seed-intersphinx-mapping

Release 1.2.2

Populate the Sphinx 'intersphinx_mapping' dictionary from the project's requirements.

Dominic Davis-Foster

Jul 21, 2023

Contents

Chapter

ONE

Overview

This avoids having to manually compile (and keep updated) a mapping like:

```
intersphinx_mapping = {
        "attrs": ("https://www.attrs.org/en/stable/", None),
        "Flask": ("https://flask.palletsprojects.com/en/1.1.x/", None),
        "matplotlib": ("https://matplotlib.org/stable/", None),
        "numpy": ("https://numpy.org/doc/stable/", None),
        "pandas": ("https://pandas.pydata.org/docs/", None),
        "Pyramid": ("https://docs.pylonsproject.org/projects/pyramid/en/latest/",␣
↪None),
        "scikit-learn": ("https://scikit-learn.org/stable/", None),
        "scipy": ("https://docs.scipy.org/doc/scipy/reference/", None),
        "Sphinx": ("https://www.sphinx-doc.org/en/stable/", None),
        }
# Source: https://gist.github.com/bskinn/0e164963428d4b51017cebdb6cda5209
```

Note: Not all projects include a link to their documentation in the Project-URL field of Python's core metadata. Why not submit a pull request to them to include it?

For setuptools' `setup.cfg`, this would look like:

```
project_urls =
    Documentation = <documentation_url, e.g. https://domdf-python-tools.readthedocs.
↪io/en/latest>
```

Or, in pyproject.toml:

```
[project.urls]
Documentation = "<documentation_url, e.g. https://domdf-python-tools.readthedocs.io/
↪en/latest>"
```

In the meantime you will still need to manually include an entry for that project in your `intersphinx_mapping`.

See also:

The Sphinx documentation for `sphinx.ext.intersphinx`.

Chapter

TWO

Usage

2.1 Installation

2.1.1 from PyPI

```
$ python3 -m pip install seed_intersphinx_mapping --user
```

2.1.2 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 seed_intersphinx_mapping
```

2.1.3 from GitHub

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

Enable `seed_intersphinx_mapping` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'seed_intersphinx_mapping',
    ]
```

For more information see
https://www.sphinx-doc.org/en/master/usage/extensions#third-party-extensions .

2.2 Configuration

pkg_requirements_source
The requirements source. This may be one of:

- A list of directories (relative to *repository_root*) in which to search for `requirements.txt` files. Any files found will be used to compile the list of requirements.
- The string `'requirements'`. The list of requirements will be determined from the `requirements.txt` file in the directory given by the *repository_root* option.
- The string `'pyproject'` (or `'pyproject.toml'`). The list will be parsed from the `[project.dependencies]` table of the `pyproject.toml` file in the *repository_root*.

  See also:

  PEP 621 – Storing project metadata in pyproject.toml

- The string `'flit'`. The list will be parsed from the `[tool.flit.metadata.requires]` table of the `pyproject.toml` file in the *repository_root*.

repository_root
The path to the repository root, relative to the Sphinx source directory.

E.g., for this repository structure:

```
.
├── LICENSE
├── README.rst
├── doc-source  # <- this is the Sphinx source directory
│   ├── index.rst
│   └── conf.py
├── requirements.txt  # <- this is the file containing the requirements
├── seed_intersphinx_mapping
│   └── __init__.py
├── setup.py
├── tests
└── tox.ini
```

the value would be `'..'`, which is the default.

2.3 Caching

`seed_intersphinx_mapping` caches the documentation URLs for PyPI packages. The cache can be cleared as follows:

```
$ python3 -m seed_intersphinx_mapping
```

Chapter

THREE

Public API

In addition to the Sphinx extension `seed_intersphinx_mapping` provides a public API.

3.1 seed_intersphinx_mapping

Populate the Sphinx 'intersphinx_mapping' dictionary from the project's requirements.

Changed in version 0.5.0: The functions formerly in `seed_intersphinx_mapping.core` can now be found here.

Functions:

fallback_mapping()	Returns the fallback mapping for projects that do not provide a link to their documentation on PyPI.
get_sphinx_doc_url(pypi_name)	Returns the URL to the given project's Sphinx documentation.
seed_intersphinx_mapping(*requirements)	Returns an intersphinx mapping dictionary for the project's requirements.

fallback_mapping()

Returns the fallback mapping for projects that do not provide a link to their documentation on PyPI.

The mapping is loaded from JSON data on demand, and consists of `project_name: url` pairs.

Return type `Dict[str, str]`

get_sphinx_doc_url(*pypi_name*)

Returns the URL to the given project's Sphinx documentation.

Not all projects include this URL in their distributions, and therefore it may not be possible to determine it from PyPI.

Responses are cached to prevent overloading the PyPI server. The cache can be cleared as follows:

```
$ python3 -m seed_intersphinx_mapping
```

Parameters **pypi_name** (`str`) – The name of the project on PyPI

Return type `str`

Returns The URL of the project's Sphinx documentation.

Raises

- `ValueError` if the url could not be determined.

- packaging.requirements.InvalidRequirement if the project could not be found on PyPI.

Changed in version 0.4.0: Now raises InvalidRequirement rather than apeye.slumber_url.exceptions.HttpNotFoundError if the project could not be found on PyPI.

seed_intersphinx_mapping(**requirements*)

Returns an intersphinx mapping dictionary for the project's requirements.

Parameters ***requirements** (Union[Requirement, str]) – The requirements to find the documentation for.

Changed in version 0.4.0: Now takes the requirements as arguments rather than a directory to read the `requirements.txt` file from.

Return type Dict[str, Tuple[str, Optional[str]]]

3.2 seed_intersphinx_mapping.extension

Sphinx-specific functionality.

