sphinx-pyproject

Release 0.3.0

Move some of your Sphinx configuration into pyproject.toml

Dominic Davis-Foster

Apr 24, 2024

Contents

Chapter

ONE

Installation

1.1 from PyPI

```
$ python3 -m pip install sphinx-pyproject --user
```

1.2 from Anaconda

First add the required channels

```
$ conda config --add channels https://conda.anaconda.org/conda-forge
$ conda config --add channels https://conda.anaconda.org/domdfcoding
```

Then install

```
$ conda install sphinx-pyproject
```

1.3 from GitHub

```
$ python3 -m pip install git+https://github.com/sphinx-toolbox/sphinx-pyproject@master --user
```

Chapter

TWO

Usage

The *SphinxConfig* class will load the configuration from `pyproject.toml`. By passing `globalns=globals()` to the class constructor, the keys parsed from the `pyproject.toml` file will be added to the global namespace of the `conf.py` file.

For example:

```
# conf.py

from sphinx_pyproject import SphinxConfig

config = SphinxConfig("../pyproject.toml", globalns=globals())

author  # This name *looks* to be undefined, but it isn't.
```

The *SphinxConfig* class also provides a `collections.abc.Mapping` interface. If you are going to override or modify one of the configuration values after parsing it, the recommended approach is to explicitly assign the name:

```
extensions = config["extensions"]
extensions.append("sphinx.ext.autodoc")
```

This will prevent warnings from linters etc., but is not necessary for Sphinx to see the configuration.

Note: At time of writing the "Poetry" tool does not support PEP 621. To enable a mode compatible with the `[tool.poetry]` table supply the argument `style="poetry"`. For example:

```
config = SphinxConfig("../pyproject.toml", style="poetry")
```

Additionally the `SphinxConfig` class takes an optional parameter `config_overrides` that can be used to dynamically update values from `pyproject.toml`. This can be helpful for setting dynamic values like `version`.

```
# conf.py
from sphinx_pyproject import SphinxConfig

from myproject import __version__ as myproject_version

config = SphinxConfig("../pyproject.toml", globalns=globals(), config_overrides = {
↪"version": myproject_version})
```

2.1 Configuration

`sphinx-pyproject` parses the configuration from the `[project]` and `[tool.sphinx-pyproject]` tables in `pyproject.toml`. The `[project]` table is defined in **PEP 621**. `sphinx-pyproject` only uses the following keys:

- name – The name of the project.
- version – The version of the project.
- description – The summary description of the project.
- One of authors/maintainers.

The remaining Sphinx configuration values can be provided in the `[tool.sphinx-pyproject]` table.

See this project's pyproject.toml file for an example of this configuration.

Chapter

THREE

API Reference

Move some of your Sphinx configuration into `pyproject.toml`.

Classes:

SphinxConfig([pyproject_file, globalns, ...])	Read the Sphinx configuration from `pyproject.toml`.
ProjectParser()	Parser for **PEP 621** metadata from `pyproject.toml`.
PoetryProjectParser()	Parser for poetry metadata from `pyproject.toml`.

class SphinxConfig(*pyproject_file='../pyproject.toml'*, ***, *globalns=None*, *style='pep621'*, *config_overrides=None*)

Bases: `Mapping`[`str`, `Any`]

Read the Sphinx configuration from `pyproject.toml`.

Parameters

- **pyproject_file** (`Union`[`str`, `Path`, `PathLike`]) – The path to the `pyproject.toml` file. Default `'../pyproject.toml'`.
- **globalns** (`Optional`[`MutableMapping`]) – The global namespace of the `conf.py` file. The variables parsed from the `[tool.sphinx-pyproject]` table will be added to this namespace. By default, or if explicitly `None`, this does not happen.
- **style** (`str`) – Either `pep621` (default), or `poetry` to read configuration from the `[tool.poetry]` table.
- **config_overrides** (`Optional`[`MutableMapping`]) – Custom configuration overrides. This parameter can be used to dynamically update values from `pyproject.toml`. This can be used to patch dynamic values like `version`. By default, or if explicitly `None`, no config updates are performed.

