sphinxcontrib-towncrier

Release 0.2.0a1.dev11+g56826fe.d20210529

Sviatoslav Sydorenko

May 29, 2021

GO DEEPER:

An RST directive for injecting a Towncrier-generated changelog draft containing fragments for the unreleased (next) project version.

CHAPTER

ONE

HOW TO USE THIS?

```
$ pip install sphinxcontrib-towncrier
```

```
extensions = ['sphinxcontrib.towncrier']

# Options: draft/sphinx-version/sphinx-release
towncrier_draft_autoversion_mode = 'draft'
towncrier_draft_include_empty = True
towncrier_draft_working_directory = PROJECT_ROOT_DIR
# Not yet supported:
# towncrier_draft_config_path = 'pyproject.toml'  # relative to cwd
```

Make sure to point to the dir with `pyproject.toml` and pre-configure towncrier itself in the config.

If everything above is set up correctly, you should be able to add

```
.. towncrier-draft-entries::
```

to your documents, like `changelog.rst`. With no argument, the version title will be generated using the strategy set up in the `towncrier_draft_autoversion_mode` setting.

If you want to be in control, override it with an argument you like:

```
.. towncrier-draft-entries:: |release| [UNRELEASED DRAFT]
```

Native RST substitutions in the argument work, just make sure to declare any non-default ones via `rst_epilog` or at the end of the document where the `towncrier-draft-entries` directive is being used.

CHAPTER

TWO

DOES ANYBODY ACTUALLY USE THIS?

So far we know about two projects using `sphinxcontrib-towncrier` — ansible/pylibssh and pypa/pip. Also, this Sphinx extension is inspired by and somewhat based on the ideas used in pytest-dev/pytest and tox-dev/tox. We believe that these projects are full of wonderful tricks that you may want to explore regardless of whether you'll use our project.

2.1 Advanced changelog layout setup

This page attempts do demonstrate one way of setting up a docs site for use with sphinxcontrib-towncrier and includes a few opinionated integration solutions. But of course, this layout may not fit everybody's needs. When configuring your project, try to figure out what works for you — you don't have to follow everything laid out here blindly.

2.1.1 Project structure

The author likes the following project directory layout:

```
{{ project_root }}/
│
├─ docs/
│   ├─ changelog.d/
│   │   ├─ .gitignore
│   │   ├─ .towncrier-template.rst.j2
│   │   ├─ {{ issue_number }}.{{ changelog_fragment_type }}.rst
│   │   ├─ ...
│   │   └─ README.rst
│   ├─ changelog.rst
│   ├─ conf.py
│   ├─ index.rst
│   ├─ requirements.in
│   └─ requirements.txt
├─ src/
│   └─ {{ python_importable_name }}/
│       └─ ...
├─ .readthedocs.yml
├─ CHANGELOG.rst
├─ pyproject.toml
├─ README.rst
├─ tox.ini
└─ ...
```

This is an src-layout project with a Python package located at src/{{ python_importable_name }}/ but we won't touch this topic. There are several automation, configuration, documentation and metadata files in the project root that will be described later on this page. Finally, a Sphinx-based site is located under the docs/.

The rest of this page will describe what to have in each of those files.

2.1.2 docs/changelog.d/

Let's start with the docs/changelog.d/. This is a folder where the end-users are supposed to add their changelog fragments for Towncrier to consume.

docs/changelog.d/.gitignore

First, let's make sure Git only tracks files that we want there by adding a .gitignore file in this folder. First thing, it adds everything to "ignore" but then allows .gitignore, .gitignore, README.rst and any RST documents matching Towncrier change note fragment naming convention.

```
*

!.gitignore
!.towncrier-template.rst.j2
!*.*.rst
!README.rst
```

docs/changelog.d/.towncrier-template.rst.j2

Then, there's .towncrier-template.rst.j2. It's a changelog template, for Towncrier to use. It can be copied from https://github.com/twisted/towncrier/tree/master/src/towncrier/templates. This name is set in pyproject.toml in the project root.

docs/changelog.d/{{ issue_number }}.{{ changelog_fragment_type }}.rst

These are changelog fragments in RST format. They are absorbed by Towncrier during the release and before that, these files will be used in the preview generated by sphinxcontrib-towncrier.

docs/changelog.d/README.rst

This README.rst file would normally contain — a guide for the contributors on how to write change notes. For example, setuptools has a useful write-up on authoring changelog fragments. It is useful to have it in this place so that it shows up on GitHub when the users navigate to the folder with the fragments via the web UI.

