tag
A boolean set to True if the current version being built is a git tag and is the most recent commit out of just git tags.

scv_is_root
A boolean set to True if the current version being built is in the web root (defined by `--root-ref`).

scv_is_tag
A boolean set to True if the current version being built is from a git tag.

1.6 Supported Themes

Below are screen shots of the supported built-in Sphinx themes. You can the "Versions" section in each screen shot on sidebars.



Changelog

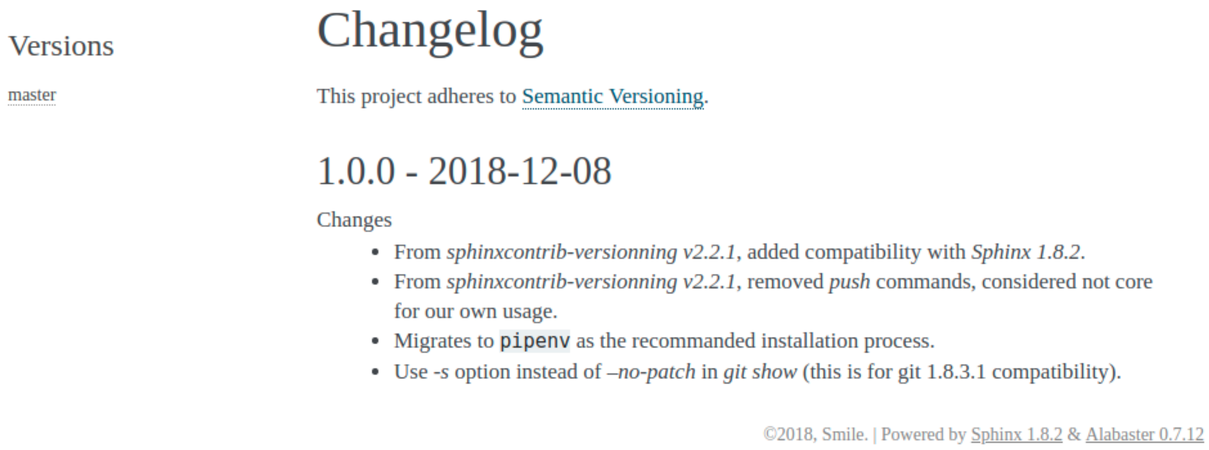
This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile Last updated on Dec 6, 2018, 10:30:35 AM.



Changelog

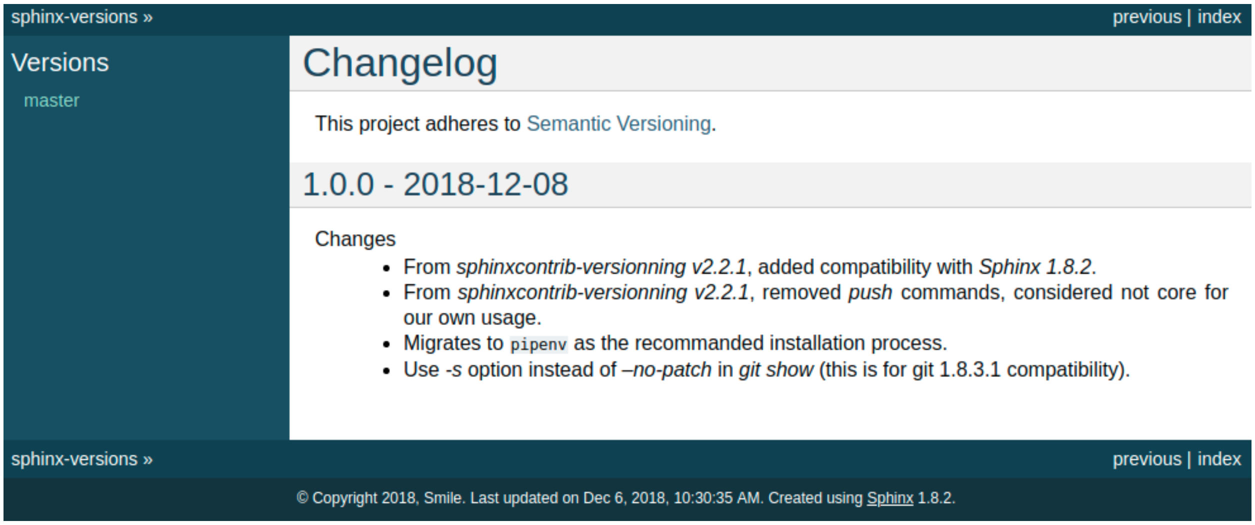
This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -s option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

