
sphinx-toolbox

Release 2.14.1

Box of handy tools for Sphinx

Dominic Davis-Foster

Sep 28, 2021

Contents

1	Installation	1
1.1	from PyPI	1
1.2	from Anaconda	1
1.3	from GitHub	1
I	Extensions	3
2	Overview	5
3	assets	7
3.1	Usage	7
3.2	Configuration	8
3.3	API Reference	8
4	changeset	11
4.1	Usage	11
4.2	API Reference	12
4.3	Examples	12
5	code	15
5.1	Usage	15
5.2	API Reference	17
6	collapse	21
6.1	Usage	21
6.2	API Reference	22
7	confval	25
7.1	Usage	25
7.2	API Reference	26
8	decorators	29
8.1	Usage	29
8.2	API Reference	30
9	documentation_summary	31
9.1	Configuration	31
9.2	Usage	31
9.3	API Reference	32
10	flake8	33
10.1	Usage	33

10.2	API Reference	33
11	formatting	35
11.1	Usage	35
11.2	API Reference	36
12	github	39
12.1	Configuration	39
12.2	Usage	39
12.3	Caching	41
12.4	API Reference	41
12.5	github.issues submodule	42
12.6	github.repos_and_users submodule	45
13	installation	49
13.1	Configuration	49
13.2	Usage	49
13.3	API Reference	52
13.4	ExtensionsDirective	52
13.5	InstallationDirective	52
13.6	Sources	53
13.7	conda_installation	54
13.8	copy_asset_files	54
13.9	github_installation	54
13.10	make_installation_instructions	55
13.11	pypi_installation	55
13.12	setup	55
13.13	sources	55
14	issues	57
14.1	Usage	57
14.2	Caching	58
14.3	API Reference	58
15	latex	59
15.1	Example Footnotes	59
15.2	Usage	60
15.3	API Reference	60
16	pre_commit	65
16.1	Usage	65
16.2	API Reference	66
17	rest_example	69
17.1	Usage	69
17.2	API Reference	70
18	shields	71
18.1	Usage	71
18.2	API Reference	75
19	sidebar_links	77
19.1	Usage	77
19.2	API Reference	78

20	source	79
20.1	Usage	79
20.2	API Reference	80
21	wikipedia	81
21.1	Configuration	81
21.2	Usage	81
21.3	API Reference	82
22	more_autodoc	83
22.1	more_autodoc.augment_defaults	83
22.2	more_autodoc.autonamedtuple	84
22.3	more_autodoc.autoprotocol	88
22.4	more_autodoc.autotypeddict	93
22.5	more_autodoc.generic_bases	98
22.6	more_autodoc.genericalias	99
22.7	more_autodoc.no_docstring	100
22.8	more_autodoc.overloads	101
22.9	more_autodoc.regex	105
22.10	more_autodoc.sourcelink	110
22.11	more_autodoc.typehints	111
22.12	more_autodoc.typevars	117
22.13	more_autodoc.variables	120
23	more_autosummary	125
23.1	Configuration	125
23.2	API Reference	127
24	tweaks	131
24.1	tweaks.footnote_symbols	131
24.2	tweaks.latex_layout	132
24.3	tweaks.latex_toc	132
24.4	tweaks.param_dash	134
24.5	tweaks.sphinx_panels_tabs	135
24.6	tweaks.tabsize	137
25	Changelog	139
25.1	2.14.1	139
25.2	2.14.0	139
25.3	2.13.0	139
II	Developer API	141
26	sphinx_toolbox	143
26.1	setup	143
27	sphinx_toolbox.config	145
27.1	InvalidOptionError	145
27.2	MissingOptionError	145
27.3	ToolboxConfig	145
27.4	validate_config	147
28	sphinx_toolbox.testing	149
28.1	Sphinx	150

28.2	run_setup	153
28.3	RunSetupOutput	154
28.4	remove_html_footer	154
28.5	check_html_regression	154
28.6	remove_html_link_tags	154
28.7	check_asset_copy	155
28.8	HTMLRegressionFixture	155
28.9	html_regression	155
29	sphinx_toolbox.utils	157
29.1	add_nbsp_substitution	158
29.2	allow_subclass_add	158
29.3	baseclass_is_private	158
29.4	begin_generate	158
29.5	code_repr	159
29.6	escape_trailing__	159
29.7	filter_members_warning	159
29.8	flag	159
29.9	get_first_matching	159
29.10	GITHUB_COM	159
29.11	is_namedtuple	160
29.12	make_github_url	160
29.13	NoMatchError	160
29.14	OptionSpec	160
29.15	Param	160
29.16	parse_parameters	160
29.17	Purger	161
29.18	SetupFunc	162
29.19	SphinxExtMetadata	162
29.20	typed_flag_regex	162
29.21	typed_param_regex	163
29.22	unknown_module_warning	163
29.23	untyped_param_regex	163
	Python Module Index	165
	Index	167

Installation

1.1 from PyPI

```
$ python3 -m pip install sphinx-toolbox --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-toolbox
```

1.3 from GitHub

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


Part I

Extensions

Overview

Enable `sphinx-toolbox` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx.ext.viewcode',  
    'sphinx_tabs.tabs',  
    'sphinx-prompt',  
    'sphinx_toolbox',  
]
```

The following features are enabled by default:

- `sphinx_toolbox.assets`
- `sphinx_toolbox.changeset`
- `sphinx_toolbox.code`
- `sphinx_toolbox.collapse`
- `sphinx_toolbox.confval`
- `sphinx_toolbox.decorators`
- `sphinx_toolbox.formatting`
- `sphinx_toolbox.github`
- `sphinx_toolbox.installation`
- `sphinx_toolbox.issues`
- `sphinx_toolbox.latex`
- `sphinx_toolbox.rest_example`
- `sphinx_toolbox.shields` *
- `sphinx_toolbox.sidebar_links`
- `sphinx_toolbox.source`
- `sphinx_toolbox.wikipedia`
- `sphinx_toolbox.more_autodoc.autonamedtuple`
- `sphinx_toolbox.more_autodoc.autoprotocol`
- `sphinx_toolbox.more_autodoc.autotypeddict`

* Indicates that the extension cannot be enabled separately from the rest of `sphinx_toolbox`.

Some features must be enabled separately:

- `sphinx_toolbox.more_autodoc`
 - `sphinx_toolbox.more_autodoc.augment_defaults`
 - `sphinx_toolbox.more_autodoc.generic_bases`
 - `sphinx_toolbox.more_autodoc.genericalias`
 - `sphinx_toolbox.more_autodoc.no_docstring`
 - `sphinx_toolbox.more_autodoc.overloads`
 - `sphinx_toolbox.more_autodoc.regex`
 - `sphinx_toolbox.more_autodoc.sourcelink`
 - `sphinx_toolbox.more_autodoc.typehints`
 - `sphinx_toolbox.more_autodoc.typevars`
 - `sphinx_toolbox.more_autodoc.variables`

`sphinx_toolbox.more_autodoc` can also be specified as an extension, which enables all of the above features.

- `sphinx_toolbox.more_autosummary`

Provides a patched version of `sphinx.ext.autosummary.Autosummary` to fix an issue where the module name is sometimes duplicated.

I.e. `foo.bar.baz()` became `foo.bar.foo.bar.baz()`, which of course doesn't exist and created a broken link.

assets

Role to provide a link to open a file within the web browser, rather than downloading it.

New in version 0.5.0.

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

```
extensions = [
    ...
    'sphinx_toolbox.assets',
]
```

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

3.1 Usage

:asset:

Adds a link to a local file that can be viewed within the web browser. The file will be copied from the directory set in `assets_dir` to `/_assets` in the HTML output.

This is similar to the `download` role, but that role will download the file to the user's computer instead. This role may be useful for PDFs, which most web browsers can display.

If the file can't be found an error will be shown in the build output:

```
<page where the error occurred>: Asset file '<missing asset file name>' not found.
```

Asset

```
:asset:`hello_world.txt`

:asset:`hello_world <hello_world.txt>`
```

hello_world.txt

hello_world

Download

```
:download:`hello_world.txt <../assets/hello_world.txt>`
```

hello_world.txt

3.2 Configuration

assets_dir

Type: `str`

Required: `False`

Default: `'./assets'`

The directory in which to find assets for the `asset` role.

3.3 API Reference

Functions:

<code>asset_role(name, rawtext, text, lineno, inliner)</code>	Adds a link to an asset.
<code>visit_asset_node(translator, node)</code>	Visit an <code>AssetNode</code> .
<code>depart_asset_node(translator, node)</code>	Depart an <code>AssetNode</code> .
<code>setup(app)</code>	Setup <code>sphinx_toolbox.assets</code> .

Classes:

<code>AssetNode([rawsource, text])</code>	Node representing a link to an asset.
---	---------------------------------------

asset_role(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*=`{}`, *content*=`[]`)

Adds a link to an asset.

Parameters

- **name** (`str`) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (`str`) – A string containing the entire interpreted text input, including the role and markup.
- **text** (`str`) – The interpreted text content.
- **lineno** (`int`) – The line number where the interpreted text begins.
- **inliner** (`Inline`) – The `docutils.parsers.rst.states.Inliner` object that called `source_role()`. It contains the several attributes useful for error reporting and document tree access.
- **options** (`Dict`) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default `{}`.
- **content** (`List[str]`) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default `[]`.

Return type `Tuple[Sequence[AssetNode], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

class AssetNode(*rawsource*="", *text*="", **children*, ***attributes*)

Bases: `reference`

Node representing a link to an asset.

visit_asset_node(*translator, node*)

Visit an *AssetNode*.

Parameters

- **translator** (*HTMLTranslator*)
- **node** (*AssetNode*) – The node being visited.

depart_asset_node(*translator, node*)

Depart an *AssetNode*.

Parameters

- **translator** (*HTMLTranslator*)
- **node** (*AssetNode*) – The node being visited.

setup(*app*)

Setup *sphinx_toolbox.assets*.

New in version 1.0.0.

Parameters **app** (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

changeset

Customised versions of Sphinx’s *versionadded*, *versionchanged* and *deprecated* directives to correctly handle bullet lists.

New in version 2.11.0.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.changeset',  
]
```

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

4.1 Usage

.. versionadded:: version

Documents the version of the project which added the described feature.

The first argument must be given and is the version in question; you can add a second argument consisting of a *brief* explanation of the change. Alternatively, a longer description may be given in the body of the directive.

.. versionchanged:: version

Similar to *versionadded*, but describes when and what changed in the feature in some way (new parameters, changed side effects, etc.).

.. deprecated:: version

Similar to *versionchanged*, but describes when the feature was deprecated. An explanation can also be given, for example to inform the reader what should be used instead.

This extension also adds the following directive:

.. versionremoved:: version [details]

Similar to *versionchanged*, but describes when the feature was or will be removed. An explanation can also be given, for example to inform the reader what should be used instead.

4.2 API Reference

class **VersionChange**(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `VersionChange`

Directive to describe a addition/change/deprecation/removal in a specific version.

setup(*app*)

Setup `sphinx_toolbox.changeset`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

4.3 Examples

```
.. versionadded:: 2.4
.. versionadded:: 2.5 The *spam* parameter.

.. versionadded:: 2.6
   The *parrot* parameter.

.. deprecated:: 3.1
   Use :func:`spam` instead.

.. deprecated:: 3.2 Use :func:`lobster` instead.

.. versionremoved:: 1.2.3 Use :func:`foo` instead.

.. versionremoved:: 1.2.3

   Due to an unfixable bug this function has been removed.
   If you desperately need this functionality please write to the mailing list at
   :email:`python-users@example.org`

.. versionchanged:: 0.3.0

   * Parameters for ``__init__`` can be documented either in the class docstring
     or alongside the attribute.
     The class docstring has priority.
   * Added support for `autodocsumm <https://github.com/Chilipp/autodocsumm>`_.
```

New in version 2.4.

New in version 2.5: The *spam* parameter.

New in version 2.6: The *parrot* parameter.

Deprecated since version 3.1: Use `spam()` instead.

Deprecated since version 3.2: Use `lobster()` instead.

Removed in version 1.2.3: Use `foo()` instead.

Removed in version 1.2.3: Due to an unfixable bug this function has been removed. If you desperately need this functionality please write to the mailing list at python-users@example.org

Changed in version 0.3.0:

- Parameters for `__init__` can be documented either in the class docstring or alongside the attribute. The class docstring has priority.
- Added support for [autodocsumm](#).

code

Customised `.. code-block::` directive with an adjustable indent size.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.code',  
]
```

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

5.1 Usage

`.. code-block::` [language]

`.. sourcecode::` [language]

Customised `.. code-block::` directive with an adjustable indent size.

`:tab-width: width (integer)`

Sets the size of the indentation in spaces.

All other options from `code-block` are available, see the [Sphinx documentation](#) for details.

Examples

```
.. code-block:: python  
  
    def print(text):  
        sys.stdout.write(text)
```

```
def print(text):  
    sys.stdout.write(text)
```

```
.. code-block:: python  
    :tab-width: 8  
  
    def print(text):  
        sys.stdout.write(text)
```

```
def print(text):  
    sys.stdout.write(text)
```

```
.. code-cell:: [language]
.. output-cell:: [language]
```

Customised `.. code-block::` directives which display an execution count to the left of the code block, similar to a Jupyter Notebook cell.

New in version 2.6.0.

:execution-count: count (positive integer)

The execution count of the cell.

All other options from the `code-block` directive above are available.

Examples

```
.. code-cell:: python
   :execution-count: 1

   def print(text):
       sys.stdout.write(text)

   print("hello world")

.. output-cell::
   :execution-count: 1

   hello world
```

In [1]:

```
def print(text):
    sys.stdout.write(text)

print("hello world")
```

[1]:

```
hello world
```

```
.. code-cell:: python
   :execution-count: 2
   :tab-width: 8

   def print(text):
       sys.stdout.write(text)
```

In [2]:

```
def print(text):
    sys.stdout.write(text)
```

See also: [nbsphinx](#), which inspired these directives and provides additional functionality for integrating Jupyter Notebooks with Sphinx.

5.2 API Reference

Classes:

<code>CodeBlock</code> (name, arguments, options, content, ...)	Directive for a code block with special highlighting or line numbering settings.
<code>CodeCell</code> (name, arguments, options, content, ...)	Customised code block which displays an execution count to the left of the code block, similar to a Jupyter Notebook cell.
<code>OutputCell</code> (name, arguments, options, ...)	Variant of <code>CodeCell</code> for displaying the output of a cell in a Jupyter Notebook.
<code>Prompt</code> ([rawsource, text])	Represents a cell prompt for a <code>CodeCell</code> and <code>OutputCell</code> .

Functions:

<code>visit_prompt_html</code> (translator, node)	Visit a <code>Prompt</code> node with the HTML translator.
<code>visit_prompt_latex</code> (translator, node)	Visit a <code>Prompt</code> node with the LaTeX translator.
<code>copy_asset_files</code> (app[, exception])	Copy additional stylesheets into the HTML build directory.
<code>configure</code> (app, config)	Configure <code>sphinx_toolbox.code</code> .
<code>setup</code> (app)	Setup <code>sphinx_toolbox.code</code> .

class `CodeBlock`(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)

Bases: `CodeBlock`

Directive for a code block with special highlighting or line numbering settings.

The indent_size can be adjusted with the `:tab-width: <int>` option.

run()

Process the content of the code block.

Return type `List[Node]`

class `CodeCell`(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)

Bases: `CodeBlock`

Customised code block which displays an execution count to the left of the code block, similar to a Jupyter Notebook cell.

The indent_size can be adjusted with the `:tab-width: <int>` option.

The execution count can be set using the `:execution-count: <int>` option.

New in version 2.6.0.

run()

Process the content of the code block.

Return type `List[Node]`

class `OutputCell`(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)
Bases: `CodeCell`

Variant of `CodeCell` for displaying the output of a cell in a Jupyter Notebook.

The `indent_size` can be adjusted with the `:tab-width:` `<int>` option.

The execution count can be set using the `:execution-count:` `<int>` option.

New in version 2.6.0.

class `Prompt`(*rawsource="", text="", *children, **attributes*)
Bases: `General`, `FixedTextElement`

Represents a cell prompt for a `CodeCell` and `OutputCell`.

New in version 2.6.0.

visit_prompt_html(*translator, node*)
Visit a `Prompt` node with the HTML translator.

New in version 2.6.0.

Parameters

- **translator** (`HTMLTranslator`)
- **node** (`Prompt`)

visit_prompt_latex(*translator, node*)
Visit a `Prompt` node with the LaTeX translator.

New in version 2.6.0.

Parameters

- **translator** (`LaTeXTranslator`)
- **node** (`Prompt`)

copy_asset_files(*app, exception=None*)
Copy additional stylesheets into the HTML build directory.

New in version 2.6.0.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **exception** (`Optional[Exception]`) – Any exception which occurred and caused Sphinx to abort. Default `None`.

configure(*app, config*)
Configure `sphinx_toolbox.code`.

New in version 2.9.0.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **config** (`Config`)

setup(*app*)

Setup *sphinx_toolbox.code*.

New in version 1.0.0.

Parameters *app* (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

collapse

Adds a collapsible section to an HTML page using a `details` element.

New in version 2.5.0.

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

```
extensions = [
    ...
    'sphinx_toolbox.collapse',
]
```

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

6.1 Usage

.. collapse:: [label]

Adds a collapsible section to an HTML page using a `details` element.

With non-HTML builders, the content will be added as-is.

```
.. collapse:: Details

    Something small enough to escape casual notice.

.. collapse:: A Different Label
   :class: custom-summary
   :name: summary0

    Something else that might escape notice.

.. collapse:: A long code block

    .. code-block:: python

        print("Not really")
```

Something small enough to escape casual notice.

Something else that might escape notice.

```
print("Not really")
```

6.2 API Reference

Classes:

<code>CollapseDirective</code> (name, arguments, options, ...)	A Sphinx directive to add a collapsible section to an HTML page using a <code>details</code> element.
<code>CollapseNode</code> ([rawsource, label])	Node that represents a collapsible section.

Functions:

<code>visit_collapse_node</code> (translator, node)	Visit a <code>CollapseNode</code> .
<code>depart_collapse_node</code> (translator, node)	Depart a <code>CollapseNode</code> .
<code>setup</code> (app)	Setup <code>sphinx_toolbox.collapse</code> .

class `CollapseDirective`(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)

Bases: `SphinxDirective`

A Sphinx directive to add a collapsible section to an HTML page using a `details` element.

Methods:

<code>run()</code>	Process the content of the directive.
--------------------	---------------------------------------

run()

Process the content of the directive.

Return type `Sequence`[`Node`]

class `CollapseNode`(rawsource="", label=None, *children, **attributes)

Bases: `Body`, `Element`

Node that represents a collapsible section.

Parameters

- **rawsource** (`str`) – Default `' '`.
- **label** (`Optional`[`str`]) – Default `None`.

visit_collapse_node(translator, node)

Visit a `CollapseNode`.

Parameters

- **translator** (`HTMLTranslator`)
- **node** (`CollapseNode`) – The node being visited.

depart_collapse_node(translator, node)

Depart a `CollapseNode`.

Parameters

- **translator** (`HTMLTranslator`)
- **node** (`CollapseNode`) – The node being visited.

setup(*app*)

Setup *sphinx_toolbox.collapse*.

Parameters *app* (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

confval

The confval directive and role for configuration values.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.confval',  
]
```

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

7.1 Usage

.. **confval::** name

Used to document a configuration value.

:type: (string)

Indicates the configuration value's type.

:default: (string)

Indicates the default value.

:required: (flag)

Indicates the whether the configuration value is required.

:noindex: (flag)

Disables the index entry and cross-referencing for this configuration value.

New in version 2.11.0.

:confval:

Role which provides a cross-reference to a *confval* directive.

Examples:

```
.. confval:: demo  
   :type: string  
   :default: ``"Hello World"``  
   :required: False
```

demo

Type: string

Required: False

Default: "Hello World"

To enable this feature set the `:confval:`demo`` configuration value to "True".

To enable this feature set the `demo` configuration value to "True".

7.2 API Reference

Classes:

<code>ConfigurationValue</code> (name, arguments, options, ...)	The confval directive.
---	------------------------

Functions:

<code>register_confval</code> (app[, override])	Create and register the confval role and directive.
<code>setup</code> (app)	Setup <code>sphinx_toolbox.confval</code> .

class `ConfigurationValue`(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)

Bases: `GenericObject`

The confval directive.

Changed in version 1.1.0: The formatting of the type, required and default options can be customised using the `self.format_*` methods.

Changed in version 2.11.0: Added the `:noindex:` option, which disables the index entry and cross-referencing for this configuration value.

Methods:

<code>run()</code>	Process the content of the directive.
<code>format_type</code> (the_type)	Formats the <code>:type:</code> option.
<code>format_required</code> (required)	Formats the <code>:required:</code> option.
<code>format_default</code> (default)	Formats the <code>:default:</code> option.

`run()`

Process the content of the directive.

Return type `List[Node]`

static `format_type`(the_type)

Formats the `:type:` option.

New in version 1.1.0.

Parameters `the_type` (`str`)

Return type `str`

static `format_required`(required)

Formats the `:required:` option.

New in version 1.1.0.

Parameters `required` (`str`)

Return type `bool`

static `format_default(default)`
Formats the `:default:` option.

New in version 1.1.0.

Parameters `default` (`str`)

Return type `str`

register_confval(*app*, *override=False*)
Create and register the `confval` role and directive.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **override** (`bool`) – Default `False`.

setup(*app*)
Setup `sphinx_toolbox.confval`.

New in version 0.7.0.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

decorators

reStructuredText XRef role for decorators.

New in version 0.9.0.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.decorators',  
]
```

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

8.1 Usage

:deco:

Adds a cross reference to a decorator, prefixed with an @.

```
.. decorator:: my_decorator  
  
    A decorator.  
  