Changed in version 0.2.0: Added the `style` keyword argument.

Changed in version 0.3.0: Added the `config_overrides` keyword argument.

Attributes:

name	The value of the project.name key in the **PEP 621** metadata.
version	The value of the project.version key in the **PEP 621** metadata.
description	The value of the project.description key in the **PEP 621** metadata.
author	A string giving the names of the authors.

Methods:

__getitem__(item)	Returns the value of the given key in the `tool.sphinx-pyproject` table.
__len__()	Returns the number of keys in the `tool.sphinx-pyproject` table.
__iter__()	Returns an iterator over the keys in the `tool.sphinx-pyproject` table.

name
Type: str

The value of the project.name key in the **PEP 621** metadata.

Underscores are replaced by dashes but **PEP 508** normalization is *not* applied.

The recommendation is to assign this to the project variable in `conf.py`:

```
from sphinx_pyproject import SphinxConfig

config = SphinxConfig()
project = config.name
```

version
Type: str

The value of the project.version key in the **PEP 621** metadata.

Converted to a string if the value was a number in the `pyproject.toml` file.

description
Type: str

The value of the project.description key in the **PEP 621** metadata.

author
Type: str

A string giving the names of the authors.

This is parsed from the project.authors key in the **PEP 621** metadata, or the project.maintainers key as a fallback.

The names are joined together, e.g.:

```
# pyproject.toml

[[project.authors]]
name = "Dominic Davis-Foster"

[[project.authors]]
name = "Joe Bloggs"

[[project.authors]]
name = "Jane Doe"
```

```
>>> SphinxConfig("pyproject.toml").author
'Dominic Davis-Foster, Joe Bloggs and Jane Doe'
```

__getitem__(*item*)
Returns the value of the given key in the `tool.sphinx-pyproject` table.

Parameters **item** (str)

Return type Any

__len__()
Returns the number of keys in the `tool.sphinx-pyproject` table.

Return type int

__iter__()
Returns an iterator over the keys in the `tool.sphinx-pyproject` table.

Return type Iterator[str]

class ProjectParser
Bases: AbstractConfigParser

Parser for **PEP 621** metadata from `pyproject.toml`.

Methods:

get_namespace(filename, config)	Returns the `[project]` table in a `project.toml` file.
parse_name(config)	Parse the name key.
parse_version(config)	Parse the version key.
parse_description(config)	Parse the description key.
parse_author(config)	Parse the authors/maintainers key.
parse(config[, set_defaults])	Parse the TOML configuration.

static get_namespace(*filename*, *config*)
Returns the `[project]` table in a `project.toml` file.

Parameters

- **filename** (PathPlus) – The filename the TOML data was read from. Used in error messages.
- **config** (Dict[str, Any]) – The data from the TOML file.

New in version 0.2.0.

Return type Dict[str, Any]

parse_name(*config*)
Parse the name key.

Parameters **config** (Dict[str, Any]) – The unparsed TOML config for the `[project]` table.

Return type str

parse_version(*config*)
Parse the version key.

Parameters **config** (Dict[str, Any]) – The unparsed TOML config for the `[project]` table.

Return type str

parse_description(*config*)
Parse the description key.

Parameters **config** (Dict[str, Any]) – The unparsed TOML config for the [project] table.

Return type str

static parse_author(*config*)
Parse the authors/maintainers key.

Parameters **config** (Dict[str, Any]) – The unparsed TOML config for the [project] table.

Return type str

parse(*config*, *set_defaults=False*)
Parse the TOML configuration.

Parameters

- **config** (Dict[str, Any])
- **set_defaults** (bool) – Has no effect in this class. Default False.

Return type Dict[str, Any]

class PoetryProjectParser
Bases: *ProjectParser*

Parser for poetry metadata from pyproject.toml.

New in version 0.2.0.

Methods:

get_namespace(filename, config)	Returns the [tool.poetry] table in a project.toml file.
parse_author(config)	Parse poetry's authors key.

static get_namespace(*filename*, *config*)
Returns the [tool.poetry] table in a project.toml file.