Functions:

setup(app)	Setup *seed_intersphinx_mapping*.
sphinx_seed_intersphinx_mapping(app, config)	Updates the `intersphinx_mapping` dictionary in the sphinx configuration.

setup(*app*)

Setup *seed_intersphinx_mapping*.

Parameters **app** (Sphinx)

Return type Dict[str, Any]

sphinx_seed_intersphinx_mapping(*app*, *config*)

Updates the `intersphinx_mapping` dictionary in the sphinx configuration. to include the documentation for the project's requirements.

pkg_requirements_source may be one of:

- A list of directories (relative to *repository_root*) in which to search for `requirements.txt` files. Any files found will be used to compile the list of requirements.
- The string `'requirements'`. The list of requirements will be determined from the `requirements.txt` file in the directory given by the *repository_root* option.
- The string `'pyproject'` (or `'pyproject.toml'`). The list will be parsed from the `[project.dependencies]` table of the `pyproject.toml` file in the *repository_root*.

  See also:

  PEP 621 – Storing project metadata in pyproject.toml

- The string `'flit'`. The list will be parsed from the `[tool.flit.metadata.requires]` table of the `pyproject.toml` file in the *repository_root*.

Parameters

- **app** (Sphinx)
- **config** (Config)

3.3 seed_intersphinx_mapping.requirements_parsers

Contains functions for parsing requirements.

Functions:

parse_flit_requirements(base_dir)	Returns a list of package names listed as requirements in the `[tool.flit]` section of `pyproject.toml`.
parse_pyproject_toml(base_dir)	Returns a list of package names listed as requirements in the `pyproject.toml` file.
parse_requirements_txt(base_dir)	Returns a list of package names listed as requirements in the `requirements.txt` file.

parse_flit_requirements(*base_dir*)

Returns a list of package names listed as requirements in the `[tool.flit]` section of `pyproject.toml`.

New in version 0.4.0.

Parameters **base_dir** (Union[str, Path, PathLike]) – The directory in which to find the `pyproject.toml` file.

Return type List[str]

parse_pyproject_toml(*base_dir*)

Returns a list of package names listed as requirements in the `pyproject.toml` file.

New in version 0.4.0.

Parameters **base_dir** (Union[str, Path, PathLike]) – The directory in which to find the `pyproject.toml` file.

Return type List[str]

parse_requirements_txt(*base_dir*)

Returns a list of package names listed as requirements in the `requirements.txt` file.

Parameters **base_dir** (Union[str, Path, PathLike]) – The directory in which to find the `requirements.txt` file.

Return type List[str]

Chapter

FOUR

Downloading source code

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

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

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

```
Cloning into 'seed_intersphinx_mapping'...
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

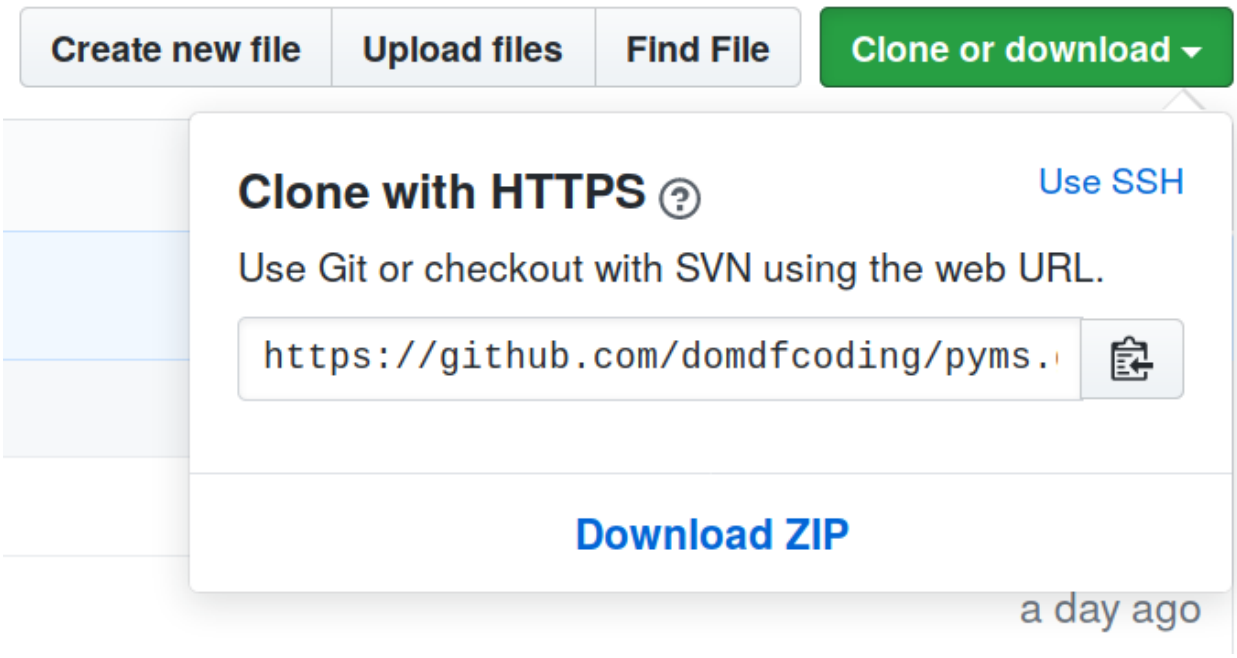

Fig. 1: Downloading a 'zip' file of the source code

4.1 Building from source

The recommended way to build `seed_intersphinx_mapping` 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

`seed_intersphinx_mapping` 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 ⇒

Python Module Index

S

Index