©2018, Smile. | Powered by Sphinx 1.8.2 & Alabaster 0.7.12



Changelog

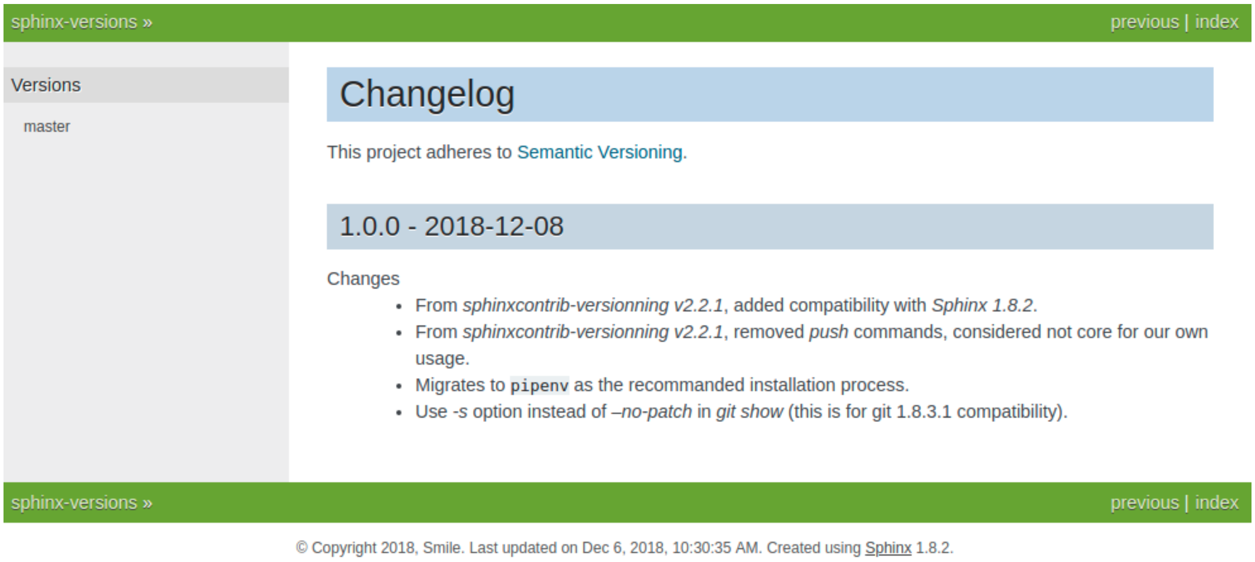
This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -*s* option instead of –*no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.