Parameters

- **filename** (PathPlus) – The filename the TOML data was read from. Used in error messages.
- **config** (Dict[str, Any]) – The data from the TOML file.

Return type Dict[str, Any]

static parse_author(*config*)
Parse poetry's authors key.

Parameters **config** (Dict[str, Any]) – The unparsed TOML config for the [tool.poetry] table.

Return type str

Chapter

FOUR

Downloading source code

The `sphinx-pyproject` source code is available on GitHub, and can be accessed from the following URL: https://github.com/sphinx-toolbox/sphinx-pyproject

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/sphinx-toolbox/sphinx-pyproject
```

```
Cloning into 'sphinx-pyproject'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a 'zip' file by clicking:
Clone or download –> Download Zip

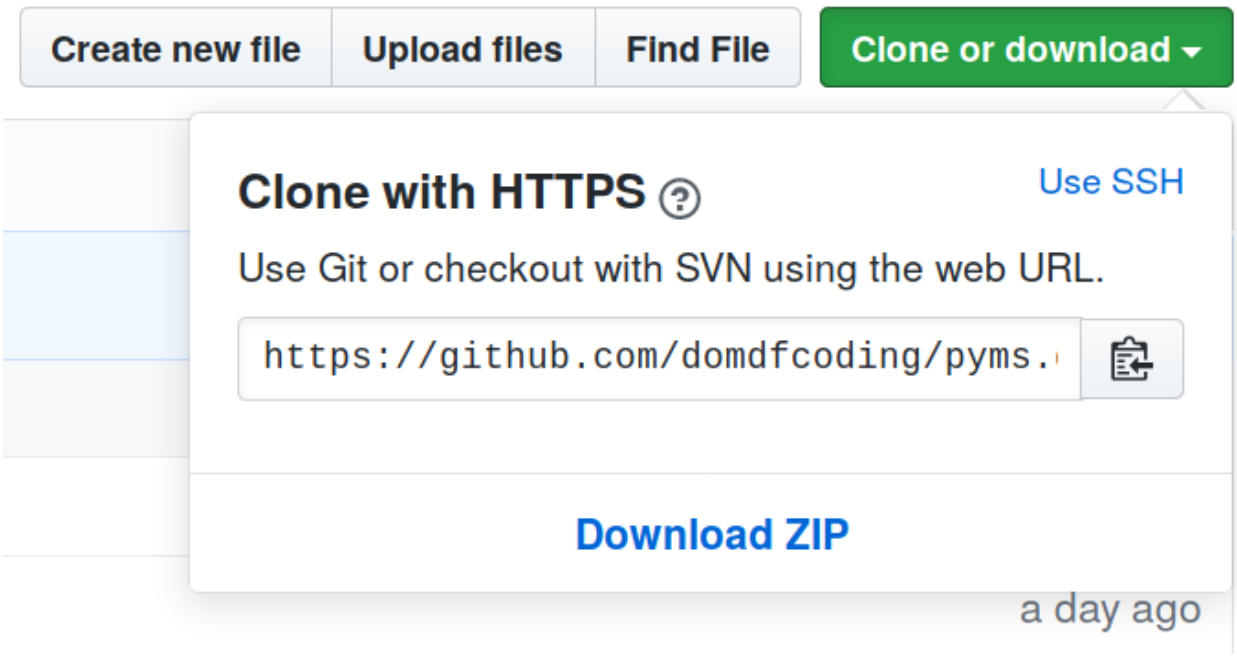

Fig. 1: Downloading a 'zip' file of the source code

4.1 Building from source

The recommended way to build `sphinx-pyproject` is to use tox:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use pep517.build or another **PEP 517**-compatible build tool.

Chapter

FIVE

License

`sphinx-pyproject` is licensed under the MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Private use – The licensed material may be used and modified in private.

Conditions

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.

Limitations

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

See more information on choosealicense.com ⇒


```
Copyright (c) 2021 Dominic Davis-Foster

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
OR OTHER DEALINGS IN THE SOFTWARE.
```

Python Module Index

S

Index