:deco: `my_decorator`  
:deco: `@my_decorator`  
:deco: `Title <my_decorator>`
```

@my_decorator
A decorator.
@my_decorator
@my_decorator
Title

8.2 API Reference

Classes:

<i>PyDecoXRefRole</i> ([fix_parens, lowercase, ...])	XRef Role for decorators members.
--	-----------------------------------

Functions:

<i>setup</i> (app)	Setup <i>sphinx_toolbox.decorators</i> .
--------------------	--

class PyDecoXRefRole(*fix_parens=False, lowercase=False, nodeclass=None, innernodeclass=None, warn_dangling=False*)

Bases: PyXRefRole

XRef Role for decorators members.

Methods:

<i>process_link</i> (env, refnode, ...)	Called after parsing title and target text, and creating the reference node (given in refnode).
---	---

process_link(*env, refnode, has_explicit_title, title, target*)

Called after parsing title and target text, and creating the reference node (given in refnode).

This method can alter the reference node and must return a new (or the same) (title, target) tuple.

Parameters

- **env** (*BuildEnvironment*)
- **refnode** (*Element*)
- **has_explicit_title** (*bool*)
- **title** (*str*)
- **target** (*str*)

Return type *Tuple*[*str*, *str*]

setup(*app*)

Setup *sphinx_toolbox.decorators*.

Parameters **app** (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

documentation_summary

Allows insertion of a summary line on the title page generated with the LaTeX builder, and at a custom location throughout the document.

New in version 2.2.0.

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

```
extensions = [
    ...
    'sphinx_toolbox.documentation_summary',
]
```

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

9.1 Configuration

documentation_summary

Type: `str`

The documentation summary to display on the title page with the LaTeX builder, and at the location of *documentation-summary* directives for other builders.

If undefined no summary is shown.

9.2 Usage

.. documentation-summary::

Adds the documentation summary as configured above.

Example

```
.. documentation-summary::
```

:meta:

Include the summary as a *meta* “description” tag in the HTML output.

The structure of the description is `{project} -- {summary}`, where `project` is configured in `conf.py`.

See [the sphinx documentation](#) for more information on the `project` option.

New in version 2.10.0.

9.3 API Reference

Classes:

<code>DocumentationSummaryDirective(name, ...)</code>	A Sphinx directive for creating a summary line.
---	---

Functions:

<code>configure(app, config)</code>	Configure <code>sphinx_toolbox.documentation_summary</code> .
<code>setup(app)</code>	Setup <code>sphinx_toolbox.documentation_summary</code> .

class `DocumentationSummaryDirective`(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

A Sphinx directive for creating a summary line.

Methods:

<code>run()</code>	Process the content of the directive.
--------------------	---------------------------------------

run()

Process the content of the directive.

Return type `List[Node]`

configure(*app, config*)

Configure `sphinx_toolbox.documentation_summary`.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **config** (`Config`)

setup(*app*)

Setup `sphinx_toolbox.documentation_summary`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

flake8

A Sphinx directive for documenting flake8 codes.

New in version 1.6.0.

Enable `sphinx_toolbox.flake8` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.flake8',
]
```

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

10.1 Usage

.. **flake8-codes::** plugin

Adds a table documenting a flake8 plugin's codes.

The directive takes a single argument – the fully qualified name of the flake8 plugin module.

Codes to document are given in the body of the directive.

Example

```
.. flake8-codes:: flake8_dunder_all

DALL000
```

Code	Description
DALL000	Module lacks <code>__all__</code> .

10.2 API Reference

class Flake8CodesDirective(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

A Sphinx directive for documenting flake8 codes.

setup(*app*)

Setup *sphinx_toolbox.flake8*.

Parameters *app* (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

formatting

Directives, roles and nodes for text formatting.

New in version 0.2.0.

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

```
extensions = [
    ...
    'sphinx_toolbox.formatting',
]
```

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

11.1 Usage

:iabbr:

An abbreviation. If the role content contains a parenthesized explanation, it will be treated specially: it will be shown in a tool-tip in HTML, and output only once in LaTeX.

Unlike Sphinx's `abbr` role, this one shows the abbreviation in italics.

New in version 0.2.0.

Example

```
:iabbr:`LIFO (last-in, first-out)`
```

LIFO (*last-in, first-out*)

:bold-title:

Role for displaying a pseudo title in bold.

This is useful for breaking up Python docstrings.

New in version 2.12.0.

Example

```
:bold-title: `Examples:`
:bold-title: `Other Extensions`
```

Examples:

Other Extensions

11.2 API Reference

Classes:

<code><i>ItalicAbbreviationNode</i>([rawsource, text])</code>	Docutils Node to show an abbreviation in italics.
<code><i>ItalicAbbreviation</i>()</code>	Docutils role to show an abbreviation in italics.

Functions:

<code><i>visit_iabbr_node</i>(translator, node)</code>	Visit an <i>ItalicAbbreviationNode</i> .
<code><i>depart_iabbr_node</i>(translator, node)</code>	Depart an <i>ItalicAbbreviationNode</i> .
<code><i>latex_visit_iabbr_node</i>(translator, node)</code>	Visit an <i>ItalicAbbreviationNode</i> .
<code><i>latex_depart_iabbr_node</i>(translator, node)</code>	Depart an <i>ItalicAbbreviationNode</i> .
<code><i>setup</i>(app)</code>	Setup <i>sphinx_toolbox.formatting</i> .

class `ItalicAbbreviationNode`(rawsource="", text="", *children, **attributes)

Bases: `abbreviation`

Docutils Node to show an abbreviation in italics.

class `ItalicAbbreviation`

Bases: `Abbreviation`

Docutils role to show an abbreviation in italics.

Methods:

<code><i>run</i>()</code>	Process the content of the italic abbreviation role.
---------------------------	--

run()

Process the content of the italic abbreviation role.

Return type `Tuple[List[Node], List[system_message]]`

visit_iabbr_node(translator, node)

Visit an *ItalicAbbreviationNode*.

Parameters

- **translator** (`HTMLTranslator`)
- **node** (*ItalicAbbreviationNode*) – The node being visited.

depart_iabbr_node(translator, node)

Depart an *ItalicAbbreviationNode*.

Parameters

- **translator** (`HTMLTranslator`)
- **node** (*ItalicAbbreviationNode*) – The node being visited.

latex_visit_iabbr_node(*translator, node*)

Visit an *ItalicAbbreviationNode*.

Parameters

- **translator** (*LaTeXTranslator*)
- **node** (*ItalicAbbreviationNode*) – The node being visited.

latex_depart_iabbr_node(*translator, node*)

Depart an *ItalicAbbreviationNode*.

Parameters

- **translator** (*LaTeXTranslator*)
- **node** (*ItalicAbbreviationNode*) – The node being visited.

setup(*app*)

Setup *sphinx_toolbox.formatting*.

Parameters **app** (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

github

Sphinx domain for GitHub.com, and related utilities.

New in version 2.4.0.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.github',  
]
```

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

12.1 Configuration

`github_username`

Type: `str`

Required: True

The username of the GitHub account that owns the repository this documentation corresponds to.

`github_repository`

Type: `str`

Required: True

The GitHub repository this documentation corresponds to.

12.2 Usage

`:github:issue:`

Role which shows a link to the given issue on GitHub.

If the issue exists, the link has a tooltip that shows the title of the issue.

Example

```
:github:issue:`1`
```

#1

You can also reference an issue in a different repository by adding the repository name inside `<>`.

```
:github:issue:`7680 <pytest-dev/pytest>`
```

[pytest-dev/pytest#7680](#)

:github:pull:

Role which shows a link to the given pull request on GitHub.

If the pull requests exists, the link has a tooltip that shows the title of the pull requests.

Example

```
:github:pull:`2`
```

[#2](#)

You can also reference a pull request in a different repository by adding the repository name inside `<>`.

```
:github:pull:`7671 <pytest-dev/pytest>`
```

[pytest-dev/pytest#7671](#)

:github:repo:

Role which shows a link to the given repository on GitHub.

Example

```
:github:repo:`sphinx-toolbox/sphinx-toolbox`
```

[sphinx-toolbox/sphinx-toolbox](#)

You can also use a different label for the link:.

```
See more in the :github:repo:`pytest repository <pytest-dev/pytest>`.
```

See more in the [pytest repository](#).

:github:user:

Role which shows a link to the given user on GitHub.

Example

```
:github:user:`domdfcoding`
```

[@domdfcoding](#)

You can also use a different label for the link:.

```
See more of my :github:user:`repositories <domdfcoding>`.
```

See more of my [repositories](#).

:github:org:

Role which shows a link to the given organization on GitHub.

Example

```
:github:org:`sphinx-toolbox`
```

@sphinx-toolbox

You can also use a different label for the link:.

See more repositories in the :github:org:`pytest-dev org <pytest-dev>`.

See more repositories in the [pytest-dev.org](#).

12.3 Caching

HTTP requests to obtain issue/pull request titles are cached for four hours.

To clear the cache manually, run:

```
$ python3 -m sphinx_toolbox
```

12.4 API Reference

Enable this extension from your extension's setup function like so:

```
def setup(app: Sphinx) -> Dict[str, Any]:
    app.setup_extension('sphinx_toolbox.github')
    return {}
```

This will guarantee that the following values will be available via `app.config`:

- **github_username** (`str`) – The username of the GitHub account that owns the repository this documentation corresponds to.
- **github_repository** (`str`) – The GitHub repository this documentation corresponds to.
- **github_url** (`apeye.requests_url.RequestsURL`) – The complete URL of the repository on GitHub.
- **github_source_url** (`RequestsURL`) – The base URL for the source code on GitHub.
- **github_issues_url** (`RequestsURL`) – The base URL for the issues on GitHub.
- **github_pull_url** (`RequestsURL`) – The base URL for the pull requests on GitHub.

If the user has not provided either `github_username` or `github_repository` a `MissingOptionError` will be raised.

Classes:

<code>GitHubDomain(env)</code>	Sphinx domain for GitHub.com .
--------------------------------	--

Functions:

<code>validate_config(app, config)</code>	Validate the provided configuration values.
<code>setup(app)</code>	Setup sphinx_toolbox.github .

```
class GitHubDomain(env)
    Bases: Domain
    Sphinx domain for GitHub.com.
```

validate_config(*app*, *config*)

Validate the provided configuration values.

See [ToolboxConfig](#) for a list of the configuration values.**Parameters**

- **app** ([Sphinx](#)) – The Sphinx application.
- **config** ([Config](#))

setup(*app*)Setup [sphinx_toolbox.github](#).

New in version 1.0.0.

Parameters **app** ([Sphinx](#)) – The Sphinx application.**Return type** [SphinxExtMetadata](#)

12.5 github.issues submodule

Roles and nodes for GitHub issues and Pull Requests.

Classes:

IssueNode (<i>issue_number</i> , <i>refuri</i> , <i>**kwargs</i>)	Docutils Node to represent a link to a GitHub <i>Issue</i> or <i>Pull Request</i> .
IssueNodeWithName (<i>repo_name</i> , <i>issue_number</i> , ...)	Docutils Node to represent a link to a GitHub <i>Issue</i> or <i>Pull Request</i> , with the repository name shown.

Functions:

issue_role (<i>name</i> , <i>rawtext</i> , <i>text</i> , <i>lineno</i> , <i>inliner</i>)	Adds a link to the given issue on GitHub.
pull_role (<i>name</i> , <i>rawtext</i> , <i>text</i> , <i>lineno</i> , <i>inliner</i>)	Adds a link to the given pull request on GitHub.
visit_issue_node (<i>translator</i> , <i>node</i>)	Visit an IssueNode .
depart_issue_node (<i>translator</i> , <i>node</i>)	Depart an IssueNode .
get_issue_title (<i>issue_url</i>)	Returns the title of the issue with the given url, or None if the issue isn't found.

class [IssueNode](#)(*issue_number*, *refuri*, ***kwargs*)Bases: [reference](#)Docutils Node to represent a link to a GitHub *Issue* or *Pull Request*.**Parameters**

- **issue_number** ([Union](#)[[str](#), [int](#)]) – The number of the issue or pull request.
- **refuri** ([Union](#)[[str](#), [URL](#)]) – The URL of the issue / pull request on GitHub.

class [IssueNodeWithName](#)(*repo_name*, *issue_number*, *refuri*, ***kwargs*)Bases: [IssueNode](#)Docutils Node to represent a link to a GitHub *Issue* or *Pull Request*, with the repository name shown.

New in version 2.4.0.

Parameters

- **repo_name** (`str`) – The full name of the repository, in the form `owner/name`.
- **issue_number** (`Union[str, int]`) – The number of the issue or pull request.
- **refuri** (`Union[str, URL]`) – The URL of the issue / pull request on GitHub.

issue_role(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*={}, *content*=[])

Adds a link to the given issue on GitHub.

Parameters

- **name** (*str*) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (*str*) – A string containing the entire interpreted text input, including the role and markup.
- **text** (*str*) – The interpreted text content.
- **lineno** (*int*) – The line number where the interpreted text begins.
- **inliner** (*Inliner*) – The `docutils.parsers.rst.states.Inliner` object that called `issue_role()`. It contains the several attributes useful for error reporting and document tree access.
- **options** (*Dict[str, Any]*) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default `{}`.
- **content** (*List[str]*) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default `[]`.

Return type `Tuple[List[IssueNode], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

pull_role(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*={}, *content*=[])

Adds a link to the given pull request on GitHub.

Parameters

- **name** (*str*) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (*str*) – A string containing the entire interpreted text input, including the role and markup.
- **text** (*str*) – The interpreted text content.
- **lineno** (*int*) – The line number where the interpreted text begins.
- **inliner** (*Inliner*) – The `docutils.parsers.rst.states.Inliner` object that called `pull_role()`. It contains the several attributes useful for error reporting and document tree access.
- **options** (*Dict[str, Any]*) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default `{}`.
- **content** (*List[str]*) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default `[]`.

Return type `Tuple[List[IssueNode], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

visit_issue_node(*translator*, *node*)

Visit an *IssueNode*.

If the node points to a valid issue / pull request, add a tooltip giving the title of the issue / pull request and a hyperlink to the page on GitHub.

Parameters

- **translator** (HTMLTranslator)
- **node** (*IssueNode*) – The node being visited.

depart_issue_node(*translator*, *node*)

Depart an *IssueNode*.

Parameters

- **translator** (HTMLTranslator)
- **node** (*IssueNode*) – The node being visited.

get_issue_title(*issue_url*)

Returns the title of the issue with the given url, or *None* if the issue isn't found.

Parameters *issue_url* (*str*)

Return type *Optional*[*str*]

12.6 github.repos_and_users submodule

Roles and nodes for referencing GitHub repositories and organizations.

Classes:

<i>GitHubObjectLinkNode</i> (<i>name</i> , <i>refuri</i> , <i>**kwargs</i>)	Docutils Node to represent a link to a GitHub repository.
---	---

Functions:

<i>repository_role</i> (<i>name</i> , <i>rawtext</i> , <i>text</i> , <i>lineno</i> , ...)	Adds a link to the given repository on GitHub.
<i>user_role</i> (<i>name</i> , <i>rawtext</i> , <i>text</i> , <i>lineno</i> , <i>inliner</i>)	Adds a link to the given user / organization on GitHub.
<i>visit_github_object_link_node</i> (<i>translator</i> , <i>node</i>)	Visit a <i>GitHubObjectLinkNode</i> .
<i>depart_github_object_link_node</i> (<i>translator</i> , <i>node</i>)	Depart an <i>GitHubObjectLinkNode</i> .

class *GitHubObjectLinkNode*(*name*, *refuri*, ***kwargs*)

Bases: *reference*

Docutils Node to represent a link to a GitHub repository.

Parameters

- **repo_name** – The full name of the repository, in the form *owner/name*.
- **refuri** (*Union*[*str*, *URL*]) – The URL of the issue / pull request on GitHub.

repository_role(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*={}, *content*=[])

Adds a link to the given repository on GitHub.

Parameters

- **name** (*str*) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (*str*) – A string containing the entire interpreted text input, including the role and markup.
- **text** (*str*) – The interpreted text content.
- **lineno** (*int*) – The line number where the interpreted text begins.
- **inliner** (*Inliner*) – The `docutils.parsers.rst.states.Inliner` object that called `repository_role()`. It contains the several attributes useful for error reporting and document tree access.
- **options** (*Dict[str, Any]*) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default `{}`.
- **content** (*List[str]*) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default `[]`.

Return type `Tuple[List[reference], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

user_role(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*={}, *content*=[])

Adds a link to the given user / organization on GitHub.

Parameters

- **name** (*str*) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (*str*) – A string containing the entire interpreted text input, including the role and markup.
- **text** (*str*) – The interpreted text content.
- **lineno** (*int*) – The line number where the interpreted text begins.
- **inliner** (*Inliner*) – The `docutils.parsers.rst.states.Inliner` object that called `user_role()`. It contains the several attributes useful for error reporting and document tree access.
- **options** (*Dict[str, Any]*) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default `{}`.
- **content** (*List[str]*) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default `[]`.

Return type `Tuple[List[reference], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

visit_github_object_link_node(*translator, node*)

Visit a *GitHubObjectLinkNode*.

Parameters

- **translator** (HTMLTranslator)
- **node** (*GitHubObjectLinkNode*) – The node being visited.

depart_github_object_link_node(*translator, node*)

Depart an *GitHubObjectLinkNode*.

Parameters

- **translator** (HTMLTranslator)
- **node** (*GitHubObjectLinkNode*) – The node being visited.

installation

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

```
extensions = [  
    ...  
    'sphinx_toolbox.installation',  
]
```

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

13.1 Configuration

`conda_channels`

Type: `List[str]`

Required: False

Default: `[]`

The conda channels required to install the library from Anaconda.

An alternative to setting it within the *installation* directive.

13.2 Usage

`.. installation:: name`

Adds a series of tabs providing installation instructions for the project from a number of sources.

The directive takes a single required argument – the name of the project. If the project uses a different name on PyPI and/or Anaconda, the `:pypi-name:` and `:conda-name:` options can be used to set the name for those repositories.

`:pypi: (flag)`

Flag to indicate the project can be installed from PyPI.

`:pypi-name: name (string)`

The name of the project on PyPI.

`:conda: (flag)`

Flag to indicate the project can be installed with Conda.

`:conda-name: name (string)`

The name of the project on Conda.

:conda-channels: channels (comma separated strings)

Comma-separated list of required Conda channels.

This can also be set via the `conda_channels` option.

:github: (flag)

Flag to indicate the project can be installed from GitHub.

To use this option add the following to your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.github',  
]  
  
github_username = '<your username>'  
github_repository = '<your repository>'
```

See `sphinx_toolbox.github` for more information.

Example

```
.. installation:: sphinx-toolbox  
:pypi:  
:anaconda:  
:conda-channels: domdfcoding,conda-forge  
:github:
```

13.2.2 from PyPI

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

13.2.3 from Anaconda

First add the required channels

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

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

Then install

```
$ conda install sphinx-toolbox
```

13.2.4 from GitHub

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


.. extensions::

Shows instructions on how to enable a Sphinx extension.

The directive takes a single argument – the name of the extension.

:import-name: (string)

The name used to import the extension, if different from the name of the extension.

:no-preamble: (flag)

Disables the preamble text.

:no-postamble: (flag)

Disables the postamble text.

:first: (flag)

Puts the entry for extension before its dependencies. By default is placed at the end.

New in version 0.4.0.

Example

```
.. extensions:: sphinx-toolbox
   :import-name: sphinx_toolbox

   sphinx.ext.viewcode
   sphinx_tabs.tabs
   sphinx-prompt
```

Enable sphinx-toolbox by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx.ext.viewcode',
    'sphinx_tabs.tabs',
    'sphinx-prompt',
    'sphinx_toolbox',
]
```

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

13.3 API Reference

Classes:

<code>ExtensionsDirective(name, arguments, ...)</code>	Directive to show instructions for enabling the extension.
<code>InstallationDirective(name, arguments, ...)</code>	Directive to show installation instructions.
<code>Sources(*args, **kwargs)</code>	Class to store functions that provide installation instructions for different sources.

Functions:

<code>conda_installation(options, env)</code>	Source to provide instructions for installing from Anaconda.
<code>copy_asset_files(app[, exception])</code>	Copy additional stylesheets into the HTML build directory.
<code>github_installation(options, env)</code>	Source to provide instructions for installing from GitHub.
<code>make_installation_instructions(options, env)</code>	Make the content of an installation node.
<code>pypi_installation(options, env)</code>	Source to provide instructions for installing from PyPI.
<code>setup(app)</code>	Setup <code>sphinx_toolbox.installation</code> .

Data:

<code>sources</code>	Instance of <code>Sources</code> .
----------------------	------------------------------------

class `ExtensionsDirective`(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

Directive to show instructions for enabling the extension.

Methods:

<code>run()</code>	Create the extensions node.
--------------------	-----------------------------

`run()`

Create the extensions node.

Return type `List[Node]`

class `InstallationDirective`(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

Directive to show installation instructions.

Attributes:

<code>options</code>	Mapping of option names to values.
----------------------	------------------------------------

Methods:

<code>run()</code>	Create the installation node.
<code>run_generic()</code>	Generate generic reStructuredText output.
<code>run_html()</code>	Generate output for HTML builders.

options**Type:** `Dict[str, Any]`

Mapping of option names to values.

The options are as follows:

- **pypi**: Flag to indicate the project can be installed from PyPI.
- **pypi-name**: The name of the project on PyPI.
- **conda**: Flag to indicate the project can be installed with Conda.
- **conda-name**: The name of the project on Conda.
- **conda-channels**: Comma-separated list of required Conda channels.
- **github**: Flag to indicate the project can be installed from GitHub.

The GitHub username and repository are configured in `conf.py` and are available in `env.config`.**run()**

Create the installation node.

Return type `List[Node]`**run_generic()**

Generate generic reStructuredText output.

Return type `List[Node]`**run_html()**

Generate output for HTML builders.

Return type `List[Node]`**class Sources(*args, **kwargs)**Bases: `List[Tuple[str, str, Callable, Callable, Optional[Dict[str, Callable]]]]`

Class to store functions that provide installation instructions for different sources.

The syntax of each entry is:

```
(option_name, source_name, getter_function, validator_function, extra_options)
```

- **option_name** – a string to use in the directive to specify the source to use,
- **source_name** – a string to use in the tabs to indicate the installation source,
- **getter_function** – the function that returns the installation instructions,
- **validator_function** – a function to validate the option value provided by the user,
- **extra_options** – a mapping of additional options for the directive that are used by the `getter_function`.

Methods:

<code>register(option_name, source_name[, ...])</code>	Decorator to register a function.
--	-----------------------------------

register(*option_name*, *source_name*, *validator*=<function 'unchanged'>, *extra_options*=None)

Decorator to register a function.

The function must have the following signature:

```
def function(  
    options: Dict[str, Any], # Mapping of option names to values.  
    env: sphinx.environment.BuildEnvironment, # The Sphinx build environment.  
    ) -> List[str]: ...
```

Parameters

- **option_name** (`str`) – A string to use in the directive to specify the source to use.
- **source_name** (`str`) – A string to use in tabbed installation instructions to represent this source.
- **validator** (`Callable`) – A function to validate the option value provided by the user. Default `docutils.parsers.rst.directives.unchanged()`.
- **extra_options** (`Optional[Dict[str, Callable]]`) – An optional mapping of extra option names to validator functions. Default `{}`.

Return type `Callable`

Returns The registered function.

Raises `SyntaxError` if the decorated function does not take the correct arguments.

conda_installation(*options*, *env*)

Source to provide instructions for installing from Anaconda.

Parameters

- **options** (`Dict[str, Any]`) – Mapping of option names to values.
- **env** (`BuildEnvironment`) – The Sphinx build environment.

Return type `List[str]`

copy_asset_files(*app*, *exception*=None)

Copy additional stylesheets into the HTML build directory.

New in version 1.2.0.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **exception** (`Optional[Exception]`) – Any exception which occurred and caused Sphinx to abort. Default `None`.

github_installation(*options*, *env*)

Source to provide instructions for installing from GitHub.

Parameters

- **options** (`Dict[str, Any]`) – Mapping of option names to values.
- **env** (`BuildEnvironment`) – The Sphinx build environment.

Return type `List[str]`

make_installation_instructions(*options*, *env*)

Make the content of an installation node.

Parameters

- **options** (`Dict[str, Any]`)
- **env** (`BuildEnvironment`) – The Sphinx build environment.

Return type `List[str]`

pypi_installation(*options*, *env*)

Source to provide instructions for installing from PyPI.

Parameters

- **options** (`Dict[str, Any]`) – Mapping of option names to values.
- **env** (`BuildEnvironment`) – The Sphinx build environment.

Return type `List[str]`

setup(*app*)

Setup `sphinx_toolbox.installation`.

New in version 0.7.0.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

sources

Type: `Sources`

Instance of `Sources`.

issues

Add links to GitHub issues and Pull Requests.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.issues',  
]
```

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

14.1 Usage

:issue:

Role which shows a link to the given issue on GitHub.