Changelog

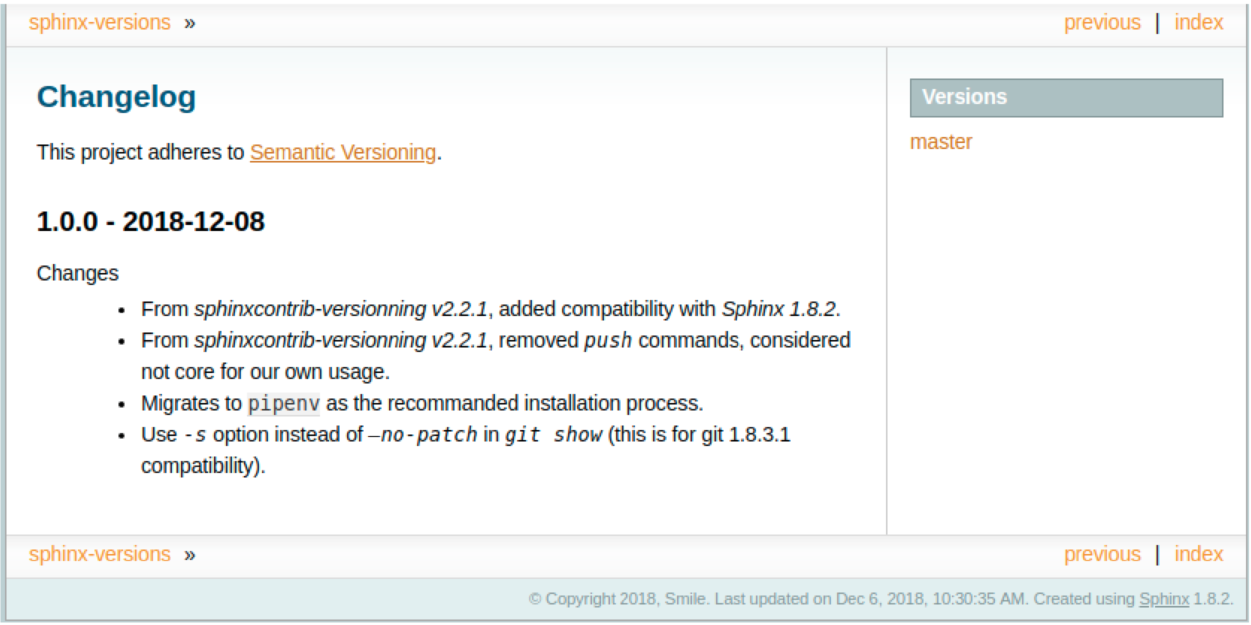
This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -*s* option instead of –*no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.



Changelog

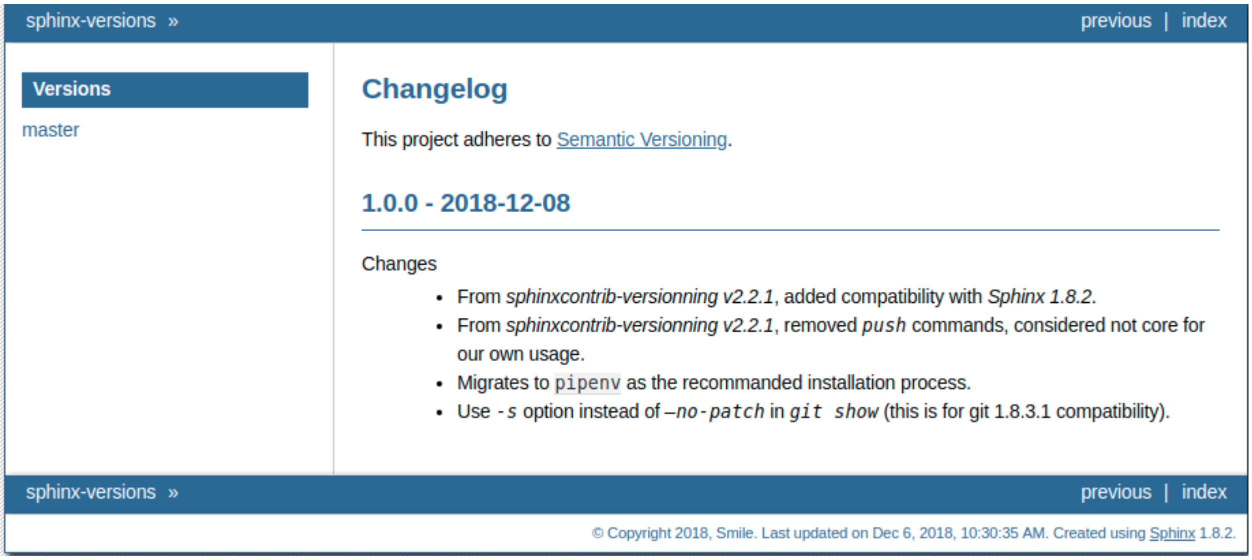
This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.