2.1.3 docs/

docs/changelog.rst

This is a Sphinx page that contains both the future version changelog preview via `.. towncrier-draft-entries::` directive and the changelog for all already released versions that is managed by Towncrier in a separate RST document `CHANGELOG.rst` in the project root.

```
*********
Changelog
*********

Versions follow `Semantic Versioning`_ (``<major>.<minor>.<patch>``).
Backward incompatible (breaking) changes will only be introduced in major
versions with advance notice in the **Deprecations** section of releases.

.. _Semantic Versioning: https://semver.org/

.. towncrier-draft-entries:: |release| [UNRELEASED DRAFT] as on |today|

.. include:: ../CHANGELOG.rst
```

docs/conf.py

The Sphinx configuration demonstrates how to keep the version information known to Sphinx in sync with the Git tag based metadata. Note the exclusion of `docs/changelog.d/` and the settings prefixed with `towncrier_draft_`.

```
"""Configuration for the Sphinx documentation generator."""

from functools import partial
from pathlib import Path

from setuptools_scm import get_version


# -- Path setup --------------------------------------------------------------

PROJECT_ROOT_DIR = Path(__file__).parents[1].resolve()
get_scm_version = partial(get_version, root=PROJECT_ROOT_DIR)


# -- Project information -----------------------------------------------------

github_url = 'https://github.com'
github_repo_org = 'your-org'
github_repo_name = 'your-project'
github_repo_slug = f'{github_repo_org}/{github_repo_name}'
github_repo_url = f'{github_url}/{github_repo_slug}'
github_sponsors_url = f'{github_url}/sponsors'

project = github_repo_name
author = f'{project} Contributors'
```

(continues on next page)

(continued from previous page)

```
copyright = f'2021, {author}'

# The short X.Y version
version = '.'.join(
    get_scm_version(
        local_scheme='no-local-version',
    ).split('.')[:3],
)

# The full version, including alpha/beta/rc tags
release = get_scm_version()

rst_epilog = f"""
.. |project| replace:: {project}
"""


# -- General configuration ---------------------------------------------------

extensions = [
    # Built-in extensions:
    'sphinx.ext.extlinks',
    'sphinx.ext.intersphinx',

    # Third-party extensions:
    'sphinxcontrib.towncrier',  # provides `towncrier-draft-entries` directive
]

exclude_patterns = [
    '_build', 'Thumbs.db', '.DS_Store',  # <- Defaults
    'changelog.d/**',  # Towncrier-managed change notes
]


# -- Options for HTML output -------------------------------------------------

html_theme = 'furo'


# -- Extension configuration -------------------------------------------------

# -- Options for intersphinx extension ---------------------------------------

intersphinx_mapping = {
    'python': ('https://docs.python.org/3', None),
    'rtd': ('https://docs.rtfd.io/en/stable', None),
    'sphinx': ('https://www.sphinx-doc.org/en/master', None),
}

# -- Options for extlinks extension ------------------------------------------

extlinks = {
```

(continues on next page)

(continued from previous page)

```
    'issue': (f'{github_repo_url}/issues/%s', '#'),
    'pr': (f'{github_repo_url}/pull/%s', 'PR #'),
    'commit': (f'{github_repo_url}/commit/%s', ''),
    'gh': (f'{github_url}/%s', 'GitHub: '),
    'user': (f'{github_sponsors_url}/%s', '@'),
}

# -- Options for towncrier_draft extension -----------------------------------

towncrier_draft_autoversion_mode = 'draft'  # or: 'sphinx-version', 'sphinx-release'
towncrier_draft_include_empty = True
towncrier_draft_working_directory = PROJECT_ROOT_DIR
# Not yet supported: towncrier_draft_config_path = 'pyproject.toml'  # relative to cwd

# -- Strict mode -------------------------------------------------------------

default_role = 'any'

nitpicky = True
```

docs/index.rst

The root document includes most of the README excluding one badge and its title. It allows to flexibly control what information goes to the PyPI and GitHub repo pages and what appears in the docs.