If the issue exists, the link has a tooltip that shows the title of the issue.

Example

```
:issue:`1`
```

#1

You can also reference an issue in a different repository by adding the repository name inside `<>`.

```
:issue:`7680 <pytest-dev/pytest>`
```

pytest-dev/pytest#7680

:pull:

Role which shows a link to the given pull request on GitHub.

If the pull requests exists, the link has a tooltip that shows the title of the pull requests.

Example

```
:pull:`2`
```

#2

You can also reference a pull request in a different repository by adding the repository name inside `<>`.

```
:pull:`7671 <pytest-dev/pytest>`
```

`pytest-dev/pytest#7671`

Changed in version 2.4.0: *issue* and *pull* now show the repository name when the name differs from that configured in `conf.py`.

Changed in version 2.4.0: These directives are also available in the *github* domain.

The only difference between the *issue* and *pull* roles is in the URL. GitHub uses the same numbering scheme for issues and pull requests, and automatically redirects to the pull request if the user tries to navigate to an issue with that same number.

14.2 Caching

HTTP requests to obtain issue/pull request titles are cached for four hours.

To clear the cache manually, run:

```
$ python3 -m sphinx_toolbox
```

14.3 API Reference

Changed in version 2.4.0: The following moved to *sphinx_toolbox.github.issues*:

- *IssueNode*
- *IssueNodeWithName*
- *issue_role()*
- *pull_role()*
- *visit_issue_node()*
- *depart_issue_node()*
- *get_issue_title()*

Functions:

<i>setup</i> (app)	Setup <i>sphinx_toolbox.issues</i> .
--------------------	--------------------------------------

setup(app)

Setup *sphinx_toolbox.issues*.

New in version 1.0.0.

Parameters *app* (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

latex

Sphinx utilities for LaTeX builders.

New in version 2.8.0.

In addition to the developer API (see below), `sphinx_toolbox.latex` configures Sphinx and LaTeX to correctly handle symbol footnotes.

Changed in version 2.12.0: Sphinx is also configured to respect `.. only:: html` etc. directives surrounding toctree directives when determining the overall toctree depth.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.latex',  
]
```

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

15.1 Example Footnotes

Hello¹

Goodbye²

Symbol^{*}

Another Symbol[†]

Number Again³

Symbol 3[‡]

Symbol 4[§]

Symbol 5[¶]

Symbol 6^{||}

Symbol 7^{**}

Symbol 8^{††}

¹ One

² Two

^{*} Buckle my shoe

[†] The second symbol

³ The number after the symbol

[‡] Symbol 3

[§] Symbol 4

[¶] Symbol 5

^{||} Symbol 6

^{**} Symbol 7

^{††} Symbol 8

Symbol 9^{‡‡}

15.2 Usage

.. latex:samepage::

.. samepage::

Configures LaTeX to make all content within this directive appear on the same page.

This can be useful to avoid awkward page breaks.

This directive has no effect with non-LaTeX builders.

New in version 2.9.0.

.. latex:clearpage::

.. clearpage::

Configures LaTeX to start a new page.

This directive has no effect with non-LaTeX builders.

New in version 2.10.0.

See also: [latex:cleardoublepage](#)

.. latex:cleardoublepage::

.. cleardoublepage::

Configures LaTeX to start a new page.

In a two-sided printing it also makes the next page a right-hand (odd-numbered) page, inserting a blank page if necessary.

This directive has no effect with non-LaTeX builders.

New in version 2.10.0.

See also: [latex:clearpage](#)

.. latex:vspace:: space

Configures LaTeX to add or remove vertical space.

The value for `space` is passed verbatim to the `\vspace{}` command. Ensure you pass a valid value.

This directive has no effect with non-LaTeX builders.

New in version 2.11.0.

15.3 API Reference

Functions:

use_package (package, config, *args, **kwargs)	Configure LaTeX to use the given package.
visit_footnote (translator, node)	Visit a <code>docutils.nodes.footnote</code> node with the LaTeX translator.
depart_footnote (translator, node)	Depart a <code>docutils.nodes.footnote</code> node with the LaTeX translator.
replace_unknown_unicode (app[, exception])	Replaces certain unknown unicode characters in the Sphinx LaTeX output with the best equivalents.
better_header_layout (config[, space_before, ...])	Makes LaTeX chapter names lowercase, and adjusts the spacing above and below the chapter name.
configure (app, config)	Configure sphinx_toolbox.latex .
setup (app)	Setup sphinx_toolbox.latex .

Classes:

SamepageDirective (name, arguments, options, ...)	Directive which configures LaTeX to make all content within this directive appear on the same page.
---	---

continues on next page

^{‡‡} Symbol 9

Table 2 – continued from previous page

<code>ClearPageDirective</code> (name, arguments, options, ...)	Directive which configures LaTeX to start a new page.
<code>ClearDoublePageDirective</code> (name, arguments, ...)	Directive which configures LaTeX to start a new page.
<code>VSpaceDirective</code> (name, arguments, options, ...)	Directive which configures LaTeX to add or remove vertical space.
<code>LaTeXDomain</code> (env)	Domain containing various LaTeX-specific directives.

use_package(package, config, *args, **kwargs)

Configure LaTeX to use the given package.

The `\usepackage` entry is added to the `sphinx.config.Config.latex_elements["preamble"]` attribute.

Parameters

- **package** (str)
- **config** (Config)
- ***args**
- ****kwargs**

visit_footnote(translator, node)

Visit a `docutils.nodes.footnote` node with the LaTeX translator.

Unlike the default `visit_footnote` function, this one handles footnotes using symbols.

New in version 2.8.0.

Parameters

- **translator** (LaTeXTranslator)
- **node** (footnote)

depart_footnote(translator, node)

Depart a `docutils.nodes.footnote` node with the LaTeX translator.

New in version 2.8.0.

Parameters

- **translator** (LaTeXTranslator)
- **node** (footnote)

class SamepageDirective(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)

Bases: `SphinxDirective`

Directive which configures LaTeX to make all content within this directive appear on the same page.

This can be useful to avoid awkward page breaks.

This directive has no effect with non-LaTeX builders.

New in version 2.9.0.

class ClearPageDirective(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [SphinxDirective](#)

Directive which configures LaTeX to start a new page.

This directive has no effect with non-LaTeX builders.

New in version 2.10.0.

See also: [ClearDoublePageDirective](#)

class ClearDoublePageDirective(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [SphinxDirective](#)

Directive which configures LaTeX to start a new page.

In a two-sided printing it also makes the next page a right-hand (odd-numbered) page, inserting a blank page if necessary.

This directive has no effect with non-LaTeX builders.

New in version 2.10.0.

See also: [ClearPageDirective](#)

class VSpaceDirective(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [SphinxDirective](#)

Directive which configures LaTeX to add or remove vertical space.

This directive has no effect with non-LaTeX builders.

New in version 2.11.0.

class LaTeXDomain(*env*)

Bases: [Domain](#)

Domain containing various LaTeX-specific directives.

New in version 2.11.0.

replace_unknown_unicode(*app, exception=None*)

Replaces certain unknown unicode characters in the Sphinx LaTeX output with the best equivalents.

This function can be hooked into the [build-finished](#) event as follows:

```
app.connect("build-finished", replace_unknown_unicode)
```

New in version 2.9.0.

Parameters

- **app** ([Sphinx](#)) – The Sphinx application.
- **exception** ([Optional\[Exception\]](#)) – Any exception which occurred and caused Sphinx to abort. Default [None](#).

better_header_layout(*config*, *space_before*=10, *space_after*=20)

Makes LaTeX chapter names lowercase, and adjusts the spacing above and below the chapter name.

New in version 2.10.0.

Parameters

- **config** (Config) – The Sphinx configuration object.
- **space_before** (int) – The space, in pixels, before the chapter name. Default 10.
- **space_after** (int) – The space, in pixels, after the chapter name. Default 20.

configure(*app*, *config*)

Configure *sphinx_toolbox.latex*.

Parameters

- **app** (Sphinx) – The Sphinx application.
- **config** (Config)

setup(*app*)

Setup *sphinx_toolbox.latex*.

New in version 2.8.0.

Parameters **app** (Sphinx) – The Sphinx application.

pre_commit

Sphinx extension to show examples of `.pre-commit-config.yaml` configuration.

New in version 1.6.0.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.pre_commit',  
]
```

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

16.1 Usage

.. pre-commit::

Directive which shows an example snippet of `.pre-commit-config.yaml`.

:rev: (string)

The revision or tag to clone at.

:hooks: (comma separated list)

A list of hooks IDs to document.

If not given the hooks will be parsed from `.pre-commit-hooks.yaml`.

:args: (comma separated list)

A list arguments that should or can be provided to the first hook ID.

New in version 1.7.2.

Example

```
.. pre-commit::  
    :rev: v0.0.4  
    :hooks: some-hook,some-other-hook
```

```
- repo: https://github.com/sphinx-toolbox/sphinx-toolbox  
  rev: v0.0.4  
  hooks:  
  - id: some-hook  
  - id: some-other-hook
```

.. pre-commit:flake8:: version

Directive which shows an example snippet of `.pre-commit-config.yaml` for a flake8 plugin.

The directive takes a single argument – the version of the flake8 plugin to install from PyPI.

:flake8-version: (string)

The version of flake8 to use. Default 3.8.4.

:plugin-name: (string)

The name of the plugin to install from PyPI. Defaults to the repository name.

Example

```
.. pre-commit:flake8:: 0.0.4

- repo: https://github.com/pycqa/flake8
  rev: 3.8.4
  hooks:
  - id: flake8
    additional_dependencies:
    - sphinx-toolbox==0.0.4
```

Changed in version 2.8.0: The repository URL now points to GitHub.

16.2 API Reference

Classes:

<code>Flake8PreCommitDirective(name, arguments, ...)</code>	A Sphinx directive for documenting flake8 plugins' pre-commit hooks.
<code>PreCommitDirective(name, arguments, options, ...)</code>	A Sphinx directive for documenting pre-commit hooks.

Functions:

<code>parse_hooks(hooks)</code>	Parses the comma, semicolon and/or space delimited list of hook IDs.
<code>setup(app)</code>	Setup <code>sphinx_toolbox.pre_commit</code> .

class Flake8PreCommitDirective(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

A Sphinx directive for documenting flake8 plugins' pre-commit hooks.

class PreCommitDirective(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

A Sphinx directive for documenting pre-commit hooks.

parse_hooks(*hooks*)

Parses the comma, semicolon and/or space delimited list of hook IDs.

Parameters **hooks** (`str`)

Return type `List[str]`

setup(*app*)

Setup `sphinx_toolbox.pre_commit`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

rest_example

Directive to show example reStructuredText and the rendered output.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.rest_example',  
]
```

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

17.1 Usage

.. rest-example::

Directive to show example reStructuredText and the rendered output.

:force: (flag)

If given, minor errors on highlighting are ignored.

:emphasize-lines: line numbers (comma separated numbers)

Emphasize particular lines of the code block:

:tab-width: number (number)

Sets the size of the indentation in spaces.

:dedent: number (number)

Strip indentation characters from the code block,

Example

```
.. rest-example::  
  
    :source: `sphinx_toolbox/config.py`  
  
    Here is the :source: `source code <sphinx_toolbox/config.py>`
```

```
:source: `sphinx_toolbox/config.py`  
  
Here is the :source: `source code <sphinx_toolbox/config.py>`
```

`sphinx_toolbox/config.py`

Here is the source code

17.2 API Reference

Classes:

<code>reSTExampleDirective</code> (name, arguments, ...)	Directive to show some reStructuredText source, and the rendered output.
--	--

Functions:

<code>make_rest_example</code> (options, env, content)	Make the content of a reST Example node.
<code>setup</code> (app)	Setup <code>sphinx_toolbox.rest_example</code> .

Data:

<code>rest_example_purger</code>	Purger to track rest-example nodes, and remove redundant ones.
----------------------------------	--

class `reSTExampleDirective`(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)

Bases: `SphinxDirective`

Directive to show some reStructuredText source, and the rendered output.

Methods:

<code>run</code> ()	Create the rest_example node.
---------------------	-------------------------------

run()
Create the rest_example node.

Return type `List[Node]`

make_rest_example(options, env, content)
Make the content of a reST Example node.

Parameters

- **options** (`Dict[str, Any]`)
- **content** (`Sequence[str]`) – The user-provided content of the directive.

Return type `List[str]`

rest_example_purger = `Purger('all_rest_example_nodes')`
Type: `Purger`

Purger to track rest-example nodes, and remove redundant ones.

setup(app)
Setup `sphinx_toolbox.rest_example`.

New in version 0.7.0.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

shields

Directives for shield/badge images.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.shields',  
]
```

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

18.1 Usage

Several shield/badge directives are available, like this one:

They function similarly to the `.. image::` directives, although not all options are available. As with the `image` directive, shields can be used as part of substitutions, e.g.

```
This repository uses pre-commit |pre-commit|  
  
.. |pre-commit| pre-commit::
```

All shields have the following options:

:alt:

Alternative text for the shield, used when the image cannot be displayed or the user uses a screen reader.

:height:

:width:

:scale:

The height/width/scale of the shield.

:name:

:class: (string)

Additional CSS class for the shield. All shields have the `sphinx_toolbox_shield` class by default.

18.1.1 Shields

.. rtd-shield::

Shield to show the [ReadTheDocs](#) documentation build status.

:project:

The name of the project on *ReadTheDocs*.

:version:

The documentation version. Default latest.

:target:

The hyperlink target of the shield. Useful if the documentation uses a custom domain.

New in version 1.8.0.

.. pypi-shield::

Shield to show information about the project on [PyPI](#).

:project:

The name of the project on *PyPI*.

Only one of the following options is permitted:

:version:

Show the package version.

:py-versions:

Show the supported python versions.

:implementations:

Show the supported python implementations.

:wheel:

Show whether the package has a wheel.

:license:

Show the license listed on PyPI.

:downloads:

Show the downloads for the given period (day / week / month)

Changed in version 2.5.0: Shields created with this option now link to [pypistats](#).

.. maintained-shield::

Shield to indicate whether the project is maintained.

Takes a single argument: the current year.

.. github-shield::

Shield to show information about a GitHub repository.

:username:

The GitHub username. Defaults to *github_username*.

:repository:

The GitHub repository. Defaults to *github_repository*.

:branch:
 The branch to show information about. Default `master`.
 Required for `commits-since` and `last-commit`.

Only one of the following options is permitted:

:contributors: (flag)
 Show the number of contributors.

:commits-since: tag (string)
 Show the number of commits since the given tag.

:last-commit: (flag)
 Show the date of the last commit.

:top-language: (flag)
 Show the top language and percentage.

:license: (flag)
 Show the license detected by GitHub.

.. actions-shield::
 Shield to show the *GitHub Actions* build status.

:username:
 The GitHub username. Defaults to `github_username`.

:repository:
 The GitHub repository. Defaults to `github_repository`.

:workflow:
 The workflow to show the status for.

.. requires-io-shield::
 Shield to show the *Requires.io* status.

:username:
 The GitHub username. Defaults to `github_username`.

:repository:
 The GitHub repository. Defaults to `github_repository`.

:branch:
 The branch to show the build status for. Default `master`.

.. coveralls-shield::
 Shield to show the code coverage from *Coveralls.io*.

:username:
 The GitHub username. Defaults to `github_username`.

:repository:
 The GitHub repository. Defaults to `github_repository`.

:branch:
 The branch to show the build status for. Default `master`.

.. codefactor-shield::
Shield to show the code quality score from [Codefactor](#).

:username:
 The GitHub username. Defaults to [github_username](#).

:repository:
 The GitHub repository. Defaults to [github_repository](#).

.. pre-commit-shield::
Shield to indicate that the project uses [pre-commit](#).

.. pre-commit-ci-shield::
New in version 1.7.0.

 Shield to show the [pre-commit.ci](#) status.

:username:
 The GitHub username. Defaults to [github_username](#).

:repository:
 The GitHub repository. Defaults to [github_repository](#).

:branch:
 The branch to show the status for. Default master.

Data:

SHIELDS_IO	Base URL for shields.io
shield_default_option_spec	Options common to all shields.

Classes:

Shield (name, arguments, options, content, ...)	Directive for shields.io shields/badges.
RTFDShield (name, arguments, options, ...)	Shield to show the ReadTheDocs documentation build status.
PyPiShield (name, arguments, options, ...)	Shield to show information about the project on PyPI .
MaintainedShield (name, arguments, options, ...)	Shield to indicate whether the project is maintained.
GitHubBackedShield (name, arguments, options, ...)	Base class for badges that are based around GitHub.
GitHubShield (name, arguments, options, ...)	Shield to show information about a GitHub repository.
GitHubActionsShield (name, arguments, ...)	Shield to show the GitHub Actions build status.
RequiresIOShield (name, arguments, options, ...)	Shield to show the Requires.io status.
CoverallsShield (name, arguments, options, ...)	Shield to show the code coverage from Coveralls.io .
CodefactorShield (name, arguments, options, ...)	Shield to show the code quality score from Codefactor .
PreCommitShield (name, arguments, options, ...)	Shield to indicate that the project uses pre-commit .
PreCommitCIShield (name, arguments, options, ...)	Shield to show the pre-commit.ci status.

Functions:

copy_asset_files (app[, exception])	Copy additional stylesheets into the HTML build directory.
setup (app)	Setup sphinx_toolbox.shields .

18.2 API Reference

SHIELDS_IO = `URL('https://img.shields.io')`

Type: `URL`

Base URL for shields.io

shield_default_option_spec

Type: `Mapping[str, Callable[[str], Any]]`

Options common to all shields.

class Shield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

Directive for shields.io shields/badges.

class RTFDShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `Shield`

Shield to show the [ReadTheDocs](#) documentation build status.

Changed in version 1.8.0: Added the `:target:` option, to allow a custom target to be specified. Useful if the documentation uses a custom domain.

class PyPiShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `Shield`

Shield to show information about the project on [PyPI](#).

class MaintainedShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `Shield`

Shield to indicate whether the project is maintained.

class GitHubBackedShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `Shield`

Base class for badges that are based around GitHub.

get_repo_details()

Returns the username and repository name, either parsed from the directive's options or from `conf.py`.

class GitHubShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `GitHubBackedShield`

Shield to show information about a GitHub repository.

class GitHubActionsShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `GitHubBackedShield`

Shield to show the *GitHub Actions* build status.

class RequiresIOShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [GitHubBackedShield](#)

Shield to show the [Requires.io](#) status.

class CoverallsShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [GitHubBackedShield](#)

Shield to show the code coverage from [Coveralls.io](#).

class CodefactorShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [GitHubBackedShield](#)

Shield to show the code quality score from [Codefactor](#).

class PreCommitShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [Shield](#)

Shield to indicate that the project uses [pre-commit](#).

class PreCommitCIShield(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: [GitHubBackedShield](#)

Shield to show the [pre-commit.ci](#) status.

New in version 1.7.0.

copy_asset_files(*app, exception=None*)

Copy additional stylesheets into the HTML build directory.

New in version 2.3.1.

Parameters

- **app** ([Sphinx](#)) – The Sphinx application.
- **exception** ([Optional](#)[[Exception](#)]) – Any exception which occurred and caused Sphinx to abort. Default [None](#).

setup(*app*)

Setup [sphinx_toolbox.shields](#).