Changelog

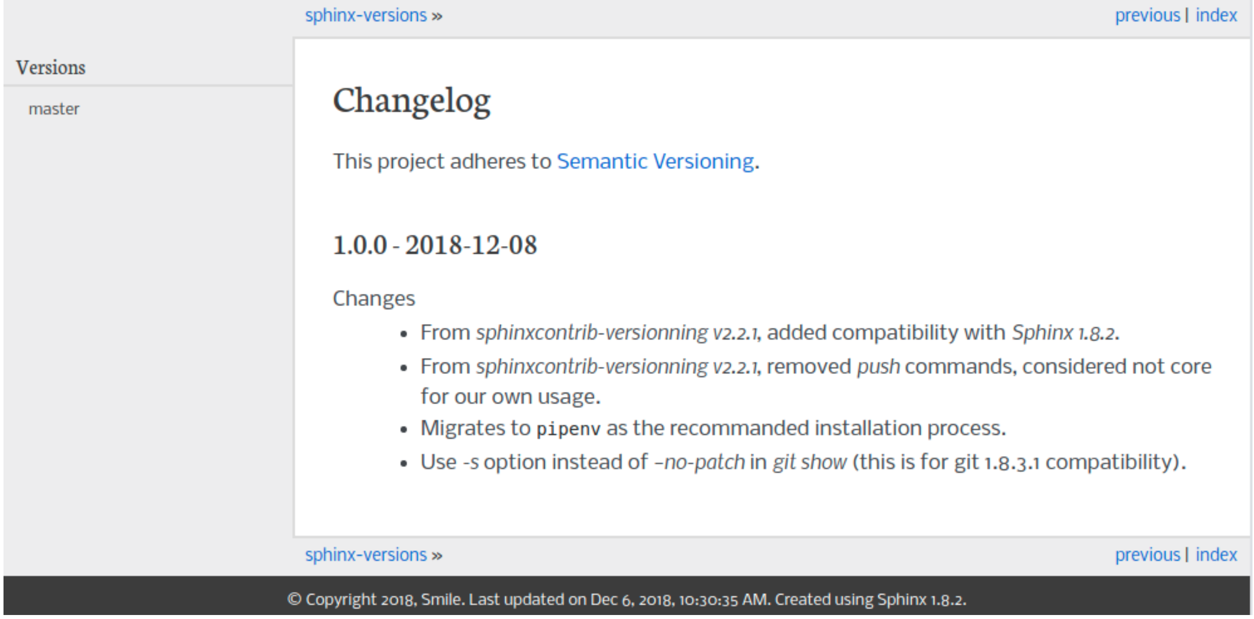
This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.



Changelog

This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning* v2.2.1, added compatibility with *Sphinx* 1.8.2.
- From *sphinxcontrib-versionning* v2.2.1, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use -s option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.

1.6.1 sphinx_rtd_theme

1.6.2 alabaster

1.6.3 classic

1.6.4 nature

1.6.5 sphinxdoc

1.6.6 bizstyle

1.6.7 pyramid

1.6.8 traditional

1.7 How to contribute

Make sure you meet the *requirements to use* the project, first.