This document must contain a `.. toctree::` directive that has a pointer to the `changelog` document in the list.

```
Welcome to |project|'s documentation!
======================================

.. include:: ../README.rst
   :end-before: DO-NOT-REMOVE-docs-badges-END

.. include:: ../README.rst
   :start-after: DO-NOT-REMOVE-docs-intro-START

.. toctree::
   :maxdepth: 2
   :caption: Contents:

   changelog
```

docs/requirements.in

`requirements.in` is a standard `requirements.txt`-type file that only lists dependencies that are directly used by the Sphinx static docs site generator. It may optionally contain the minimum necessary versions of those.

```
furo
setuptools-scm
Sphinx
sphinxcontrib-towncrier
```

docs/requirements.txt

But stating just the direct dependencies without strict version restrictions is not enough for reproducible builds. Since it is important to keep the docs build predictable over time, we use pip-tools to generate a `constraints.txt`-type pip-compatible lockfile with pinned version constraints for the whole transitive dependency tree. This file is `requirements.txt` and using it will ensure that the virtualenv for building the docs always has the same software with the same versions in it.

Tip: As a bonus, having a `.in` + `.txt` pair of files is natively supported by GitHub Dependabot.

2.1.4 .readthedocs.yml

To set up Read the Docs, add a `.readthedocs.yml` file in the project root. The following configuration makes sure that the lockfile is used to provision the build env. It also configures how Sphinx should behave like failing the build on any warnings and having nice URLs.

```
---
version: 2

formats: all

sphinx:
  builder: dirhtml
  configuration: docs/conf.py
  fail_on_warning: true

build:
  image: latest

python:
  version: 3.8
  install:
  - requirements: docs/requirements.txt
...
```

Note: When you have a Read the Docs YAML config in your repository, none of the settings supported by it are derived from the web UI.

Tip: Having Read the Docs plugged into your project it is also possible to enable pull-request builds.

2.1.5 CHANGELOG.rst

This file in the project root contains the compiled changelog with the notes from the released project versions. It is managed by Towncrier and should not be edited by you manually.

```
.. towncrier release notes start
```

2.1.6 pyproject.toml

`pyproject.toml` in the root contains the setup for Towncrier itself under the `[tool.towncrier]` section. It binds it all together pointing at the directory for the change notes, the target changelog document and the template to use when generating it. It also lists the categories for the change fragments.

```
[tool.towncrier]
  directory = "docs/changelog.d/"
  filename = "CHANGELOG.rst"
  issue_format = ":issue:`{issue}`"
  package_dir = "src"
  template = "docs/changelog.d/.towncrier-template.rst.j2"
  title_format = "v{version} ({project_date})"
  underlines = ["=", "^", "-", "~"]

  [[tool.towncrier.section]]
    path = ""

  [[tool.towncrier.type]]
    directory = "bugfix"
    name = "Bugfixes"
    showcontent = true

  [[tool.towncrier.type]]
    directory = "feature"
    name = "Features"
    showcontent = true

  [[tool.towncrier.type]]
    directory = "deprecation"
    name = "Deprecations (removal in next major release)"
    showcontent = true

  [[tool.towncrier.type]]
    directory = "breaking"
    name = "Backward incompatible changes"
    showcontent = true

  [[tool.towncrier.type]]
    directory = "doc"
```

(continues on next page)

(continued from previous page)

```
    name = "Documentation"
    showcontent = true

  [[tool.towncrier.type]]
    directory = "misc"
    name = "Miscellaneous"
    showcontent = true
```

2.1.7 README.rst

The README document is an important bit of your project. It shows up on GitHub and is normally shown on PyPI. Besides that, it's possible to include its fragments into the docs front page.