Parameters **app** ([Sphinx](#)) – The Sphinx application.

Return type [SphinxExtMetadata](#)

sidebar_links

Directive which adds a toctree to the sidebar containing links to the GitHub repository, PyPI project page etc.

New in version 2.9.0.

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

```
extensions = [
    ...
    'sphinx_toolbox.sidebar_links',
]
```

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

19.1 Usage

.. sidebar-links::

Adds a toctree to the sidebar containing links to the GitHub repository, PyPI project page etc. The toctree is only shown in the sidebar and is hidden with non-HTML builders.

Note: This directive can only be used on the root document (i.e. `index.rst`).

:github: (flag)

Flag to add a link to the project's GitHub repository.

To use this option add the following to your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.github',
]

github_username = '<your username>'
github_repository = '<your repository>'
```

See `sphinx_toolbox.github` for more information.

:pypi: (string)

Flag to add a link to the project page on PyPI.

The name of the project on PyPI must be passed as the option's value.

:caption: (string)

The caption of the toctree. Defaults to `Links`

Additional toctree entries may be added as the content of the directive, in the same manner as normal toctrees.

19.2 API Reference

Classes:

<code>SidebarLinksDirective(name, arguments, ...)</code>	Directive which adds a toctree to the sidebar containing links to the GitHub repository, PyPI project page etc.
--	---

Functions:

<code>setup(app)</code>	Setup <code>sphinx_toolbox.sidebar_links</code> .
-------------------------	---

class `SidebarLinksDirective`(*name, arguments, options, content, lineno, content_offset, block_text, state, state_machine*)

Bases: `SphinxDirective`

Directive which adds a toctree to the sidebar containing links to the GitHub repository, PyPI project page etc.

Methods:

<code>process_github_option()</code>	Process the <code>:github:</code> flag.
<code>run()</code>	Create the installation node.

process_github_option()

Process the `:github:` flag.

Return type `str`

run()

Create the installation node.

Return type `List[Node]`

setup(*app*)

Setup `sphinx_toolbox.sidebar_links`.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

source

Add hyperlinks to source files, either on GitHub or in the documentation itself.

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

```
extensions = [
    ...
    'sphinx_toolbox.source',
]
```

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

If you're looking for a `[source]` button to go at the end of your class and function signatures, checkout `sphinx.ext.linkcode` and `sphinx.ext.viewcode`.

20.1 Usage

source_link_target

Type: `str`

Required: False

Default: 'Sphinx'

The target of the source link, either 'GitHub' or 'Sphinx'. Case insensitive.

:source:

Role which shows a link to the given source file, either on GitHub or within the Sphinx documentation.

By default, the link points to the code within the documentation, but can be configured to point to GitHub by setting `source_link_target` to 'GitHub'.

Example

```
:source: `sphinx_toolbox/config.py`
```

Here is the `:source:` `source code` `<sphinx_toolbox/config.py>`

`sphinx_toolbox/config.py`

Here is the source code

20.2 API Reference

Functions:

<code>source_role(name, rawtext, text, lineno, inliner)</code>	Adds a link to the given Python source file in the documentation or on GitHub.
<code>setup(app)</code>	Setup <code>sphinx_toolbox.source</code> .

source_role(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*={}, *content*=[])

Adds a link to the given Python source file in the documentation or on GitHub.

Parameters

- **name** (*str*) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (*str*) – A string containing the entire interpreted text input, including the role and markup.
- **text** (*str*) – The interpreted text content.
- **lineno** (*int*) – The line number where the interpreted text begins.
- **inliner** (*Inliner*) – The `docutils.parsers.rst.states.Inliner` object that called [`source_role\(\)`](#). It contains the several attributes useful for error reporting and document tree access.
- **options** (*Dict*) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default {}.
- **content** (*List[str]*) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default [].

Return type `Tuple[Sequence[Node], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

Changed in version 2.8.0: Now returns a sequence of `nodes.reference` and `addnodes.pending_xref` as the first tuple element, rather than `nodes.reference` and `addnodes.pending_xref` as in previous versions.

setup(*app*)

Setup [`sphinx_toolbox.source`](#).

Parameters **app** (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

wikipedia

Sphinx extension to create links to Wikipedia articles.

New in version 0.2.0.

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

```
extensions = [  
    ...  
    'sphinx_toolbox.wikipedia',  
]
```

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

21.1 Configuration

wikipedia_lang

Type: `str`

Required: False

Default: `'en'`

The Wikipedia language to use for *wikipedia* roles.

New in version 0.2.0.

21.2 Usage

:wikipedia:

Role which shows a link to the given article on Wikipedia.

The title and language can be customised.

Example

```
:wikipedia:`Sphinx`  
  
:wikipedia:`mythical creature <Sphinx>`  
  
:wikipedia:`Answer to the Ultimate Question of Life, the Universe, and Everything  
↪<:de:42 (Antwort)>`
```

Sphinx

mythical creature

Answer to the Ultimate Question of Life, the Universe, and Everything

21.3 API Reference

Functions:

<code>make_wikipedia_link(name, rawtext, text, ...)</code>	Adds a link to the given article on Wikipedia .
<code>setup(app)</code>	Setup sphinx-toolbox.wikipedia .

make_wikipedia_link(*name*, *rawtext*, *text*, *lineno*, *inliner*, *options*=*{}*, *content*=*[]*)

Adds a link to the given article on [Wikipedia](#).

Parameters

- **name** (*str*) – The local name of the interpreted role, the role name actually used in the document.
- **rawtext** (*str*) – A string containing the entire interpreted text input, including the role and markup.
- **text** (*str*) – The interpreted text content.
- **lineno** (*int*) – The line number where the interpreted text begins.
- **inliner** (*Inliner*) – The `docutils.parsers.rst.states.Inliner` object that called `source_role()`. It contains the several attributes useful for error reporting and document tree access.
- **options** (*Dict*) – A dictionary of directive options for customization (from the `role` directive), to be interpreted by the function. Used for additional attributes for the generated elements and other functionality. Default `{}`.
- **content** (*List[str]*) – A list of strings, the directive content for customization (from the `role` directive). To be interpreted by the function. Default `[]`.

Return type `Tuple[List[reference], List[system_message]]`

Returns A list containing the created node, and a list containing any messages generated during the function.

setup(*app*)

Setup [sphinx-toolbox.wikipedia](#).

New in version 1.0.0.

Parameters **app** (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

22.1 more_autodoc.augment_defaults

Sphinx's autodoc module allows for default options to be set, and allows for those defaults to be disabled for an `:auto*` directive and different values given instead.

However, it does not appear to be possible to augment the defaults, such as to globally exclude certain members and then exclude additional members of a single class. This module monkeypatches in that behaviour.

Enable `sphinx_toolbox.more_autodoc.augment_defaults` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.more_autodoc.augment_defaults',  
]
```

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

Changed in version 0.6.0: Moved from `sphinx_toolbox.autodoc_augment_defaults`.

Functions:

<code>process_documenter_options(documenter, ...)</code>	Recognize options of Documenter from user input.
<code>setup(app)</code>	Setup <code>sphinx_toolbox.more_autodoc.augment_defaults</code> .

process_documenter_options(*documenter*, *config*, *options*)

Recognize options of Documenter from user input.

Parameters

- **documenter** (`Type[Documenter]`)
- **config** (`Config`)
- **options** (`Dict`)

Return type `Options`

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.augment_defaults`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.2 more_autodoc.autonamedtuple

A Sphinx directive for documenting `NamedTuples` in Python.

New in version 0.8.0.

Enable `sphinx_toolbox.more_autodoc.autonamedtuple` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.autonamedtuple',
]
```

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

Changed in version 1.5.0: `__new__` methods are documented regardless of other exclusion settings if the annotations differ from the `namedtuple` itself.

22.2.1 Usage

`.. autonamedtuple::`

Directive to automatically document a `typing.NamedTuple` or `collections.namedtuple()`.

The output is based on the `autoclass` directive. The list of parameters and the attributes are replaced by a list of Fields, combining the types and docstrings from the class docstring individual attributes. These will always be shown regardless of the state of the `:members:` option.

Otherwise the directive behaves the same as `autoclass`, and takes all of its arguments. See <https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html> for further information.

New in version 0.8.0.

`:namedtuple:`

Role which provides a cross-reference to the documentation generated by `autonamedtuple`.

See also: <https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html>

Examples

```
1 # Examples from
2 # https://docs.python.org/3/library/typing.html#typing.NamedTuple
3 # https://www.python.org/dev/peps/pep-0589/#totality
4 # https://github.com/python/typing/pull/700
5
6 # stdlib
7 import collections
8 from typing import NamedTuple
9
10 __all__ = ["Animal", "Employee", "Movie"]
11
12
13 class Animal(NamedTuple):
```

(continues on next page)

(continued from previous page)

```

14     """
15     An animal.
16
17     :param name: The name of the animal.
18     :param voice: The animal's voice.
19     """
20
21     name: str
22     voice: str
23
24
25 class Employee(NamedTuple):
26     """
27     Represents an employee.
28
29     :param id: The employee's ID number
30     """
31
32     #: The employee's name
33     name: str
34
35     id: int = 3
36
37     def __repr__(self) -> str:
38         return f'<Employee {self.name}, id={self.id}>'
39
40     def is_executive(self) -> bool:
41         """
42         Returns whether the employee is an executive.
43
44         Executives have ID numbers < 10.
45         """
46
47
48 class Movie(NamedTuple):
49     """
50     Represents a movie.
51     """
52
53     #: The name of the movie.
54     name: str
55
56     #: The movie's release year.
57     year: int
58
59     based_on: str

```

```

.. automodule:: autonamedtuple_demo
   :no-autosummary:
   :exclude-members: Movie

.. autonamedtuple:: autonamedtuple_demo.Movie

```

(continues on next page)

(continued from previous page)

This function takes a single argument, the `:namedtuple:~.Movie` to watch.

namedtuple `Animal(name, voice)`Bases: `NamedTuple`

An animal.

Fields0) **name** (`str`) – The name of the animal.1) **voice** (`str`) – The animal’s voice.`__repr__()`

Return a nicely formatted representation string

namedtuple `Employee(name, id=3)`Bases: `NamedTuple`

Represents an employee.

Fields0) **name** (`str`) – The employee’s name1) **id** (`int`) – The employee’s ID number`__repr__()`

Return repr(self).

Return type `str`**is_executive()**

Returns whether the employee is an executive.

Executives have ID numbers < 10.

Return type `bool`**namedtuple** `Movie(name, year, based_on)`Bases: `NamedTuple`

Represents a movie.

Fields0) **name** (`str`) – The name of the movie.1) **year** (`int`) – The movie’s release year.2) **based_on** (`str`) – Alias for field number 2

`__repr__()`

Return a nicely formatted representation string

This function takes a single argument, the *Movie* to watch.

22.2.2 API Reference

Classes:

<code>NamedTupleDocumenter</code> (directive, name[, indent])	Sphinx autodoc Documenter for documenting <code>typing.NamedTuples</code> .
---	---

Functions:

<code>setup</code> (app)	Setup <code>sphinx_toolbox.more_autodoc.autonamedtuple</code> .
--------------------------	---

class `NamedTupleDocumenter`(*directive, name, indent=""*)

Bases: `ClassDocumenter`

Sphinx autodoc Documenter for documenting `typing.NamedTuples`.

New in version 0.8.0.

Changed in version 0.1.0: Will no longer attempt to find attribute docstrings from other namedtuple classes.

Methods:

<code>add_content</code> (more_content[, no_docstring])	Add extra content (from docstrings, attribute docs etc.), but not the <code>typing.NamedTuple</code> 's docstring.
<code>add_directive_header</code> (sig)	Add the directive's header, and the inheritance information if the <code>:show-inheritance:</code> flag set.
<code>can_document_member</code> (member, membername, ...)	Called to see if a member can be documented by this documenter.
<code>filter_members</code> (members, want_all)	Filter the list of members to always include <code>__new__</code> if it has a different signature to the tuple.
<code>sort_members</code> (documenters, order)	Sort the <code>typing.NamedTuple</code> 's members.

add_content(*more_content, no_docstring=True*)

Add extra content (from docstrings, attribute docs etc.), but not the `typing.NamedTuple`'s docstring.

Parameters

- **more_content** (*Any*)
- **no_docstring** (*bool*) – Default `True`.

add_directive_header(*sig*)

Add the directive's header, and the inheritance information if the `:show-inheritance:` flag set.

Parameters **sig** (*str*) – The `NamedTuple`'s signature.

classmethod `can_document_member`(*member*, *membername*, *isattr*, *parent*)

Called to see if a member can be documented by this documenter.

Parameters

- **member** (*Any*) – The member being checked.
- **membername** (*str*) – The name of the member.
- **isattr** (*bool*)
- **parent** (*Any*) – The parent of the member.

Return type *bool*

filter_members(*members*, *want_all*)

Filter the list of members to always include `__new__` if it has a different signature to the tuple.

Parameters

- **members** (*List*[*Tuple*[*str*, *Any*]])
- **want_all** (*bool*)

Return type *List*[*Tuple*[*str*, *Any*, *bool*]]

sort_members(*documenters*, *order*)

Sort the `typing.NamedTuple`'s members.

Parameters

- **documenters** (*List*[*Tuple*[*Documenter*, *bool*]])
- **order** (*str*)

Return type *List*[*Tuple*[*Documenter*, *bool*]]

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.autonamedtuple`.

New in version 0.8.0.

Parameters **app** (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

22.3 more_autodoc.autoprotocol

A Sphinx directive for documenting `Protocols` in Python.

New in version 0.2.0.

Enable `sphinx_toolbox.more_autodoc.autoprotocol` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.more_autodoc.autoprotocol',  
]
```

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

Changed in version 0.6.0: Moved from `sphinx_toolbox.autoprotocol`.

Changed in version 2.13.0: Added support for generic bases, such as `class SupportsAbs(Protocol[T_co]):` .
...

22.3.1 Usage

.. autoprotocol::

Directive to automatically document a `typing.Protocol`.

The output is based on the `autoclass` directive, but with a few differences:

- Private members are always excluded.
- Special members (dunder methods) are always included.
- Undocumented members are always included.

The following options from `autoclass` are available:

:noindex: (flag)

Do not generate index entries for the documented object (and all autodocumented members).

:member-order: (string)

Override the global value of `autodoc_member_order` for one directive.

:show-inheritance: (flag)

Inserts a list of base classes just below the protocol's signature.

See <https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html> for further information.

:protocol:

Role which provides a cross-reference to the documentation generated by `autoprotocol`.

See also: <https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html>

Examples:

```
1  # stdlib
2  from abc import abstractmethod
3  from typing import Any, TypeVar
4
5  # 3rd party
6  from domdf_python_tools.doctools import prettify_docstrings
7  from typing_extensions import Protocol, runtime_checkable
8
9  __all__ = ["HasLessThan", "HasGreaterThan", "Frobnicator"]
10
11
12  @prettify_docstrings
13  class HasLessThan(Protocol):
14      """
15      :class:`typing.Protocol` for classes that support the ``<`` operator.
16      """
17
18      def __lt__(self, other) -> bool: ...
19
20
21  @prettify_docstrings
22  class HasGreaterThan(Protocol):
23
24      def __gt__(self, other) -> bool: ...
25
26
27  @runtime_checkable
28  class Frobnicator(Protocol):
29
30      def frobnicate(self, something) -> Any: ...
31
```

```
.. automodule:: autoprotocol_demo
   :members:
   :no-autosummary:
   :exclude-members: HasGreaterThan
.. autoprotocol:: autoprotocol_demo.HasGreaterThan
```

The objects being sorted must implement the `:protocol:`~.HasGreaterThan`` protocol.

protocol Frobnicator

Bases: `Protocol`

This protocol is `runtime checkable`.

Classes that implement this protocol must have the following methods / attributes:

frobnicate(*something*)

Return type `Any`

protocol HasLessThanBases: `Protocol``typing.Protocol` for classes that support the < operator.

Classes that implement this protocol must have the following methods / attributes:

`__lt__(other)`Return `self < other`.**Return type** `bool`**protocol HasGreaterThan**Bases: `Protocol`

Classes that implement this protocol must have the following methods / attributes:

`__gt__(other)`Return `self > other`.**Return type** `bool`The objects being sorted must implement the `HasGreaterThan` protocol.

22.3.2 API Reference

Classes:

<code>ProtocolDocumenter(directive, name[, indent])</code>	Sphinx autodoc Documenter for documenting <code>typing.Protocol</code> s.
--	---

Functions:

<code>setup(app)</code>	Setup <code>sphinx_toolbox.more_autodoc.autoprotocol</code> .
-------------------------	---

class ProtocolDocumenter(directive, name, indent="")Bases: `ClassDocumenter`Sphinx autodoc Documenter for documenting `typing.Protocol`s.**Methods:**

<code>add_content(more_content[, no_docstring])</code>	Add the autodocumenter content.
<code>add_directive_header(sig)</code>	Add the directive header.
<code>can_document_member(member, membername, ...)</code>	Called to see if a member can be documented by this documenter.
<code>document_members([all_members])</code>	Generate reST for member documentation.
<code>filter_members(members, want_all)</code>	Filter the given member list.
<code>format_signature(**kwargs)</code>	Protocols do not have a signature.

add_content(*more_content*, *no_docstring=False*)

Add the autodocenter content.

Parameters

- **more_content** (*Any*)
- **no_docstring** (*bool*) – Default *False*.

add_directive_header(*sig*)

Add the directive header.

Parameters **sig** (*str*)

classmethod can_document_member(*member*, *membername*, *isattr*, *parent*)

Called to see if a member can be documented by this documenter.

Parameters

- **member** (*Any*) – The member being checked.
- **membername** (*str*) – The name of the member.
- **isattr** (*bool*)
- **parent** (*Any*) – The parent of the member.

Return type *bool*

document_members(*all_members=False*)

Generate reST for member documentation.

All members are always documented.

filter_members(*members*, *want_all*)

Filter the given member list.

Parameters

- **members** (*List*[*Tuple*[*str*, *Any*]])
- **want_all** (*bool*)

Return type *List*[*Tuple*[*str*, *Any*, *bool*]]

format_signature(***kwargs*)

Protocols do not have a signature.

Return type *str*

setup(*app*)
 Setup `sphinx_toolbox.more_autodoc.autoprotocol`.

Parameters *app* (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.4 more_autodoc.autotypeddict

A Sphinx directive for documenting `TypedDicts` in Python.

Only supports `typing_extensions`'s `TypedDict` until python/typing#700 is implemented in CPython.

New in version 0.5.0.

Enable `sphinx_toolbox.more_autodoc.autotypeddict` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.autotypeddict',
]
```

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

Changed in version 0.6.0: Moved from `sphinx_toolbox.autotypeddict`.

22.4.1 Usage

.. autotypeddict::

Directive to automatically document a `typing.TypedDict`.

The output is based on the `autoclass` directive, but with a few differences:

- Private and Special members are always excluded.
- Undocumented members are always included.
- The default sort order is `bysource`.

The following options are available:

:noindex: (flag)

Do not generate index entries for the documented object (and all autodocumented members).

:alphabetical: (flag)

Sort the keys alphabetically. By default the keys are listed in the order they were defined.

:show-inheritance: (flag)

Inserts a list of base classes just below the `TypedDict`'s signature.

See <https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html> for further information.

:typeddict:

Role which provides a cross-reference to the documentation generated by `autotypeddict`.

Examples:

```
1  # Examples from
2  # https://www.python.org/dev/peps/pep-0589/#totality
3  # https://github.com/python/typing/pull/700
4  """
5  Demo of ``.. autotypeddict::``
6  """
7
8  # 3rd party
9  from typing_extensions import TypedDict
10
11  __all__ = ["Movie", "Animal", "OldStyleAnimal", "Cat", "Bird", "AquaticBird"]
12
13
14  class Movie(TypedDict):
15      """
16      Represents a movie.
17      """
18
19      #: The name of the movie.
20      name: str
21
22      #: The movie's release year.
23      year: int
24
25      based_on: str
26
27
28  class _Animal(TypedDict):
29      """
30      Keys required by all animals.
31      """
32
33      #: The name of the animal
34      name: str
35
36
37  class Animal(_Animal, total=False):
38      """
39      Optional keys common to all animals.
40      """
41
42      #: The animal's voice.
43      voice: str
44
45
46  #: Old style TypedDict for Python 2 and where keys aren't valid Python identifiers.
47  OldStyleAnimal = TypedDict(
48      "OldStyleAnimal", {
49          "animal-name": str,
50          "animal-voice": str,
51      }, total=False
```

(continues on next page)

(continued from previous page)

```

52         )
53
54
55 class Cat(Animal):
56     """
57     A cat.
58     """
59
60     #: The colour of the cat's fur.
61     fur_color: str
62
63
64 class Bird(Animal):
65     """
66     A bird.
67     """
68
69     #: The size of the bird's egg, in mm.
70     egg_size: float
71
72
73 class AquaticBird(Bird):
74
75     #: The bird's habitat (e.g. lake, sea)
76     habitat: float

```

```

.. automodule:: autotypeddict_demo
   :no-autosummary:
   :exclude-members: Movie,AquaticBird,OldStyleAnimal