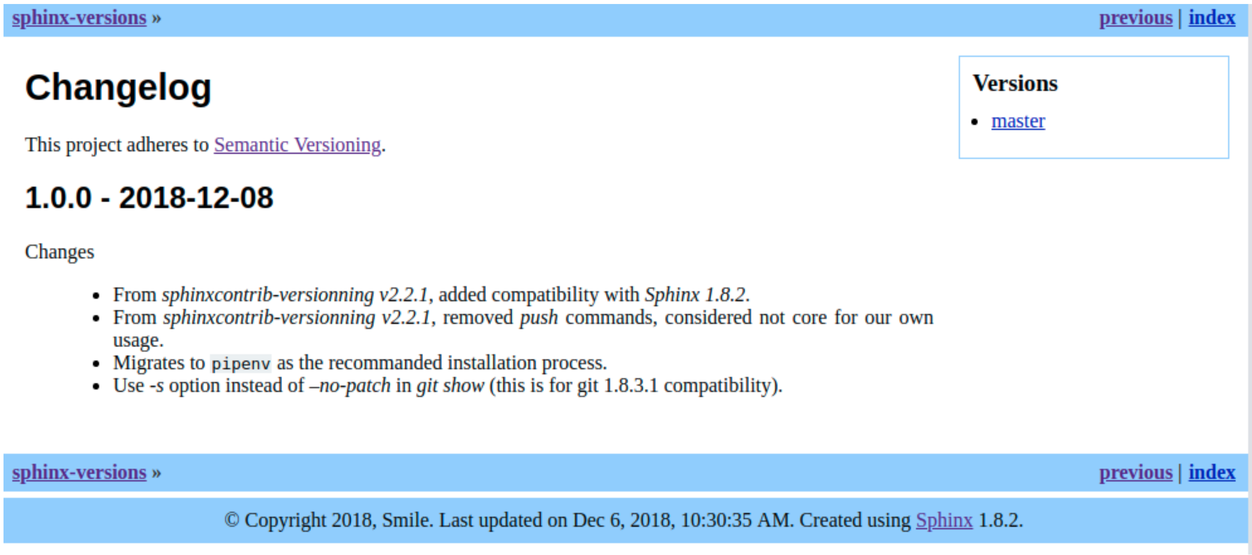
1.7.1 Requirements

Install the virtualenv dependencies

To install required dependencies, use the command:

```
$ pipenv install --dev
```

This will install in your local virtualenv all the required dependencies to contribute to this project.



Changelog

This project adheres to Semantic Versioning.

1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.
- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

Versions

- master

© Copyright 2018, Smile. Last updated on Dec 6, 2018, 10:30:35 AM. Created using Sphinx 1.8.2.

The `--dev` option allows the installation of the *dev-packages* dependencies.

Install build and distribution tools

- Install latest version of required tools

```
pip install --user -U setuptools wheel twine
```

1.7.2 Build and upload a new version of sphinx-versions

Update the version

You need to update two different files :

- `setup.py`: contains the *VERSION* constant, used to identify the version built and uploaded to the nexus.
- `sphinxcontrib/versioning/__init__.py`: contains the `__version__` constant, used to identify the package version.

Update the README.rst

You need to add a section in the `README.rst` for the newly created version (follow the pattern of other versions).

Generate package to distribute

This builds your python project and creates the *dist* directory (among other things).

```
python3 setup.py sdist bdist_wheel
```

Upload your package to nexus

```
twine upload dist/*
```

After this command, your package is available on https://pypi.org. Anyone can install it using *pip install sphinx-versions*.

1.8 Changelog

This project adheres to Semantic Versioning.

1.8.1 1.1.3 - 2019-07-18

Changes

- Fix pdf copy target directory

1.8.2 1.1.2 - 2019-07-18

Changes

- Removes the rest of *unicode* function calls

1.8.3 1.1.1 - 2019-07-18

Changes

- Removes compatibility with Python 2 (and make it work properly on Pyhton 3 : removing a call to *unicode* function)

1.8.4 1.1.0 - 2019-07-18

Changes

- Add *-P pdf-file-name.pdf* option, thanks to *Anybotics fork <https://github.com/ANYbotics/sphinx-versions>*

1.8.5 1.0.1 - 2019-04-23

Changes

- Update sphinx version from 1.8.4 to 1.8.5

1.8.6 1.0.0 - 2018-12-08

Changes

- From *sphinxcontrib-versionning v2.2.1*, added compatibility with *Sphinx 1.8.2*.
- From *sphinxcontrib-versionning v2.2.1*, removed *push* commands, considered not core for our own usage.

- Migrates to `pipenv` as the recommanded installation process.
- Use *-s* option instead of *–no-patch* in *git show* (this is for git 1.8.3.1 compatibility).

Index