The example below shows how to use comment markers to include a part of the badges into a Sphinx document also embedding some prose from the README. Scroll up and see how it's being embedded into `docs/index.rst`.

```
.. image:: https://img.shields.io/pypi/v/your-project.svg?logo=Python&logoColor=white
   :target: https://pypi.org/project/your-project
   :alt: your-project @ PyPI

.. image:: https://github.com/your-org/your-project/actions/workflows/tox-tests.yaml/
↪badge.svg?event=push
   :target: https://github.com/your-org/your-project/actions/workflows/tox-tests.yaml
   :alt: GitHub Actions CI/CD build status

.. DO-NOT-REMOVE-docs-badges-END

.. image:: https://img.shields.io/readthedocs/your-project/latest.svg?logo=Read%20The
↪%20Docs&logoColor=white
   :target: https://your-project.rtfd.io/en/latest/?badge=latest
   :alt: Documentation Status @ RTD

your-project
============

.. DO-NOT-REMOVE-docs-intro-START

A project with Sphinx-managed documentation and description sourced
from this README.
```

2.1.8 tox.ini

This is an example of setting up a tox-based Sphinx invocation

```
[tox]
envlist = python
isolated_build = true
minversion = 3.21.0
```

(continues on next page)

(continued from previous page)

```
[testenv:docs]
basepython = python3
deps =
  -r{toxinidir}{/}docs{/}requirements.txt
description = Build The Docs
commands =
  # Retrieve possibly missing commits:
  -git fetch --unshallow
  -git fetch --tags

  # Build the html docs with Sphinx:
  {envpython} -m sphinx \
    -j auto \
    -b html \
    {tty:--color} \
    -a \
    -n \
    -W --keep-going \
    -d "{temp_dir}{/}.doctrees" \
    {posargs:} \
    . \
    "{envdir}{/}docs_out"

  # Print out the output docs dir and a way to serve html:
  -{envpython} -c\
  'import pathlib;\
  docs_dir = pathlib.Path(r"{envdir}") / "docs_out";\
  index_file = docs_dir / "index.html";\
  print("\n" + "=" * 120 +\
  f"\n\nDocumentation available under:\n\n\
  \tfile://\{index_file\}\n\nTo serve docs, use\n\n\
  \t$ python3 -m http.server --directory \
  \N\{QUOTATION MARK\}\{docs_dir\}\N\{QUOTATION MARK\} 0\n\n" +\
  "=" * 120)'
changedir = {toxinidir}{/}docs
isolated_build = true
passenv =
  SSH_AUTH_SOCK
skip_install = true
whitelist_externals =
  git
```

With this setup, run `tox -e docs` to build the site locally. Integrate the same command in your CI.

2.2 Contributing to sphinxcontrib-towncrier

Attention: sphinxcontrib-towncrier project exists solely to allow embedding the unreleased changelog fragments that are prepared for the Towncrier tool into Sphinx-based docs sites. At the moment we don't accept any contributions, nor feature requests that are unrelated to this goal.

But if you want to contribute a bug fix or send a pull-request improving our CI, testing and packaging, we will gladly review it.

In order to contribute, you'll need to:

1. Fork the repository.
2. Create a branch, push your changes there.
3. Send it to us as a PR.
4. Iterate on your PR, incorporating the requested improvements and participating in the discussions.

2.3 sphinxcontrib

2.3.1 sphinxcontrib namespace

Subpackages

sphinxcontrib.towncrier package

Submodules

sphinxcontrib.towncrier._towncrier module

Towncrier related shims.

sphinxcontrib.towncrier._towncrier.**get_towncrier_config**(*project_path: pathlib.Path*, *final_config_path: pathlib.Path*) → Dict[str, Any]

Return the towncrier config dictionary.

sphinxcontrib.towncrier._version module

Version definition.