.. autotypeddict:: autotypeddict_demo.Movie

```

This function takes a single argument, the `:typeddict:`~.Movie`` to watch.

Demo of `.. autotypeddict::`

typeddict Animal

Bases: `dict`

Optional keys common to all animals.

Required Keys

- **name** (`str`) – The name of the animal

Optional Keys

- **voice** (`str`) – The animal's voice.

typeddict Bird

Bases: `dict`

A bird.

Required Keys

- **egg_size** (`float`) – The size of the bird’s egg, in mm.
- **name** (`str`) – The name of the animal

Optional Keys

- **voice** (`str`) – The animal’s voice.

typeddict CatBases: `dict`

A cat.

Required Keys

- **fur_color** (`str`) – The colour of the cat’s fur.
- **name** (`str`) – The name of the animal

Optional Keys

- **voice** (`str`) – The animal’s voice.

typeddict MovieBases: `dict`

Represents a movie.

Required Keys

- **name** (`str`) – The name of the animal
- **year** (`int`) – The movie’s release year.
- **based_on** (`str`)

This function takes a single argument, the *Movie* to watch.

22.4.2 API Reference

Classes:

<code>TypedDictDocumenter(*args)</code>	Sphinx autodoc Documenter for documenting <code>typing.TypedDicts</code> .
---	--

Functions:

<code>setup(app)</code>	Setup <code>sphinx_toolbox.more_autodoc.autotypeddict</code> .
-------------------------	--

class TypedDictDocumenter(*args)Bases: `ClassDocumenter`Sphinx autodoc Documenter for documenting `typing.TypedDicts`.**Methods:**

<code>add_content(more_content[, no_docstring])</code>	Add the autodocumenter content.
--	---------------------------------

continues on next page

Table 10 – continued from previous page

<code>can_document_member</code> (member, membername, ...)	Called to see if a member can be documented by this documenter.
<code>document_keys</code> (keys, types, docstrings)	Document keys in a <code>typing.TypedDict</code> .
<code>document_members</code> ([all_members])	Generate reST for member documentation.
<code>filter_members</code> (members, want_all)	Filter the given member list.
<code>format_signature</code> (**kwargs)	Typed Dicts do not have a signature.
<code>sort_members</code> (documenters, order)	Sort the TypedDict's members.

add_content(*more_content*, *no_docstring=False*)

Add the autodocumenter content.

Parameters

- **more_content** (*Any*)
- **no_docstring** (*bool*) – Default `False`.

classmethod can_document_member(*member*, *membername*, *isattr*, *parent*)

Called to see if a member can be documented by this documenter.

Parameters

- **member** (*Any*) – The member being checked.
- **membername** (*str*) – The name of the member.
- **isattr** (*bool*)
- **parent** (*Any*) – The parent of the member.

Return type `bool`

document_keys(*keys*, *types*, *docstrings*)

Document keys in a `typing.TypedDict`.

Parameters

- **keys** (`List[str]`) – List of key names to document.
- **types** (`Dict[str, Type]`) – Mapping of key names to types.
- **docstrings** (`Dict[str, List[str]]`) – Mapping of key names to docstrings.

document_members(*all_members=False*)

Generate reST for member documentation. All members are always documented.

filter_members(*members*, *want_all*)

Filter the given member list.

Parameters

- **members** (`List[Tuple[str, Any]]`)
- **want_all** (*bool*)

Return type `List[Tuple[str, Any, bool]]`

format_signature(***kwargs*)
Typed Dicts do not have a signature.

Return type `str`

sort_members(*documenters, order*)
Sort the TypedDict's members.

Parameters

- **documenters** (`List[Tuple[Documenter, bool]]`)
- **order** (`str`)

Return type `List[Tuple[Documenter, bool]]`

setup(*app*)
Setup `sphinx_toolbox.more_autodoc.autotypeddict`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.5 more_autodoc.generic_bases

Modifies `sphinx.ext.autodoc.ClassDocumenter`'s `:show-inheritance:` option to show generic base classes.

This requires a relatively new version of the `typing` module that implements `__orig_bases__`.

New in version 1.5.0.

Enable `sphinx_toolbox.more_autodoc.generic_bases` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.more_autodoc.generic_bases',  
]
```

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

22.5.1 Configuration

generic_bases_fully_qualified

Type: `bool`

Required: `False`

Default: `False`

Determines whether the fully qualified name should be shown for bases.

If `False` (the default):


```

class Foo
    Bases: List[str]

```

If True:

```

class Foo
    Bases: typing.List[str]

```

Corresponds to the fully_qualified argument to `sphinx_toolbox.more_autodoc.typehints.format_annotation()`.

New in version 2.13.0.

22.5.2 Example

```

class Example(iterable=())
    Bases: List[Tuple[str, float, List[str]]]

```

An example of `sphinx_toolbox.more_autodoc.generic_bases`.

22.5.3 API Reference

```

class GenericBasesClassDocumenter(*args)
    Bases: PatchedAutoSummClassDocumenter

```

Class documenter that adds inheritance info, with support for generics.

```

add_directive_header(sig)
    Add the directive header.

```

Parameters **sig** (`str`)

```

setup(app)
    Setup sphinx_toolbox.more_autodoc.generic_bases.

```

New in version 1.5.0.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.6 more_autodoc.genericalias

Documenter for alias, which usually manifest as type aliases.

New in version 0.6.0.

Enable `sphinx_toolbox.more_autodoc.genericalias` by adding the following to the `extensions` variable in your `conf.py`:

```

extensions = [
    ...
    'sphinx_toolbox.more_autodoc.genericalias',
]

```

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

Note: `sphinx_toolbox.more_autodoc.genericalias` is only supported on Python 3.7 and above.

class `PrettyGenericAliasDocumenter`(*directive, name, indent=""*)

Bases: `DataDocumenter`

Specialized Documenter subclass for GenericAliases, with prettier output than Sphinx's one.

add_content(*more_content, no_docstring=False*)

Add the autodocumenter content.

Parameters

- **more_content** (*Any*)
- **no_docstring** (*bool*) – Default `False`.

add_directive_header(*sig*)

Add the directive header and options to the generated content.

classmethod **can_document_member**(*member, membername, isattr, parent*)

Called to see if a member can be documented by this documenter.

Return type `bool`

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.genericalias`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.7 more_autodoc.no_docstring

Adds the `:no-docstring:` option to automodule directives to exclude the docstring from the output.

New in version 1.0.0.

Enable `sphinx_toolbox.more_autodoc.no_docstring` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.no_docstring',
]
```

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

Functions:

<code>automodule_add_nodocstring(app)</code>	Add the <code>:no-docstring:</code> option to automodule directives to exclude the docstring from the output.
<code>no_docstring_process_docstring(app, what, ...)</code>	Process the docstring of a module, and remove its docstring of the <code>:no-docstring:</code> flag was set..
<code>setup(app)</code>	Setup <code>sphinx_toolbox.more_autodoc.no_docstring</code> .

automodule_add_nodocstring(*app*)

Add the `:no-docstring:` option to automodule directives to exclude the docstring from the output.

Parameters `app` – The Sphinx application.

Changed in version 1.0.0: Moved from `sphinx_toolbox.more_autodoc.__init__.py`

no_docstring_process_docstring(*app, what, name, obj, options, lines*)

Process the docstring of a module, and remove its docstring of the `:no-docstring:` flag was set..

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **what**
- **name** (`str`) – The name of the object being documented.
- **obj** – The object being documented.
- **options** – Mapping of autodoc options to values.
- **lines** (`List[str]`) – List of strings representing the current contents of the docstring.

Changed in version 1.0.0: Moved from `sphinx_toolbox.more_autodoc.__init__.py`

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.no_docstring`.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.8 more_autodoc.overloads

Documenters for functions and methods which display overloads differently.

New in version 1.4.0.

Enable `sphinx_toolbox.more_autodoc.overloads` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.overloads',
]
```

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

22.8.1 Configuration

overloads_location

Type: `str`

Default: `'signature'`

The location to display overloads at:

- `'signature'` – Display overloads above the function signature.
- `'top'` – Display overloads at the top of the docstring, immediately below the signature.
- `'bottom'` – Display overloads at the bottom of the docstring, or immediately below the return type.

22.8.2 API Reference

Classes:

<code>FunctionDocumenter</code> (directive, name[, indent])	Custom <code>sphinx.ext.autodoc.FunctionDocumenter</code> which renders overloads differently.
<code>MethodDocumenter</code> (directive, name[, indent])	Custom <code>sphinx.ext.autodoc.MethodDocumenter</code> which renders overloads differently.
<code>OverloadMixin</code> ()	Mixin class for function and class documenters that changes the appearance of overloaded functions.

Functions:

<code>setup(app)</code>	Setup <code>sphinx_toolbox.more_autodoc.overloads</code> .
-------------------------	--

class OverloadMixinBases: `object`

Mixin class for function and class documenters that changes the appearance of overloaded functions.

New in version 1.4.0.

Methods:

<code>create_body_overloads()</code>	Create the overloaded implementations for insertion into to the body of the documenter's output.
<code>process_overload_signature(overload)</code>	Processes the signature of the given overloaded implementation.
<code>add_content(more_content[, no_docstring])</code>	Add content from docstrings, attribute documentation and the user.

create_body_overloads()

Create the overloaded implementations for insertion into to the body of the documenter's output.

Return type `StringList`**process_overload_signature(overload)**

Processes the signature of the given overloaded implementation.

Parameters `overload` (`Signature`)**Return type** `Signature`**add_content(more_content, no_docstring=False)**

Add content from docstrings, attribute documentation and the user.

Parameters

- `more_content` (`Any`)
- `no_docstring` (`bool`) – Default `False`.

class FunctionDocumenter(directive, name, indent="")Bases: `OverloadMixin`, `FunctionDocumenter`Custom `sphinx.ext.autodoc.FunctionDocumenter` which renders overloads differently.

New in version 1.4.0.

Methods:

<code>format_signature(**kwargs)</code>	Format the function's signature, including those for any overloaded implementations.
<code>add_directive_header(sig)</code>	Add the directive's header.
<code>process_overload_signature(overload)</code>	Processes the signature of the given overloaded implementation.

format_signature(***kwargs*)

Format the function's signature, including those for any overloaded implementations.

Return type `str`**Returns** The signature(s), as a multi-line string.**add_directive_header**(*sig*)

Add the directive's header.

Parameters **sig** (`str`)**process_overload_signature**(*overload*)

Processes the signature of the given overloaded implementation.

Parameters **overload** (`Signature`)**Return type** `Signature`**class MethodDocumenter**(*directive, name, indent=""*)Bases: `OverloadMixin`, `MethodDocumenter`Custom `sphinx.ext.autodoc.MethodDocumenter` which renders overloads differently.

New in version 1.4.0.

Methods:

<code>format_signature</code> (<i>**kwargs</i>)	Format the method's signature, including those for any overloaded implementations.
<code>add_directive_header</code> (<i>sig</i>)	Add the directive's header.
<code>process_overload_signature</code> (<i>overload</i>)	Processes the signature of the given overloaded implementation.

format_signature(***kwargs*)

Format the method's signature, including those for any overloaded implementations.

Parameters **kwargs** (`Any`)**Return type** `str`**Returns** The signature(s), as a multi-line string.**add_directive_header**(*sig*)

Add the directive's header.

Parameters **sig** (`str`)**process_overload_signature**(*overload*)

Processes the signature of the given overloaded implementation.

Parameters **overload** (`Signature`)**Return type** `Signature`

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.overloads`.

New in version 1.4.0.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.9 more_autodoc.regex

Specialized Documenter for regular expression variables, similar to `autodata`.

New in version 1.2.0.

Enable `sphinx_toolbox.more_autodoc.regex` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.regex',
]
```

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

22.9.1 Usage

.. autoregex::

Directive to automatically document a regular expression variable.

The output is based on the `autodata` directive, and takes all of its options except `:annotation:`.

:no-value:

Don't show the value of the variable.

:value: value (string)

Show this instead of the value taken from the Python source code.

:no-type:

Don't show the type of the variable.

:no-flags:

Don't show the flags of the `Pattern` object.

:flags: flags (string)

Show this instead of the flags taken from the `Pattern` object. This should be correctly formatted for insertion into `reStructuredText`, such as `:py:data:`re.ASCII``.

Changed in version 2.7.0: The flags `re.DEBUG` and `re.VERBOSE` are now hidden as they don't affect the regex itself.

:regex:

Formats a regular expression with coloured output.

```
:regex: `^Hello\s+[Ww]orld[.,](Lovely|Horrible) weather, isn't it (.*)?`
```

```
^Hello\s+[Ww]orld[.,](Lovely|Horrible) weather, isn't it (.*)?
```

Changed in version 2.11.0: Now generates coloured output with the LaTeX builder.

22.9.2 API Reference

Classes:

<code>RegexDocumenter</code> (directive, name[, indent])	Specialized Documenter subclass for regex patterns.
<code>RegexParser</code> ()	Parser for regular expressions that outputs coloured output.
<code>TerminalRegexParser</code> ()	<code>RegexParser</code> that outputs ANSI coloured output for the terminal.
<code>HTMLRegexParser</code> ()	<code>RegexParser</code> that outputs styled HTML.
<code>LaTeXRegexParser</code> ()	<code>RegexParser</code> that outputs styled LaTeX.

Functions:

<code>parse_regex_flags</code> (flags)	Convert regex flags into “bitwise-or’d” Sphinx xrefs.
<code>no_formatting</code> (value)	No-op that returns the value as a string.
<code>span</code> (css_class)	Returns a function that wraps a value in a <code>span</code> tag with the given class.
<code>latex_textcolor</code> (colour_name)	Returns a function that wraps a value in a LaTeX <code>textcolor</code> command for the given colour.
<code>copy_asset_files</code> (app[, exception])	Copy additional stylesheets into the HTML build directory.
<code>setup</code> (app)	Setup <code>sphinx_toolbox.more_autodoc.regex</code> .

class `RegexDocumenter`(directive, name, indent=“”)

Bases: `VariableDocumenter`

Specialized Documenter subclass for regex patterns.

Methods:

<code>add_content</code> (more_content[, no_docstring])	Add content from docstrings, attribute documentation and the user.
<code>add_directive_header</code> (sig)	Add the directive’s header.
<code>can_document_member</code> (member, membername, ...)	Called to see if a member can be documented by this documenter.

add_content(*more_content*, *no_docstring=False*)

Add content from docstrings, attribute documentation and the user.

Parameters

- **more_content** (*Any*)
- **no_docstring** (*bool*) – Default `False`.

add_directive_header(*sig*)

Add the directive’s header.

Parameters **sig** (*str*)

classmethod can_document_member(*member*, *membername*, *isattr*, *parent*)

Called to see if a member can be documented by this documenter.

Parameters

- **member** (*Any*) – The member being checked.
- **membername** (*str*) – The name of the member.
- **isattr** (*bool*)
- **parent** (*Any*) – The parent of the member.

Return type *bool*

class RegexpParser

Bases: *object*

Parser for regular expressions that outputs coloured output.

The formatting is controlled by the following variables:

- **AT_COLOUR** – Used for e.g. `^A\bB\Z$`
- **SUBPATTERN_COLOUR** – Used for the parentheses around subpatterns, e.g. `(Hello) World`
- **IN_COLOUR** – Used for the square brackets around character sets, e.g. `[Hh]ello`
- **REPEAT_COLOUR** – Used for repeats, e.g. `A?B+C*D{2,4}E{5}`
- **REPEAT_BRACE_COLOUR** – Used for the braces around numerical repeats.
- **CATEGORY_COLOUR** – Used for categories, e.g. `\d\D\s\D\w\W`
- **BRANCH_COLOUR** – Used for branches, e.g. `(Lovely|Horrible) Weather`
- **LITERAL_COLOUR** – Used for literal characters.
- **ANY_COLOUR** – Used for the “any” dot.

These are all `Callable[[Any], str]`.

By default no formatting is performed.

Methods:

<code>parse_pattern</code> (<i>regex</i>)	Parse the given regular expression and return the formatted pattern.
---	--

parse_pattern(regex)

Parse the given regular expression and return the formatted pattern.

Parameters `regex` (`Pattern`)

Return type `str`

class TerminalRegexParser

Bases: `RegexParser`

`RegexParser` that outputs ANSI coloured output for the terminal.

The formatting is controlled by the following functions, which are instances of `consolekit.terminal_colours.Colour`:

- `AT_COLOUR` -> `YELLOW` – Used for e.g. `^\A\b\B\Z$`
- `SUBPATTERN_COLOUR` -> `LIGHTYELLOW_EX` – Used for the parentheses around subpatterns, e.g. `(Hello) World`
- `IN_COLOUR` -> `LIGHTRED_EX` – Used for the square brackets around character sets, e.g. `[Hh]ello`
- `REPEAT_COLOUR` -> `LIGHTBLUE_EX` – Used for repeats, e.g. `A?B+C*D{2,4}E{5}`
- `REPEAT_BRACE_COLOUR` -> `YELLOW` – Used for the braces around numerical repeats.
- `CATEGORY_COLOUR` -> `LIGHTYELLOW_EX` – Used for categories, e.g. `\d\D\s\D\w\W`
- `BRANCH_COLOUR` -> `YELLOW` – Used for branches, e.g. `(Lovely|Horrible) Weather`
- `LITERAL_COLOUR` -> `GREEN` – Used for literal characters.
- `ANY_COLOUR` -> `YELLOW` – Used for the “any” dot.

class HTMLRegexParser

Bases: `RegexParser`

`RegexParser` that outputs styled HTML.

The formatting is controlled by the following functions, which wrap the character in a `span` tag with an appropriate CSS class:

- `AT_COLOUR` -> `regex_at` – Used for e.g. `^\A\b\B\Z$`
- `SUBPATTERN_COLOUR` -> `regex_subpattern` – Used for the parentheses around subpatterns, e.g. `(Hello) World`
- `IN_COLOUR` -> `regex_in` – Used for the square brackets around character sets, e.g. `[Hh]ello`
- `REPEAT_COLOUR` -> `regex_repeat` – Used for repeats, e.g. `A?B+C*D{2,4}E{5}`
- `REPEAT_BRACE_COLOUR` -> `regex_repeat_brace` – Used for the braces around numerical repeats.
- `CATEGORY_COLOUR` -> `regex_category` – Used for categories, e.g. `\d\D\s\D\w\W`
- `BRANCH_COLOUR` -> `regex_branch` – Used for branches, e.g. `(Lovely|Horrible) Weather`
- `LITERAL_COLOUR` -> `regex_literal` – Used for literal characters.
- `ANY_COLOUR` -> `regex_any` – Used for the “any” dot.

Additionally, all `span` tags the `regex` class, and the surrounding code tag has the following classes: `docutils literal notranslate regex`.

class LaTeXRegexParserBases: *RegexParser**RegexParser* that outputs styled LaTeX.

The formatting is controlled by the following functions, which wrap the character in a LaTeX `textcolor` command for an appropriate colour:

- `AT_COLOUR` -> `regex_at` – Used for e.g. `^\A\b\B\Z$`
- `SUBPATTERN_COLOUR` -> `regex_subpattern` – Used for the parentheses around subpatterns, e.g. `(Hello) World`
- `IN_COLOUR` -> `regex_in` – Used for the square brackets around character sets, e.g. `[Hh]ello`
- `REPEAT_COLOUR` -> `regex_repeat` – Used for repeats, e.g. `A?B+C*D{2,4}E{5}`
- `REPEAT_BRACE_COLOUR` -> `regex_repeat_brace` – Used for the braces around numerical repeats.
- `CATEGORY_COLOUR` -> `regex_category` – Used for categories, e.g. `\d\D\s\D\w\W`
- `BRANCH_COLOUR` -> `regex_branch` – Used for branches, e.g. `(Lovely|Horrible) Weather`
- `LITERAL_COLOUR` -> `regex_literal` – Used for literal characters.
- `ANY_COLOUR` -> `regex_any` – Used for the “any” dot.

New in version 2.11.0.

parse_regex_flags(*flags*)

Convert regex flags into “bitwise-or’d” Sphinx xrefs.

Parameters `flags` (*int*)

Return type *str*

no_formatting(*value*)

No-op that returns the value as a string.

Used for unformatted output.

Return type *str*

span(*css_class*)

Returns a function that wraps a value in a `span` tag with the given class.

Parameters `css_class` (*str*)

Return type *Callable[[Any], str]*

latex_textcolor(*colour_name*)

Returns a function that wraps a value in a LaTeX `textcolor` command for the given colour.

New in version 2.11.0.

Parameters `colour_name` (*str*)

Return type *Callable[[Any], str]*

copy_asset_files(*app*, *exception=None*)

Copy additional stylesheets into the HTML build directory.

Parameters

- `app` (*Sphinx*) – The Sphinx application.

- **exception** (`Optional[Exception]`) – Any exception which occurred and caused Sphinx to abort. Default `None`.

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.regex`.

Parameters *app* (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.10 more_autodoc.sourcelink

Source code: `sphinx_toolbox/more_autodoc/sourcelink.py`

Show a link to the corresponding source code at the top of `automodule` output.

New in version 0.6.0.

Enable `sphinx_toolbox.more_autodoc.sourcelink` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.more_autodoc.sourcelink',  
]
```

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

22.10.1 Configuration

`sphinx_toolbox.more_autodoc.sourcelink` can be configured using the `autodoc_default_options` option in `conf.py`, or with the `:sourcelink:` option flag to `automodule`.

autodoc_show_sourcelink

Type: `bool`

Default: `False`

If `True`, shows a link to the corresponding source code at the top of each `automodule` directive.

:sourcelink:

When passed as an option flag to an `automodule` directive, show a link to the corresponding source code at the top of the output *for that module only*.

Changed in version 1.1.0: Added support for the `:sourcelink:` option flag to `automodule`.

22.10.2 API Reference

Functions:

<code>sourcelinks_process_docstring(app, what, ...)</code>	Process the docstring of a module and add a link to the source code if given in the configuration.
<code>setup(app)</code>	Setup <code>sphinx_toolbox.more_autodoc.sourcelink</code> .

`sourcelinks_process_docstring`(*app, what, name, obj, options, lines*)

Process the docstring of a module and add a link to the source code if given in the configuration.

Parameters

- **`app`** ([`Sphinx`](#)) – The Sphinx application.
- **`what`**
- **`name`** ([`str`](#)) – The name of the object being documented.
- **`obj`** – The object being documented.
- **`options`** ([`Mapping\[str, Any\]`](#)) – Mapping of autodoc options to values.
- **`lines`** ([`List\[str\]`](#)) – List of strings representing the current contents of the docstring.

`setup`(*app*)

Setup [`sphinx_toolbox.more_autodoc.sourcelink`](#).

Parameters **`app`** ([`Sphinx`](#)) – The Sphinx application.

Return type [`SphinxExtMetadata`](#)

22.11 `more_autodoc.typehints`

Enhanced version of [`sphinx-autodoc-typehints`](#).

Copyright (c) Alex Grönholm

The changes are:

- `None` is formatted as [`None`](#) and not `None`. If [`intersphinx`](#) is used this will now be a link to the Python documentation.

Since [agronholm/sphinx-autodoc-typehints#154](#) this feature is now available upstream.

- If the signature of the object cannot be read, the signature provided by Sphinx will be used rather than raising an error.

This usually occurs for methods of builtin types.

- [`typing.TypeVars`](#) are linked to if they have been included in the documentation.
- If a function/method argument has a [`module`](#), [`class`](#) or [`function`](#) object as its default value a better representation will be shown in the signature.

For example:

serialise(*obj*, *library*=<module 'json'>)

Serialise an object into a JSON string.

Parameters

- **obj** (*Any*) – The object to serialise.
- **library** – The JSON library to use.

Return type *str*

Returns The JSON string.

Previously this would have shown the full path to the source file. Now it displays <module 'json'>.

- The ability to hook into the `process_docstring()` function to edit the object’s properties before the annotations are added to the docstring.

This is used by `attr-utils` to add annotations based on converter functions in `attrs` classes.

To use this, in your extension’s `setup` function:

```
def setup(app: Sphinx) -> Dict[str, Any]:
    from sphinx_toolbox.more_autodoc.typehints import docstring_hooks
    docstring_hooks.append((my_hook, 75))
    return {}
```

`my_hook` is a function that takes the object being documented as its only argument and returns that object after modification. The 75 is the priority of the hook:

- < 20 runs before `fget` functions are extracted from properties
- < 90 runs before `__new__` functions are extracted from `NamedTuples`.
- < 100 runs before `__init__` functions are extracted from classes.

- Unresolved forward references are handled better.
- Many of the built in types from the `types` module are now formatted and linked to correctly.

New in version 0.4.0.

Changed in version 0.6.0: Moved from `sphinx_toolbox.autodoc.typehints`.

Changed in version 0.8.0: Added support for `collections.namedtuple()`.

Enable `sphinx_toolbox.more_autodoc.typehints` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.typehints',
]
```

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

For configuration information see <https://github.com/agronholm/sphinx-autodoc-typehints>

In addition, the following configuration value is added by this extension:

hide_none_rtypeType: `bool`Default: `False`Hides return types of `None`.**22.11.2 API Reference****Classes:**

<i>ObjectAlias</i> (name)	Used to represent a module, class, function etc in a Sphinx function/class signature.
<i>Module</i> (name)	Used to represent a module in a Sphinx function/class signature.
<i>Function</i> (name)	Used to represent a function in a Sphinx function/class signature.
<i>Class</i> (name)	Used to represent a class in a Sphinx function/class signature.

Functions:

<i>process_signature</i> (app, what, name, obj, ...)	Process the signature for a function/method.
<i>process_docstring</i> (app, what, name, obj, ...)	Process the docstring of a class, function, method etc.
<i>format_annotation</i> (annotation[, fully_qualified])	Format a type annotation.
<i>get_all_type_hints</i> (obj, name, original_obj)	Returns the resolved type hints for the given objects.
<i>setup</i> (app)	Setup <i>sphinx_toolbox.more_autodoc.typehints</i> .

Data:

<i>docstring_hooks</i>	List of additional hooks to run in <i>process_docstring()</i> .
<i>default_preprocessors</i>	A list of 2-element tuples, comprising a function to check the default value against and a preprocessor to pass the function to if True.
<i>Preprocessor</i>	Type hint for default preprocessor functions.

class ObjectAlias(name)Bases: `object`

Used to represent a module, class, function etc in a Sphinx function/class signature.

New in version 0.9.0.

Parameters `name` (`str`) – The name of the object being aliased.**__repr__()**Returns a string representation of the *ObjectAlias*.**Return type** `str`

class `Module(name)`

Bases: `ObjectAlias`

Used to represent a module in a Sphinx function/class signature.

Parameters `name (str)` – The name of the module.

class `Function(name)`

Bases: `ObjectAlias`

Used to represent a function in a Sphinx function/class signature.

New in version 0.9.0.

Parameters `name (str)` – The name of the function.

class `Class(name)`

Bases: `ObjectAlias`

Used to represent a class in a Sphinx function/class signature.

New in version 0.9.0.

Parameters `name (str)` – The name of the class.

process_signature(*app, what, name, obj, options, signature, return_annotation*)

Process the signature for a function/method.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **what** (`str`)
- **name** (`str`) – The name of the object being documented.
- **obj** – The object being documented.
- **options** – Mapping of autodoc options to values.
- **signature**
- **return_annotation** (`Any`)

Return type `Optional[Tuple[str, None]]`

Changed in version 0.8.0: Added support for factory function default values in attrs classes.

process_docstring(*app, what, name, obj, options, lines*)

Process the docstring of a class, function, method etc.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **what** (`str`)
- **name** (`str`) – The name of the object being documented.
- **obj** (`Any`) – The object being documented.
- **options** (`Dict[str, Any]`) – Mapping of autodoc options to values.
- **lines** (`List[str]`) – List of strings representing the current contents of the docstring.

Changed in version 1.1.0: An empty `:rtype:` flag can be used to control the position of the return type annotation in the docstring.

format_annotation(*annotation*, *fully_qualified=False*)

Format a type annotation.

Parameters

- **annotation**
- **fully_qualified** (*bool*) – Whether the fully qualified name should be shown (e.g. `typing.List`) or only the object name (e.g. `List`). Default `False`.

Return type `str`

Changed in version 2.13.0: Added support for `True` and `False`

get_all_type_hints(*obj*, *name*, *original_obj*)

Returns the resolved type hints for the given objects.

Parameters

- **obj**
- **name**
- **original_obj** – The original object, before the class if `obj` is its `__init__` method.

Preprocessor

Type hint for default preprocessor functions.

Alias of `Callable[[Type], Any]`

docstring_hooks

Type: `List[Tuple[Callable[[Any], Callable], int]]`

List of additional hooks to run in `process_docstring()`.

Each entry in the list consists of:

- a function that takes the object being documented as its only argument and returns that object after modification.
- a number giving the priority of the hook, in ascending order.
 - `< 20` runs before `fget` functions are extracted from properties
 - `< 90` runs before `__new__` functions are extracted from `NamedTuples`.
 - `< 100` runs before `__init__` functions are extracted from classes.

default_preprocessors

Type: `List[Tuple[Callable[[Type], bool], Callable[[Type], Any]]]`

A list of 2-element tuples, comprising a function to check the default value against and a preprocessor to pass the function to if `True`.

setup(*app*)

Setup `sphinx_toolbox.more_autodoc.typehints`.

Parameters **app** (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

22.12 more_autodoc.typevars

Documenter for module level `typing.TypeVar`'s, similar to Sphinx's `autotypevar` but with a different appearance.
New in version 1.3.0.

Enable `sphinx_toolbox.more_autodoc.typevars` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.more_autodoc.typevars',
]
```

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

22.12.1 Configuration

`all_typevars`

Type: `bool`

Default: `False`

Document all `typing.TypeVars`, even if they have no docstring.

`no_unbound_typevars`

Type: `bool`

Default: `True`

Only document `typing.TypeVars` that have a constraint or are bound.

This option has no effect if `all_typevars` is `False`.

22.12.2 Usage

`.. autotypevar::`

Directive to automatically document a `typing.TypeVar`.

The output is based on the `autodata` directive, and takes all of its options plus these additional ones:

`:no-value:`

Don't show the value of the variable.

`:value: value (string)`

Show this instead of the value taken from the Python source code.

`:no-type:`

Don't show the type of the variable.

22.12.3 API Reference

Classes:

<code>TypeVarDocumenter</code> (directive, name[, indent])	Alternative version of <code>sphinx.ext.autodoc.TypeVarDocumenter</code> with better type hint rendering.
--	---

Functions:

<code>unskip_typevars</code> (app, what, name, obj, skip, ...)	Unskip undocumented <code>typing.TypeVars</code> if <code>all_typevars</code> is <code>True</code> .
<code>setup</code> (app)	Setup <code>sphinx_toolbox.more_autodoc.typevars</code> .

class `TypeVarDocumenter`(directive, name, indent="")

Bases: `VariableDocumenter`

Alternative version of `sphinx.ext.autodoc.TypeVarDocumenter` with better type hint rendering.

Specialized Documenter subclass for `typing.TypeVars`.

Methods:

<code>can_document_member</code> (member, membername, ...)	Called to see if a member can be documented by this documenter.
<code>resolve_type</code> (forward_ref)	Resolve a <code>typing.ForwardRef</code> using the module the <code>TypeVar</code> belongs to.
<code>add_content</code> (more_content[, no_docstring])	Add content from docstrings, attribute documentation and user.
<code>add_directive_header</code> (sig)	Add the directive's header.
<code>get_doc</code> ([encoding, ignore])	Decode and return lines of the docstring(s) for the object.

classmethod `can_document_member`(member, membername, isattr, parent)

Called to see if a member can be documented by this documenter.

Parameters

- **member** (`Any`) – The member being checked.
- **membername** (`str`) – The name of the member.
- **isattr** (`bool`)
- **parent** (`Any`) – The parent of the member.

Return type `bool`

resolve_type(forward_ref)

Resolve a `typing.ForwardRef` using the module the `TypeVar` belongs to.

Parameters **forward_ref** (`ForwardRef`)

Return type `Type`

add_content(*more_content*, *no_docstring=False*)

Add content from docstrings, attribute documentation and user.

Parameters

- **more_content** (*Any*)
- **no_docstring** (*bool*) – Default `False`.

add_directive_header(*sig*)

Add the directive's header.

Parameters **sig** (*str*)

get_doc(*encoding=None*, *ignore=None*)

Decode and return lines of the docstring(s) for the object.

Parameters

- **encoding** (*Optional[str]*) – Default `None`.
- **ignore** (*Optional[int]*) – Default `None`.

Return type `List[List[str]]`

unskip_typevars(*app*, *what*, *name*, *obj*, *skip*, *options*)

Unskip undocumented `typing.TypeVars` if `all_typevars` is `True`.

Parameters

- **app** (*Sphinx*) – The Sphinx application.
- **what** (*str*) – The type of the object which the docstring belongs to (one of 'module', 'class', 'exception', 'function', 'method', 'attribute').
- **name** (*str*) – The fully qualified name of the object.
- **obj** (*Any*) – The object itself.
- **skip** (*bool*) – A boolean indicating if autodoc will skip this member if the user handler does not override the decision.
- **options** (*Dict[str, Any]*) – The options given to the directive: an object with attributes `inherited_members`, `undoc_members`, `show_inheritance` and `noindex` that are true if the flag option of same name was given to the auto directive.

Return type `Optional[bool]`

setup(*app*)
Setup `sphinx_toolbox.more_autodoc.typevars`.
Parameters `app` (`Sphinx`) – The Sphinx application.
Return type `SphinxExtMetadata`

22.13 more_autodoc.variables

Documenter for module level variables, similar to `autodata` but with a different appearance and more customisation options.

New in version 0.6.0.

Enable `sphinx_toolbox.more_autodoc.variables` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.more_autodoc.variables',  
]
```

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

Changed in version 0.7.0: Added `*AttributeDocumenters`

Changed in version 1.1.0: Added `SlotsAttributeDocumenter`

22.13.1 Usage

.. autovvariable::

Directive to automatically document a variable.

The output is based on the `autodata` directive, and takes all of its options, plus these additional ones:

:no-value:

Don't show the value of the variable.

:value: value (string)

Show this instead of the value taken from the Python source code.

:no-type:

Don't show the type of the variable.

:type: type (string)

Show this instead of the type taken from the Python source code.

An example of the output can be seen below for `type_template`.

22.13.2 API Reference

Classes:

<i>VariableDocumenter</i> (directive, name[, indent])	Specialized Documenter subclass for data items.
<i>TypedAttributeDocumenter</i> (directive, name[, ...])	Alternative version of <code>sphinx.ext.autodoc.AttributeDocumenter</code> with better type hint rendering.
<i>InstanceAttributeDocumenter</i> (directive, name)	Alternative version of <code>sphinx.ext.autodoc.InstanceAttributeDocumenter</code> with better type hint rendering.
<i>SlotsAttributeDocumenter</i> (directive, name[, ...])	Alternative version of <code>sphinx.ext.autodoc.InstanceAttributeDocumenter</code> with better type hint rendering.

Data:

<i>type_template</i>	Template for rendering type annotations in <i>VariableDocumenter</i> , <i>TypedAttributeDocumenter</i> and <i>InstanceAttributeDocumenter</i> .
----------------------	---

Functions:

<i>get_variable_type</i> (documenter)	Returns the formatted type annotation for a variable.
<i>setup</i> (app)	Setup <code>sphinx_toolbox.more_autodoc.variables</code> .

class `VariableDocumenter`(*directive*, *name*, *indent*="")

Bases: `DataDocumenter`

Specialized Documenter subclass for data items.

Methods:

<i>add_directive_header</i> (sig)	Add the directive's header.
-----------------------------------	-----------------------------

add_directive_header(sig)
Add the directive's header.

Parameters sig (str)

class `TypedAttributeDocumenter`(*directive*, *name*, *indent*="")

Bases: `DocstringStripSignatureMixin`, `ClassLevelDocumenter`

Alternative version of `sphinx.ext.autodoc.AttributeDocumenter` with better type hint rendering.

Specialized Documenter subclass for attributes.

New in version 0.7.0.

Changed in version 1.0.0: Now uses the type of the variable if it is not explicitly annotated.

Methods:

<code>can_document_member</code> (member, membername, ...)	Called to see if a member can be documented by this documenter.
<code>add_directive_header</code> (sig)	Add the directive's header.
<code>get_doc</code> ([encoding, ignore])	Decode and return lines of the docstring(s) for the object.
<code>add_content</code> (more_content[, no_docstring])	Add content from docstrings, attribute documentation and user.

classmethod `can_document_member`(*member, membername, isattr, parent*)
Called to see if a member can be documented by this documenter.

Return type `bool`

add_directive_header(*sig*)
Add the directive's header.

Parameters `sig` (`str`)

get_doc(*encoding=None, ignore=None*)
Decode and return lines of the docstring(s) for the object.

Parameters

- **encoding** (`Optional[str]`) – Default `None`.
- **ignore** (`Optional[int]`) – Default `None`.

Return type `List[List[str]]`

add_content(*more_content, no_docstring=False*)
Add content from docstrings, attribute documentation and user.

class `InstanceAttributeDocumenter`(*directive, name, indent=""*)

Bases: `TypedAttributeDocumenter`

Alternative version of `sphinx.ext.autodoc.InstanceAttributeDocumenter` with better type hint rendering.

Specialized Documenter subclass for attributes that cannot be imported because they are instance attributes (e.g. assigned in `__init__`).

New in version 0.7.0.

Changed in version 1.0.0: Now uses the type of the variable if it is not explicitly annotated.

Methods:

<code>can_document_member</code> (member, membername, ...)	Called to see if a member can be documented by this documenter.
<code>import_parent</code> ()	Import and return the attribute's parent.
<code>add_content</code> (more_content[, no_docstring])	Never try to get a docstring from the object.

classmethod `can_document_member`(*member, membername, isattr, parent*)
Called to see if a member can be documented by this documenter.

This documenter only documents `INSTANCEATTR` members.

Parameters

- **member** (*Any*) – The member being checked.
- **membername** (*str*) – The name of the member.
- **isattr** (*bool*)
- **parent** (*Any*) – The parent of the member.

Return type *bool*

import_parent()

Import and return the attribute's parent.

Return type *Any*

add_content(*more_content*, *no_docstring=False*)

Never try to get a docstring from the object.

class SlotsAttributeDocumenter(*directive*, *name*, *indent=""*)

Bases: *TypedAttributeDocumenter*

Alternative version of `sphinx.ext.autodoc.InstanceAttributeDocumenter` with better type hint rendering.

Specialized Documenter subclass for attributes that cannot be imported because they are attributes in `__slots__`.

New in version 1.1.0.

Methods:

<i>can_document_member</i> (<i>member</i> , <i>membername</i> , ...)	Called to see if a member can be documented by this documenter.
<i>get_doc</i> (<i>encoding</i> , <i>ignore</i>)	Decode and return lines of the docstring(s) for the object.

classmethod can_document_member(*member*, *membername*, *isattr*, *parent*)

Called to see if a member can be documented by this documenter.

This documenter only documents SLOTSATTR members.

Parameters

- **member** (*Any*) – The member being checked.
- **membername** (*str*) – The name of the member.
- **isattr** (*bool*)
- **parent** (*Any*) – The parent of the member.

Return type *bool*

get_doc(*encoding=None*, *ignore=None*)

Decode and return lines of the docstring(s) for the object.

Parameters

- **encoding** (*Optional[str]*) – Default *None*.
- **ignore** (*Optional[int]*) – Default *None*.

Return type *List[List[str]]*

```
type_template = ' **Type:** |nbsp| |nbsp| |nbsp| |nbsp| %s'
Type: str
```

Template for rendering type annotations in *VariableDocumenter*, *TypedAttributeDocumenter* and *InstanceAttributeDocumenter*.

Renders like:

```
    Type: str
```

get_variable_type(*documenter*)

Returns the formatted type annotation for a variable.

Parameters *documenter* (Documenter)

Return type str

setup(*app*)

Setup *sphinx_toolbox.more_autodoc.variables*.

Parameters *app* (Sphinx) – The Sphinx application.

Return type *SphinxExtMetadata*

more_autosummary

Extensions to `sphinx.ext.autosummary`.

Provides an enhanced version of <https://autodocsumm.readthedocs.io/> which respects the `autodoc member-order` option. This can be given for an individual directive, in the `autodoc_member_order` configuration value, or via `autodocsumm_member_order`.

Also patches `sphinx.ext.autosummary.Autosummary` to fix an issue where the module name is sometimes duplicated. I.e. `foo.bar.baz()` became `foo.bar.foo.bar.baz()`, which of course doesn't exist and created a broken link.

New in version 0.7.0.

Changed in version 1.3.0: Autosummary now selects the appropriate documenter for attributes rather than falling back to `DataDocumenter`.

Changed in version 2.13.0: Also patches `sphinx.ext.autodoc.ModuleDocumenter` to fix an issue where `__all__` is not respected for autosummary tables.

23.1 Configuration

`autodocsumm_member_order`

Type: `str`

Default: `'alphabetical'`

Determines the sort order of members in `autodocsumm` summary tables. Valid values are `'alphabetical'` and `'bysource'`.

Note that for `'bysource'` the module must be a Python module with the source code available.

The member order can also be set on a per-directive basis using the `:member-order: [order]` option. This applies not only to `automodule` etc. directives, but also to `automodulesumm` etc. directives.

`autosummary_col_type`

Type: `str`

Default: `'\X'`

The LaTeX column type to use for autosummary tables.

Custom columns can be defined in the LaTeX preamble for use with this option.

For example:

```
latex_elements["preamble"] = r'''
    \makeatletter
    \newcolumntype{\Xx}[2]{>\raggedright\arraybackslash}p{\dimexpr
        (\linewidth-\arrayrulewidth)*#1/#2-\tw@ \tabcolsep-\arrayrulewidth\relax}}
    \makeatother
```

(continues on next page)

(continued from previous page)

```
'''  
autosummary_col_type = "\\Xx"
```

New in version 2.13.0.

23.2 API Reference

Classes:

<code>PatchedAutoSummClassDocumenter(*args)</code>	Patched version of <code>autodocsumm.AutoSummClassDocumenter</code> which doesn't show summary tables for aliased objects.
<code>PatchedAutoSummModuleDocumenter(*args)</code>	Patched version of <code>autodocsumm.AutoSummClassDocumenter</code> which works around a bug in Sphinx 3.4 and above where <code>__all__</code> is not respected.
<code>PatchedAutosummary(name, arguments, options, ...)</code>	Pretty table containing short signatures and summaries of functions etc.

Functions:

<code>get_documenter(app, obj, parent)</code>	Returns an <code>autodoc.Documenter</code> class suitable for documenting the given object.
<code>setup(app)</code>	Setup <code>sphinx-toolbox.more_autosummary</code> .

class `PatchedAutoSummClassDocumenter(*args)`

Bases: `AutoSummClassDocumenter`

Patched version of `autodocsumm.AutoSummClassDocumenter` which doesn't show summary tables for aliased objects.

New in version 0.9.0.

Methods:

<code>add_content(*args, **kwargs)</code>	Add content from docstrings, attribute documentation and user.
---	--

`add_content(*args, **kwargs)`

Add content from docstrings, attribute documentation and user.

class `PatchedAutoSummModuleDocumenter(*args)`

Bases: `AutoSummModuleDocumenter`

Patched version of `autodocsumm.AutoSummClassDocumenter` which works around a bug in Sphinx 3.4 and above where `__all__` is not respected.

New in version 2.13.0.

class `PatchedAutosummary(name, arguments, options, content, lineno, content_offset, block_text, state, state_machine)`

Bases: `Autosummary`

Pretty table containing short signatures and summaries of functions etc.

Patched version of `sphinx.ext.autosummary.Autosummary` to fix an issue where the module name is sometimes duplicated.

I.e. `foo.bar.baz()` became `foo.bar.foo.bar.baz()`, which of course doesn't exist and created a broken link.

New in version 0.5.1.

Changed in version 0.7.0: Moved from `sphinx_toolbox.patched_autosummary`.

Changed in version 2.13.0: Added support for customising the column type with the `autosummary_col_type` option.

Methods:

<code>create_documenter</code> (app, obj, parent, full_name)	Get an <code>autodoc.Documenter</code> class suitable for documenting the given object.
<code>get_table</code> (items)	Generate a list of table nodes for the <code>autosummary</code> directive.
<code>import_by_name</code> (name, prefixes)	Import the object with the give name.

create_documenter(app, obj, parent, full_name)

Get an `autodoc.Documenter` class suitable for documenting the given object.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **obj** (`Any`) – The object being documented.
- **parent** (`Any`) – The parent of the object (e.g. a module or a class).
- **full_name** (`str`) – The full name of the object.

Changed in version 1.3.0: Now selects the appropriate documenter for attributes rather than falling back to `DataDocumenter`.

Return type `Documenter`

get_table(*items*)

Generate a list of table nodes for the `autosummary` directive.

Parameters `items` (`List[Tuple[str, str, str, str]]`) – A list produced by `self.get_items`.

Return type `List[Node]`

import_by_name(*name*, *prefixes*)

Import the object with the give name.

Parameters

- **name** (`str`)
- **prefixes** (`List[str]`)

Return type `Tuple[str, Any, Any, str]`

Returns The real name of the object, the object, the parent of the object, and the name of the module.

get_documenter(*app*, *obj*, *parent*)

Returns an `autodoc.Documenter` class suitable for documenting the given object.

New in version 1.3.0.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **obj** (`Any`) – The object being documented.
- **parent** (`Any`) – The parent of the object (e.g. a module or a class).

Return type `Type[Documenter]`

setup(*app*)

Setup `sphinx_toolbox.more_autosummary`.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

The following tweaks are available:

24.1 `tweaks.footnote_symbols`

Tweak which monkeypatches docutils to use the following symbols for footnotes:

- † – dagger
- ‡ – double dagger
- § – section mark
- ¶ – paragraph mark (pilcrow)
- # – number sign
- ♠ – spade suit
- ♥ – heart suit
- ♦ – diamond suit
- ♣ – club suit

With some themes the superscript asterisk becomes very hard to see.

New in version 2.7.0.

Enable `sphinx_toolbox.tweaks.footnote_symbols` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.tweaks.footnote_symbols',  
]
```

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

setup(*app*)

Setup `sphinx_toolbox.tweaks.footnote_symbols`.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

24.2 tweaks.latex_layout

Makes minor adjustments to the LaTeX layout.