Module contents

Sphinx extension for injecting an unreleased changelog into docs.

class sphinxcontrib.towncrier.**TowncrierDraftEntriesDirective**(*name*, *arguments*, *options*, *content*, *lineno*, *content_offset*, *block_text*, *state*, *state_machine*)

Bases: sphinx.util.docutils.SphinxDirective

Definition of the `towncrier-draft-entries` directive.

has_content = True
May the directive have content?

run() → List[docutils.nodes.Node]
Generate a node tree in place of the directive.

class sphinxcontrib.towncrier.**TowncrierDraftEntriesEnvironmentCollector**

Bases: sphinx.environment.collectors.EnvironmentCollector

Environment collector for `TowncrierDraftEntriesDirective`.

When *TowncrierDraftEntriesDirective* is used in a document, it depends on some dynamically generated change fragments. After the first render, the doctree nodes are put in cache and are reused from there. There's a way to make Sphinx aware of the directive dependencies by calling `BuildEnvironment.note_dependency` but this will only work for fragments that have existed at the time of that first directive invocation.

In order to track newly appearing change fragment dependencies, we need to do so at the time of Sphinx identifying what documents require rebuilding. There's `env-get-outdated` that allows to extend this list of planned rebuilds and we could use it by assigning a document-to-fragments map from within the directive and reading it in the event handler later (since env contents are preserved in cache). But this approach does not take into account cleanups and parallel runs of Sphinx. In order to make it truly parallelism-compatible, we need to define how to merge our custom cache attribute collected within multiple Sphinx subprocesses into one object and that's where `EnvironmentCollector` comes into play.

Refs: * https://github.com/sphinx-doc/sphinx/issues/8040#issuecomment-671587308 * https://github.com/sphinx-contrib/sphinxcontrib-towncrier/issues/1

clear_doc(*app: sphinx.application.Sphinx*, *env: sphinx.environment.BuildEnvironment*, *docname: str*) → None
Clean up env metadata related to the removed document.

This is a handler for `env-purge-doc`.

get_outdated_docs(*app: sphinx.application.Sphinx*, *env: sphinx.environment.BuildEnvironment*, *added: Set[str]*, *changed: Set[str]*, *removed: Set[str]*) → List[str]
Mark docs with changed fragment deps for rebuild.

This is a handler for `env-get-outdated`.

merge_other(*app: sphinx.application.Sphinx*, *env: sphinx.environment.BuildEnvironment*, *docnames: Set[str]*, *other: sphinx.environment.BuildEnvironment*) → None
Merge doc-to-fragments from another proc into this env.

This is a handler for `env-merge-info`.

process_doc(*app: sphinx.application.Sphinx*, *doctree: docutils.nodes.document*) → None
React to `doctree-read` with no-op.

sphinxcontrib.towncrier.**_get_changelog_draft_entries**(*target_version: str*, *allow_empty: bool = False*, *working_dir: str = None*, *config_path: str = None*) → str

Retrieve the unreleased changelog entries from Towncrier.

sphinxcontrib.towncrier.**_get_draft_version_fallback**(*strategy: str*, *sphinx_config: sphinx.config.Config*) → str

Generate a fallback version string for towncrier draft.

sphinxcontrib.towncrier.**_lookup_towncrier_fragments**(*working_dir: str = None*, *config_path: str = None*) → Set[pathlib.Path]

Emit RST-formatted Towncrier changelog fragment paths.

sphinxcontrib.towncrier.**_nodes_from_rst**(*state: docutils.statemachine.State*, *rst_source: str*) → List[docutils.nodes.Node]

Turn an RST string into a list of nodes.

These nodes can be used in the document.

sphinxcontrib.towncrier.**setup**(*app: sphinx.application.Sphinx*) → Dict[str, Union[bool, str]]

Initialize the extension.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

Symbols

C

G

H

M

P

R

S

T