- Increases the whitespace above function signatures by 5px, to prevent the function visually merging with the previous one.
- Remove unnecessary indentation and allow “raggedright” for the fields in the body of functions, which prevents ugly whitespace and line breaks.
- Disables justification for function signatures. This is a backport of changes from Sphinx 4 added in [sphinx-doc/sphinx#8997](#).

New in version 2.12.0.

- With Sphinx 3.5, doesn’t add `\sphinxAtStartPar` before every paragraph. The change in [sphinx-doc/sphinx#8781](#) was to solve an issue with *tables*, but it isn’t clear why it then gets added for *every* paragraph so this extension removes it.

New in version 2.13.0.

- Configures hyperref to apply correct page numbering to the frontmatter.

New in version 2.14.0.

New in version 2.10.0.

Enable `sphinx_toolbox.tweaks.latex_layout` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.tweaks.latex_layout',  
]
```

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

setup(*app*)

Setup `sphinx_toolbox.tweaks.latex_layout`.

Parameters `app` (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

24.3 tweaks.latex_toc

Adjusts the default LaTeX output as follows:

- The captions from `toctree` directives are converted into document parts.
- The PDF outline has the correct hierarchy, including having the indices as top-level elements.

New in version 2.1.0.

Enable `sphinx_toolbox.tweaks.latex_toc` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_toolbox.tweaks.latex_toc',  
]
```

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

Functions:

<code>configure(app, config)</code>	Configure <code>sphinx_toolbox.tweaks.latex_toc</code> .
<code>setup(app)</code>	Setup <code>sphinx_toolbox.tweaks.latex_toc</code> .

configure(*app*, *config*)Configure `sphinx_toolbox.tweaks.latex_toc`.**Parameters**

- **app** (`Sphinx`) – The Sphinx application.
- **config** (`Config`)

setup(*app*)Setup `sphinx_toolbox.tweaks.latex_toc`.**Parameters** **app** (`Sphinx`) – The Sphinx application.**Return type** `SphinxExtMetadata`

24.4 tweaks.param_dash

Monkeypatches `sphinx.util.docfields.TypedField` to only output the endash (–) separating the parameter name from its description if a description was given.

New in version 0.9.0.

Example

```
.. class:: MyClass(foo, bar)

    This is my class.

:param foo: An argument
:param bar:
```

class `MyClass`(*foo*, *bar*)

This is my class.

Parameters

- **foo** – An argument
- **bar**

Enable `sphinx_toolbox.tweaks.param_dash` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.tweaks.param_dash',
]
```

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

setup(*app*)

Setup `sphinx_toolbox.tweaks.param_dash`.

Parameters *app* (`Sphinx`) – The Sphinx application.

Return type `SphinxExtMetadata`

24.5 tweaks.sphinx_panels_tabs

Tweak to `executablebooks/sphinx-tabs` to fix a CSS conflict with `executablebooks/sphinx-panels`.

Fix for `executablebooks/sphinx-panels#51`.

New in version 1.9.0.

Enable `sphinx_toolbox.tweaks.sphinx_panels_tabs` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [
    ...
    'sphinx_toolbox.tweaks.sphinx_panels_tabs',
]
```

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

Functions:

<code>copy_asset_files</code> (<i>app</i> [, <i>exception</i>])	Copy asset files to the output.
<code>setup</code> (<i>app</i>)	Setup <code>sphinx_toolbox.tweaks.sphinx_panels_tabs</code> .

copy_asset_files(*app*, *exception*=`None`)

Copy asset files to the output.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **exception** (`Optional[Exception]`) – Any exception which occurred and caused Sphinx to abort. Default `None`.

Changed in version 2.7.0: Renamed from `copy_assets`. The old name is deprecated and will be removed in 3.0.0

setup(*app*)

Setup `sphinx_toolbox.tweaks.sphinx_panels_tabs`.

Parameters *app* (`Sphinx`) – The Sphinx application.

Return type *SphinxExtMetadata*

24.6 tweaks.tabsize

Hack to get the docutils tab size, as there doesn't appear to be any other way.

New in version 1.0.0.

You probably don't need to use this extension directly, but if you're developing an extension of your own you can enable it like so:

```
def setup(app: Sphinx) -> Dict[str, Any]:
    app.setup_extension('sphinx_toolbox.github')
    return {}
```

This will guarantee that the following value will be available via `app.config`:

- **docutils_tab_width** (`int`) – The number of spaces that correspond to a tab when Docutils parses source files.

setup(*app*)

Setup *sphinx_toolbox.tweaks.tabsize*.

Parameters *app* (`Sphinx`) – The Sphinx application.

Return type *SphinxExtMetadata*

Changelog

25.1 2.14.1

25.1.1 Bugs Fixed

- Correctly uses UTF-8 when reading files in HTML regression tests.

25.2 2.14.0

25.2.1 Features

- `sphinx_toolbox.testing.HTMLRegressionFixture` – Added support for rendering the reference file as a jinja2 template, which can be used to account for differences between Python and Sphinx versions.
- Added support for Sphinx 4.0 and sphinx-tabs versions up to 3.2.0
- Improved support for Python 3.10.0 rc.1
- `sphinx_toolbox.tweaks.latex_layout` – Now configures `hyperref` to use correct page numbering for the frontmatter.

25.2.2 Bugs Fixed

- URLs pointing to <https://pypistats.org/> now use lowercased project names.
- The `cls` and `return` attributes are ignored from `__annotations__` when deciding whether to include the `__new__` method for a `NamedTuple` with `NamedTupleDocumenter`.

25.3 2.13.0

25.3.1 Features

- Added support for Sphinx 3.4.x and 3.5.x.
- `sphinx_toolbox.more_autodoc.autoprotocol` – Added support for generic bases, such as `class SupportsAbs(Protocol[T_co]): ...`
- `sphinx_toolbox.more_autosummary` – Added the `autosummary_col_type` configuration option.
- `sphinx_toolbox.latex.replace_unknown_unicode()` – Add support for converting `≥` and `≤`.
- `sphinx_toolbox.more_autodoc.typehints.format_annotation()` – Added support for `True` and `False`

25.3.2 Bugs Fixed

- *sphinx_toolbox.more_autosummary* – Ensure `__all__` is respected for autosummary tables.
-

Note: The changelog prior to 2.13.0 has not been compiled yet.

Part II

Developer API

sphinx_toolbox

sphinx_toolbox also provides some utility functions for building Sphinx extensions.

setup(*app*)

Setup *sphinx_toolbox*.

Parameters *app* (*Sphinx*) – The Sphinx application.

Return type *SphinxExtMetadata*

sphinx_toolbox.config

Internal configuration for sphinx-toolbox.

Exceptions:

<i>InvalidOptionError</i>	Subclass of <code>ValueError</code> to indicate an invalid value for a configuration value.
<i>MissingOptionError</i>	Subclass of <code>ValueError</code> to indicate a missing configuration option.

Classes:

<i>ToolboxConfig</i> ([config, overrides])	Subclass of <code>sphinx.config.Config</code> with type annotations for the configuration values added by sphinx-toolbox.
--	---

Functions:

<i>validate_config</i> (app, config)	Validate the provided configuration values.
--------------------------------------	---

exception InvalidOptionError

Bases: `ValueError`

Subclass of `ValueError` to indicate an invalid value for a configuration value.

exception MissingOptionError

Bases: `ValueError`

Subclass of `ValueError` to indicate a missing configuration option.

class ToolboxConfig(config={}, overrides={})

Bases: `Config`

Subclass of `sphinx.config.Config` with type annotations for the configuration values added by sphinx-toolbox.

Depending on the extensions enabled not all of these configuration values will be present.

Functionally compatible with `sphinx.config.Config`.

Attributes:

<i>all_typevars</i>	Document all <code>typing.TypeVars</code> , even if they have no docstring.
<i>assets_dir</i>	The directory in which to find assets for the <code>asset</code> role.

continues on next page

Table 4 – continued from previous page

<i>conda_channels</i>	List of required Conda channels.
<i>docutils_tab_width</i>	The tab size used by docutils.
<i>github_issues_url</i>	The base URL for the issues on GitHub.
<i>github_pull_url</i>	The base URL for the pull requests on GitHub.
<i>github_repository</i>	The GitHub repository this documentation corresponds to.
<i>github_source_url</i>	The base URL for the source code on GitHub.
<i>github_url</i>	The complete URL of the repository on GitHub.
<i>github_username</i>	The username of the GitHub account that owns the repository this documentation corresponds to.
<i>no_unbound_typevars</i>	Only document <code>typing.TypeVars</code> that have a constraint of are bound.
<i>rst_prolog</i>	A string of reStructuredText that will be included at the beginning of every source file that is read.
<i>source_link_target</i>	The target of the source link, either 'github' or 'sphinx'.
<i>wikipedia_lang</i>	The Wikipedia language to use for <i>wikipedia</i> roles.

all_typevars**Type:** `bool`Document all `typing.TypeVars`, even if they have no docstring.**assets_dir****Type:** `str`The directory in which to find assets for the *asset* role.**conda_channels****Type:** `List[str]`

List of required Conda channels.

docutils_tab_width**Type:** `int`The tab size used by docutils. This is usually 8 spaces, but can be configured in the `docutils.conf` file.**github_issues_url****Type:** `RequestsURL`

The base URL for the issues on GitHub.

github_pull_url**Type:** `RequestsURL`

The base URL for the pull requests on GitHub.

github_repository**Type:** `str`

The GitHub repository this documentation corresponds to.

github_source_url**Type:** `RequestsURL`

The base URL for the source code on GitHub.

github_url**Type:** `RequestsURL`

The complete URL of the repository on GitHub.

github_username**Type:** `str`

The username of the GitHub account that owns the repository this documentation corresponds to.

no_unbound_typevars**Type:** `bool`

Only document `typing.TypeVars` that have a constraint of are bound.

This option has no effect if `all_typevars` is False.

rst_prolog**Type:** `str`

A string of reStructuredText that will be included at the beginning of every source file that is read.

source_link_target**Type:** `str`

The target of the source link, either 'github' or 'sphinx'. Will be lowercase after `validate_config()` has run.

wikipedia_lang**Type:** `str`

The Wikipedia language to use for `wikipedia` roles.

validate_config(*app*, *config*)

Validate the provided configuration values.

See `ToolboxConfig` for a list of the configuration values.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **config** (`Config`)

sphinx_toolbox.testing

Functions for testing Sphinx extensions.

Attention: This module has the following additional requirements:

```
coincidence>=0.4.3
pygments>=2.7.4
```

These can be installed as follows:

```
$ python -m pip install sphinx-toolbox[testing]
```

See also: Sphinx’s own testing library: <https://github.com/sphinx-doc/sphinx/tree/3.x/sphinx/testing>

Classes:

<code>Sphinx()</code>	A class that pretends to be <code>sphinx.application.Sphinx</code> but that is stripped back to allow the internals to be inspected.
<code>RunSetupOutput(setup_ret, directives, roles, ...)</code>	<code>NamedTuple</code> representing the output from <code>run_setup()</code> .
<code>HTMLRegressionFixture(datadir, ...)</code>	Subclass of <code>pytest_regressions.file_regression.FileRegressionFixture</code> for checking HTML files.

Functions:

<code>run_setup(setup_func)</code>	Function for running an extension’s <code>setup()</code> function for testing.
<code>remove_html_footer(page)</code>	Remove the Sphinx footer from HTML pages.
<code>check_html_regression(page, file_regression)</code>	Check an HTML page generated by Sphinx for regressions, using <code>pytest-regressions</code> .
<code>remove_html_link_tags(page)</code>	Remove link tags from HTML pages.
<code>check_asset_copy(func, *asset_files, ...)</code>	Helper to test functions which respond to Sphinx <code>build-finished</code> events and copy asset files.
<code>html_regression(datadir, original_datadir, ...)</code>	Returns an <code>HTMLRegressionFixture</code> scoped to the test function.

class SphinxBases: `object`

A class that pretends to be `sphinx.application.Sphinx` but that is stripped back to allow the internals to be inspected. This can be used in tests to ensure the nodes, roles etc. being registered in an extension's `setup()` function are actually being registered.

Attributes:

<code>registry</code>	Instance of <code>sphinx.registry.SphinxComponentRegistry</code>
<code>config</code>	Instance of <code>sphinx.config.Config</code>
<code>events</code>	Instance of <code>sphinx.events.EventManager</code>
<code>html_themes</code>	Mapping of HTML theme names to filesystem paths.

Methods:

<code>add_builder(builder[, override])</code>	Register a new builder.
<code>add_config_value(name, default, rebuild[, types])</code>	Register a configuration value.
<code>add_event(name)</code>	Register an event called <code>name</code> .
<code>set_translator(name, translator_class[, ...])</code>	Register or override a Docutils translator class.
<code>add_node(node[, override])</code>	Register a Docutils node class.
<code>add_enumerable_node(node, figtype[, ...])</code>	Register a Docutils node class as a numfig target.
<code>add_directive(name, cls[, override])</code>	Register a Docutils directive.
<code>add_role(name, role[, override])</code>	Register a Docutils role.
<code>add_generic_role(name, nodeclass[, override])</code>	Register a generic Docutils role.
<code>add_domain(domain[, override])</code>	Register a domain.
<code>add_directive_to_domain(domain, name, cls[, ...])</code>	Register a Docutils directive in a domain.
<code>add_role_to_domain(domain, name, role[, ...])</code>	Register a Docutils role in a domain.
<code>add_index_to_domain(domain, index[, override])</code>	Register a custom index for a domain.
<code>add_object_type(directivename, rolename[, ...])</code>	Register a new object type.
<code>add_crossref_type(directivename, rolename[, ...])</code>	Register a new crossref object type.
<code>add_transform(transform)</code>	Register a Docutils transform to be applied after parsing.
<code>add_post_transform(transform)</code>	Register a Docutils transform to be applied before writing.
<code>add_js_file(filename, **kwargs)</code>	Register a JavaScript file to include in the HTML output.
<code>add_css_file(filename, **kwargs)</code>	Register a stylesheet to include in the HTML output.
<code>add_latex_package(packagename[, options, ...])</code>	Register a package to include in the LaTeX source code.
<code>add_lexer(alias, lexer)</code>	Register a new lexer for source code.
<code>add_autodocenter(cls[, override])</code>	Register a new documenter class for the autodoc extension.
<code>add_autodoc_attrgetter(typ, getter)</code>	Register a new <code>getattr</code> -like function for the autodoc extension.
<code>add_source_suffix(suffix, filetype[, override])</code>	Register a suffix of source files.
<code>add_source_parser(*args, **kwargs)</code>	Register a parser class.
<code>add_env_collector(collector)</code>	No-op for now.
<code>add_html_theme(name, theme_path)</code>	Register an HTML Theme.

continues on next page

Table 4 – continued from previous page

<code>add_html_math_renderer(name[, ...])</code>	Register a math renderer for HTML.
<code>setup_extension(extname)</code>	Import and setup a Sphinx extension module.
<code>require_sphinx(version)</code>	Check the Sphinx version if requested.
<code>connect(event, callback[, priority])</code>	Register <i>callback</i> to be called when <i>event</i> is emitted.

registry**Type:** `SphinxComponentRegistry`Instance of `sphinx.registry.SphinxComponentRegistry`**config****Type:** `Config`Instance of `sphinx.config.Config`**events****Type:** `EventManager`Instance of `sphinx.events.EventManager`**html_themes****Type:** `Dict[str, str]`

Mapping of HTML theme names to filesystem paths.

add_builder(*builder*, *override=False*)

Register a new builder.

The registered values are stored in the `app.registry.builders` dictionary (`typing.Dict[str, typing.Type[sphinx.builders.Builder]]`).**add_config_value**(*name*, *default*, *rebuild*, *types=()*)

Register a configuration value.

The registered values are stored in the `app.config.values` dictionary (`typing.Dict[str, typing.Tuple]`).**add_event**(*name*)Register an event called *name*.The registered values are stored in the `app.events.events` dictionary (`typing.Dict[str, str]`).**set_translator**(*name*, *translator_class*, *override=False*)

Register or override a Docutils translator class.

The registered values are stored in the `app.registry.translators` dictionary. (`typing.Dict[str, typing.Type[docutils.nodes.NodeVisitor]]`).

add_node(*node*, *override=False*, ***kwargs*)

Register a Docutils node class.

The registered values are stored in the `additional_nodes` set returned by `run_setup()` (`typing.Set[typing.Type[docutils.nodes.Node]]`).

add_enumerable_node(*node*, *figtype*, *title_getter=None*, *override=False*, ***kwargs*)

Register a Docutils node class as a numfig target.

add_directive(*name*, *cls*, *override=False*)

Register a Docutils directive.

add_role(*name*, *role*, *override=False*)

Register a Docutils role.

The registered values are stored in the `roles` dictionary returned by `run_setup()`. (`typing.Dict[str, typing.Callable]`).

add_generic_role(*name*, *nodeclass*, *override=False*)

Register a generic Docutils role.

add_domain(*domain*, *override=False*)

Register a domain.

add_directive_to_domain(*domain*, *name*, *cls*, *override=False*)

Register a Docutils directive in a domain.

add_role_to_domain(*domain*, *name*, *role*, *override=False*)

Register a Docutils role in a domain.

add_index_to_domain(*domain*, *index*, *override=False*)

Register a custom index for a domain.

add_object_type(*directivename*, *rolename*, *indextemplate=""*, *parse_node=None*, *ref_nodeclass=None*, *objname=""*, *doc_field_types=[]*, *override=False*)

Register a new object type.

add_crossref_type(*directivename*, *rolename*, *indextemplate=""*, *ref_nodeclass=None*, *objname=""*, *override=False*)

Register a new crossref object type.

add_transform(*transform*)

Register a Docutils transform to be applied after parsing.

add_post_transform(*transform*)

Register a Docutils transform to be applied before writing.

add_js_file(*filename*, ***kwargs*)

Register a JavaScript file to include in the HTML output.

New in version 2.8.0.

add_css_file(*filename*, ***kwargs*)

Register a stylesheet to include in the HTML output.

New in version 2.7.0.

add_latex_package(*packagename*, *options=None*, *after_hyperref=False*)

Register a package to include in the LaTeX source code.

add_lexer(*alias*, *lexer*)

Register a new lexer for source code.

add_autodocumenter(*cls*, *override=False*)

Register a new documenter class for the autodoc extension.

add_autodoc_attrgetter(*typ*, *getter*)

Register a new `getattr`-like function for the autodoc extension.

add_source_suffix(*suffix*, *filetype*, *override=False*)

Register a suffix of source files.

add_source_parser(**args*, ***kwargs*)

Register a parser class.

add_env_collector(*collector*)

No-op for now.

add_html_theme(*name*, *theme_path*)

Register an HTML Theme.

add_html_math_renderer(*name*, *inline_renderers=None*, *block_renderers=None*)

Register a math renderer for HTML.

setup_extension(*extname*)

Import and setup a Sphinx extension module.

require_sphinx(*version*)

Check the Sphinx version if requested.

No-op when testing

connect(*event*, *callback*, *priority=500*)

Register *callback* to be called when *event* is emitted.

Return type `int`

run_setup(*setup_func*)

Function for running an extension's `setup()` function for testing.

Parameters **setup_func** (`Union[Callable[[Sphinx], Optional[Dict[str, Any]]], Callable[[Sphinx], Optional[SphinxExtMetadata]], Callable[[Sphinx], Optional[Dict[str, Any]]], Callable[[Sphinx], Optional[SphinxExtMetadata]]]`) – The `setup()` function under test.

Return type `RunSetupOutput`

Returns 5-element namedtuple

namedtuple `RunSetupOutput`(*setup_ret*, *directives*, *roles*, *additional_nodes*, *app*)

Bases: `NamedTuple`

`NamedTuple` representing the output from `run_setup()`.

Fields

- 0) **setup_ret** (`Union[None, Dict[str, Any], SphinxExtMetadata]`) – The output from the `setup()` function.
- 1) **directives** (`Dict[str, Callable]`) – Mapping of directive names to directive functions.
- 2) **roles** (`Dict[str, Callable]`) – Mapping of role names to role functions.
- 3) **additional_nodes** (`Set[Type[Any]]`) – Set of custom docutils nodes registered in `setup()`.
- 4) **app** (`Sphinx`) – Instance of `sphinx_toolbox.testing.Sphinx`.

`__repr__()`

Return a string representation of the `RunSetupOutput`.

Return type `str`

remove_html_footer(*page*)

Remove the Sphinx footer from HTML pages.

The footer contains the Sphinx and theme versions and therefore changes between versions. This can cause unwanted, false positive test failures.

Parameters *page* (`BeautifulSoup`) – The page to remove the footer from.

Return type `BeautifulSoup`

Returns The page without the footer.

check_html_regression(*page*, *file_regression*)

Check an HTML page generated by Sphinx for regressions, using `pytest-regressions`.

Parameters

- **page** (`BeautifulSoup`) – The page to test.
- **file_regression** (`FileRegressionFixture`) – The file regression fixture.

Example usage

```
@pytest.mark.parametrize("page", ["index.html"], indirect=True)
def test_page(page: BeautifulSoup, file_regression: FileRegressionFixture):
    check_html_regression(page, file_regression)
```

remove_html_link_tags(*page*)

Remove link tags from HTML pages.

These may vary between different versions of Sphinx and its extensions. This can cause unwanted, false positive test failures.

Parameters *page* (`BeautifulSoup`) – The page to remove the link tags from.

Return type `BeautifulSoup`

Returns The page without the link tags.

check_asset_copy(*func*, **asset_files*, *file_regression*)

Helper to test functions which respond to Sphinx build-finished events and copy asset files.

New in version 2.0.0.

Parameters

- **func** (Callable[[Sphinx, Exception], Any]) – The function to test.
- ***asset_files** – The paths of asset files copied by the function, relative to the Sphinx output directory.
- **file_regression** (FileRegressionFixture)

class HTMLRegressionFixture(*datadir*, *original_datadir*, *request*)

Bases: FileRegressionFixture

Subclass of `pytest_regressions.file_regression.FileRegressionFixture` for checking HTML files.

New in version 2.0.0.

Methods:

<code>check</code> (<i>page</i> , *[, <i>extension</i> , <i>jinja2</i>])	Check an HTML page generated by Sphinx for regressions, using <code>pytest-regressions</code> .
--	---

check(*page*, *, *extension*='html', *jinja2*=False, ***kwargs*)

Check an HTML page generated by Sphinx for regressions, using `pytest-regressions`.

Parameters

- **page** (BeautifulSoup) – The page to test.
- **jinja2** (bool) – Whether to render the reference file as a jinja2 template. Default `False`.
- ****kwargs** – Additional keyword arguments passed to `pytest_regressions.file_regression.FileRegressionFixture.check()`.

Changed in version 2.14.0: Added the `jinja2` keyword argument.

When `jinja2` is `True`, the reference file will be rendered as a jinja2 template. The template is passed the following variables:

- `sphinx_version` – the Sphinx version number, as a tuple of integers.
- `python_version` – the Python version number, in the form returned by `sys.version_info`.

Example usage

```
@pytest.mark.parametrize("page", ["index.html"], indirect=True)
def test_page(page: BeautifulSoup, html_regression: HTMLRegressionFixture):
    html_regression.check(page, file_regression)
```

fixture html_regression

Scope: function

Returns an `HTMLRegressionFixture` scoped to the test function.

New in version 2.0.0.

Return type `HTMLRegressionFixture`

sphinx_toolbox.utils

General utility functions.

Functions:

<code>add_nbsp_substitution(config)</code>	Adds the <code> nbsp </code> substitution directive to the <code>reStructuredText</code> prolog.
<code>allow_subclass_add(app, *documenters)</code>	Add the given autodocumenters, but only if a subclass of it is not already registered.
<code>baseclass_is_private(obj)</code>	Returns <code>True</code> if the first and only base class starts with a double underscore.
<code>begin_generate(documenter[, real_modname, ...])</code>	Boilerplate for the top of <code>generate</code> in <code>sphinx.ext.autodoc.Documenter</code> subclasses.
<code>code_repr(obj)</code>	Returns the repr of the given object as <code>reStructuredText</code> inline code.
<code>escape_trailing__(string)</code>	Returns the given string with trailing underscores escaped to prevent Sphinx treating them as references.
<code>filter_members_warning(member, exception)</code>	Log a warning when filtering members.
<code>flag(argument)</code>	Check for a valid flag option (no argument) and return <code>True</code> .
<code>get_first_matching(condition, iterable[, ...])</code>	Returns the first value in <code>iterable</code> that meets <code>condition</code> , or <code>default</code> if none match.
<code>is_namedtuple(obj)</code>	Returns whether the given object is a <code>collections.namedtuple()</code> class.
<code>make_github_url(username, repository)</code>	Construct a URL to a GitHub repository from a username and repository name.
<code>parse_parameters(lines[, tab_size])</code>	Parse parameters from the docstring of a class/function.
<code>unknown_module_warning(documenter)</code>	Log a warning that the module to import the object from is unknown.

Data:

<code>GITHUB_COM</code>	Instance of <code>aiohttp.url.RequestURL</code> that points to the GitHub website.
<code>OptionSpec</code>	Type hint for the <code>option_spec</code> variable of Docutils directives.
<code>SetupFunc</code>	Type annotation for Sphinx extensions' <code>setup</code> functions.
<code>typed_flag_regex</code>	Regex to match <code>:type <name>: <type> flags</code> .
<code>typed_param_regex</code>	Regex to match <code>:param <type> <name>: <docstring> flags</code> .
<code>untyped_param_regex</code>	Regex to match <code>:param <name>: <docstring> flags</code> .

Exceptions:

<code>NoMatchError</code>	Raised when no matching values were found in <code>get_first_matching()</code> .
---------------------------	--

Classes:

<code>Param</code>	<code>TypedDict</code> to represent a parameter parsed from a class or function's docstring.
<code>Purger(attr_name)</code>	Class to purge redundant nodes.
<code>SphinxExtMetadata</code>	<code>typing.TypedDict</code> representing the metadata dictionary returned by Sphinx extensions' setup functions.

add_nbsp_substitution(*config*)

Adds the `|nbsp|` substitution directive to the reStructuredText prolog.

New in version 2.1.0.

Parameters `config` (`Config`)

allow_subclass_add(*app*, **documenters*)

Add the given autodocumenters, but only if a subclass of it is not already registered.

This allows other libraries to extend the autodocumenters.

New in version 0.8.0.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **documenters** (`Type[Documenter]`)

baseclass_is_private(*obj*)

Returns `True` if the first and only base class starts with a double underscore.

Parameters `obj` (`Type`)

Return type `bool`

begin_generate(*documenter*, *real_modname*=None, *check_module*=False)

Boilerplate for the top of `generate` in `sphinx.ext.autodoc.Documenter` subclasses.

New in version 0.2.0.

Parameters

- **documenter** (`Documenter`)
- **real_modname** (`Optional[str]`) – Default `None`.
- **check_module** (`bool`) – Default `False`.

Return type `Optional[str]`

Returns The sourcename, or `None` if certain conditions are met, to indicate that the `Documenter` class should exit early.

code_repr(*obj*)

Returns the repr of the given object as reStructuredText inline code.

New in version 0.9.0.

Parameters *obj* (*Any*)

Return type *str*

escape_trailing__(*string*)

Returns the given string with trailing underscores escaped to prevent Sphinx treating them as references.

New in version 0.8.0.

Parameters *string* (*str*)

Return type *str*

filter_members_warning(*member*, *exception*)

Log a warning when filtering members.

New in version 0.2.0.

Parameters

- *member*
- *exception* (*Exception*)

flag(*argument*)

Check for a valid flag option (no argument) and return *True*.

Used in the *option_spec* of directives.

See also: *docutils.parsers.rst.directives.flag*, which returns *None* instead of *True*.

Raises *ValueError* if an argument is given.

Return type *bool*

get_first_matching(*condition*, *iterable*, *default=no_default*)

Returns the first value in *iterable* that meets *condition*, or *default* if none match.

New in version 0.7.0.

Parameters

- *condition* (*Callable*[[*Any*], *bool*]) – The condition to evaluate.
- *iterable* (*Iterable*[~*T*])
- *default* (~*T*) – The default value to return if no values in *iterable* match. Default *no_default*.

Return type ~*T*

GITHUB_COM = *RequestsURL*('https://github.com')

Type: *RequestsURL*

Instance of *apeye.requests_url.RequestsURL* that points to the GitHub website.

is_namedtuple(*obj*)

Returns whether the given object is a `collections.namedtuple()` class.

New in version 0.8.0.

Parameters *obj* (*Any*)

Return type `bool`

make_github_url(*username*, *repository*)

Construct a URL to a GitHub repository from a username and repository name.

Parameters

- **username** (*str*) – The username of the GitHub account that owns the repository.
- **repository** (*str*) – The name of the repository.

Return type `RequestsURL`

exception NoMatchError

Bases: `ValueError`

Raised when no matching values were found in `get_first_matching()`.

New in version 0.7.0.

OptionSpec

Type hint for the `option_spec` variable of Docutils directives.

Alias of `Mapping[str, Callable[[str], Any]]`

typeddict Param

Bases: `dict`

`TypedDict` to represent a parameter parsed from a class or function's docstring.

New in version 0.8.0.

Required Keys

- **doc** (`List[str]`) – The docstring of the parameter.
- **type** (*str*) – The type of the parameter.

parse_parameters(*lines*, *tab_size=8*)

Parse parameters from the docstring of a class/function.

New in version 0.8.0.

Parameters

- **lines** (`List[str]`) – The lines of the docstring
- **tab_size** (*int*) – Default 8.

Return type `Tuple[Dict[str, Param], List[str], List[str]]`

Returns A mapping of parameter names to their docstrings and types, a list of docstring lines that appeared before the parameters, and the list of docstring lines that appear after the parameters.

class Purger(*attr_name*)

Bases: `object`

Class to purge redundant nodes.

Parameters *attr_name* (`str`) – The name of the build environment’s attribute that stores the list of nodes, e.g. `all_installation_nodes`.

Methods:

<code>purge_nodes(app, env, docname)</code>	Remove all redundant nodes.
<code>get_outdated_docnames(app, env, added, ...)</code>	Returns a list of all docnames containing one or more nodes this <i>Purger</i> is aware of.
<code>add_node(env, node, targetnode, lineno)</code>	Add a node.

purge_nodes(*app, env, docname*)

Remove all redundant nodes.

This function can be configured for the `env-purge-doc` event:

```
my_node_purger = Purger("all_my_node_nodes")

def setup(app: Sphinx):
    app.connect("env-purge-doc", my_node_purger.purge_nodes)
```

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **env** (`BuildEnvironment`) – The Sphinx build environment.
- **docname** (`str`) – The name of the document to remove nodes for.

get_outdated_docnames(*app, env, added, changed, removed*)

Returns a list of all docnames containing one or more nodes this *Purger* is aware of.

This function can be configured for the `env-get-outdated` event:

```
my_node_purger = Purger("all_my_node_nodes")

def setup(app: Sphinx):
    app.connect("env-get-outdated", my_node_purger.get_outdated_docnames)
```

New in version 2.7.0.

Parameters

- **app** (`Sphinx`) – The Sphinx application.
- **env** (`BuildEnvironment`) – The Sphinx build environment.
- **added** (`Set[str]`) – A set of newly added documents.
- **changed** (`Set[str]`) – A set of document names whose content has changed.
- **removed** (`Set[str]`) – A set of document names which have been removed.

Return type `List[str]`

add_node(*env*, *node*, *targetnode*, *lineno*)

Add a node.

Parameters

- **env** (`BuildEnvironment`) – The Sphinx build environment.
- **node** (`Node`)
- **targetnode** (`Node`)
- **lineno** (`int`)

SetupFunc

Type annotation for Sphinx extensions' `setup` functions.

New in version 1.9.0.

Alias of `Callable[[Sphinx], Optional[SphinxExtMetadata]]`

typeddict SphinxExtMetadata

Bases: `dict`

`typing.TypedDict` representing the metadata dictionary returned by Sphinx extensions' `setup` functions.

This is treated by Sphinx as metadata of the extension.

Optional Keys

- **version** (`str`) – A string that identifies the extension version. It is used for extension version requirement checking and informational purposes. If not given, 'unknown version' is substituted.
- **env_version** (`int`) – An integer that identifies the version of env data structure if the extension stores any data to environment. It is used to detect the data structure has been changed from last build. Extensions have to increment the version when data structure has changed. If not given, Sphinx considers the extension does not stores any data to environment.
- **parallel_read_safe** (`bool`) – A boolean that specifies if parallel reading of source files can be used when the extension is loaded. It defaults to `False`, i.e. you have to explicitly specify your extension to be parallel-read-safe after checking that it is.
- **parallel_write_safe** (`bool`) – A boolean that specifies if parallel writing of output files can be used when the extension is loaded. Since extensions usually don't negatively influence the process, this defaults to `True`.

typed_flag_regex

Type: `Pattern[str]`

Regex to match `:type <name>: <type> flags`.

New in version 0.8.0.

Pattern	<code>^:(paramtype type)\s*([A-Za-z_]\w*\s*):\s*(.*)</code>
Flags	<code>re.ASCII</code>

typed_param_regex

Type: `Pattern[str]`

Regex to match `:param <type> <name>: <docstring> flags`.

New in version 0.8.0.

Pat-tern	<code>^:(param parameter arg argument)\s*([A-Za-z_]\w*\s+)([A-Za-z_]\w*\s*):\s*s*(.*)</code>
Flags	<code>re.ASCII</code>

unknown_module_warning(*documenter*)

Log a warning that the module to import the object from is unknown.

New in version 0.2.0.

Parameters **documenter** (`Documenter`)

untyped_param_regex

Type: `Pattern[str]`

Regex to match `:param <name>: <docstring> flags`.

New in version 0.8.0.

Pattern	<code>^:(param parameter arg argument)\s*([A-Za-z_]\w*\s*):\s*s*(.*)</code>
Flags	<code>re.ASCII</code>

Python Module Index

a

[autonamedtuple_demo](#), 86
[autoprotocol_demo](#), 90
[autotypeddict_demo](#), 95

S

[sphinx_toolbox](#), 5
[sphinx_toolbox.__init__](#), 143
[sphinx_toolbox.assets](#), 7
[sphinx_toolbox.changeset](#), 11
[sphinx_toolbox.code](#), 15
[sphinx_toolbox.collapse](#), 21
[sphinx_toolbox.config](#), 145
[sphinx_toolbox.confval](#), 25
[sphinx_toolbox.decorators](#), 29
[sphinx_toolbox.documentation_summary](#), 31
[sphinx_toolbox.flake8](#), 33
[sphinx_toolbox.formatting](#), 35
[sphinx_toolbox.github](#), 39
[sphinx_toolbox.github.issues](#), 42
[sphinx_toolbox.github.repos_and_users](#), 45
[sphinx_toolbox.installation](#), 52
[sphinx_toolbox.issues](#), 57
[sphinx_toolbox.latex](#), 59
[sphinx_toolbox.more_autodoc](#), 83
[sphinx_toolbox.more_autodoc.augment_defaults](#), 83
[sphinx_toolbox.more_autodoc.autonamedtuple](#), 84
[sphinx_toolbox.more_autodoc.autoprotocol](#), 88
[sphinx_toolbox.more_autodoc.autotypeddict](#), 93
[sphinx_toolbox.more_autodoc.generic_bases](#), 98
[sphinx_toolbox.more_autodoc.generic_alias](#), 99
[sphinx_toolbox.more_autodoc.no_docstring](#), 100
[sphinx_toolbox.more_autodoc.overloads](#), 101
[sphinx_toolbox.more_autodoc.regex](#), 109
[sphinx_toolbox.more_autodoc.sourcelink](#), 110
[sphinx_toolbox.more_autodoc.typehints](#), 111
[sphinx_toolbox.more_autodoc.typevars](#), 117
[sphinx_toolbox.more_autodoc.variables](#), 120
[sphinx_toolbox.more_autosummary](#), 125
[sphinx_toolbox.pre_commit](#), 65
[sphinx_toolbox.rest_example](#), 69
[sphinx_toolbox.shields](#), 75
[sphinx_toolbox.sidebar_links](#), 77
[sphinx_toolbox.source](#), 79
[sphinx_toolbox.testing](#), 149
[sphinx_toolbox.tweaks](#), 131
[sphinx_toolbox.tweaks.footnote_symbols](#), 131
[sphinx_toolbox.tweaks.latex_layout](#), 132
[sphinx_toolbox.tweaks.latex_toc](#), 132
[sphinx_toolbox.tweaks.param_dash](#), 134
[sphinx_toolbox.tweaks.sphinx_panels_tabs](#), 135
[sphinx_toolbox.tweaks.tabsize](#), 137
[sphinx_toolbox.utils](#), 157
[sphinx_toolbox.wikipedia](#), 81

Symbols

- `:alphabetical:` (*directive option*)
 - `autotypeddict` (*directive*), 93
- `:alt:` (*directive option*), 71
- `:args:` (*directive option*)
 - `pre-commit` (*directive*), 65
- `:branch:` (*directive option*)
 - `coveralls-shield` (*directive*), 73
 - `github-shield` (*directive*), 72
 - `pre-commit-ci-shield` (*directive*), 74
 - `requires-io-shield` (*directive*), 73
- `:caption:` (*directive option*)
 - `sidebar-links` (*directive*), 77
- `:class:` (*directive option*), 71
- `:commits-since:` (*directive option*)
 - `github-shield` (*directive*), 73
- `:conda:` (*directive option*)
 - `installation` (*directive*), 49
- `:conda-channels:` (*directive option*)
 - `installation` (*directive*), 49
- `:conda-name:` (*directive option*)
 - `installation` (*directive*), 49
- `:contributors:` (*directive option*)
 - `github-shield` (*directive*), 73
- `:dedent:` (*directive option*)
 - `rest-example` (*directive*), 69
- `:default:` (*directive option*)
 - `confval` (*directive*), 25
- `:downloads:` (*directive option*)
 - `pypi-shield` (*directive*), 72
- `:emphasize-lines:` (*directive option*)
 - `rest-example` (*directive*), 69
- `:execution-count:` (*directive option*)
 - `code-cell` (*directive*), 16
- `:first:` (*directive option*)
 - `extensions` (*directive*), 51
- `:flags:` (*directive option*)
 - `autoregex` (*directive*), 105
- `:flake8-version:` (*directive option*)
 - `pre-commit:flake8` (*directive*), 66
- `:force:` (*directive option*)
 - `rest-example` (*directive*), 69
- `:github:` (*directive option*)
 - `installation` (*directive*), 50
 - `sidebar-links` (*directive*), 77
- `:height:` (*directive option*), 71
- `:hooks:` (*directive option*)
 - `pre-commit` (*directive*), 65
- `:implementations:` (*directive option*)
 - `pypi-shield` (*directive*), 72
- `:import-name:` (*directive option*)
 - `extensions` (*directive*), 51
- `:last-commit:` (*directive option*)
 - `github-shield` (*directive*), 73
- `:license:` (*directive option*)
 - `github-shield` (*directive*), 73
 - `pypi-shield` (*directive*), 72
- `:member-order:` (*directive option*)
 - `autoprotocol` (*directive*), 89
- `:meta:` (*directive option*)
 - `documentation-summary` (*directive*), 31
- `:name:` (*directive option*), 71
- `:no-flags:` (*directive option*)
 - `autoregex` (*directive*), 105
- `:no-postamble:` (*directive option*)
 - `extensions` (*directive*), 51
- `:no-preamble:` (*directive option*)
 - `extensions` (*directive*), 51
- `:no-type:` (*directive option*)
 - `autoregex` (*directive*), 105
 - `autotypevar` (*directive*), 117
 - `autovvariable` (*directive*), 120
- `:no-value:` (*directive option*)
 - `autoregex` (*directive*), 105
 - `autotypevar` (*directive*), 117
 - `autovvariable` (*directive*), 120
- `:noindex:` (*directive option*)
 - `autoprotocol` (*directive*), 89
 - `autotypeddict` (*directive*), 93
 - `confval` (*directive*), 25
- `:plugin-name:` (*directive option*)
 - `pre-commit:flake8` (*directive*), 66
- `:project:` (*directive option*)
 - `pypi-shield` (*directive*), 72
 - `rtfd-shield` (*directive*), 72
- `:py-versions:` (*directive option*)
 - `pypi-shield` (*directive*), 72
- `:pypi:` (*directive option*)

- installation (directive), 49
- sidebar-links (directive), 77
- :pypi-name: (directive option)
 - installation (directive), 49
- :repository: (directive option)
 - actions-shield (directive), 73
 - codefactor-shield (directive), 74
 - coveralls-shield (directive), 73
 - github-shield (directive), 72
 - pre-commit-ci-shield (directive), 74
 - requires-io-shield (directive), 73
- :required: (directive option)
 - confval (directive), 25
- :rev: (directive option)
 - pre-commit (directive), 65
- :scale: (directive option), 71
- :show-inheritance: (directive option)
 - autoprotocol (directive), 89
 - autotypeddict (directive), 93
- :sourcelink: (directive option), 110
- :tab-width: (directive option)
 - code-block (directive), 15
 - rest-example (directive), 69
- :target: (directive option)
 - rtfd-shield (directive), 72
- :top-language: (directive option)
 - github-shield (directive), 73
- :type: (directive option)
 - autovvariable (directive), 120
 - confval (directive), 25
- :username: (directive option)
 - actions-shield (directive), 73
 - codefactor-shield (directive), 74
 - coveralls-shield (directive), 73
 - github-shield (directive), 72
 - pre-commit-ci-shield (directive), 74
 - requires-io-shield (directive), 73
- :value: (directive option)
 - autoregex (directive), 105
 - autotypevar (directive), 117
 - autovvariable (directive), 120
- :version: (directive option)
 - pypi-shield (directive), 72
 - rtfd-shield (directive), 72
- :wheel: (directive option)
 - pypi-shield (directive), 72
- :width: (directive option), 71
- :workflow: (directive option)
 - actions-shield (directive), 73
- __gt__() (HasGreaterThan method), 91
- __lt__() (HasLessThan method), 91
- __repr__() (Animal method), 86
- __repr__() (Employee method), 86
- __repr__() (Movie method), 86

- __repr__() (ObjectAlias method), 113
- __repr__() (RunSetupOutput method), 154

A

- actions-shield (directive), 73
 - :repository: (directive option), 73
 - :username: (directive option), 73
 - :workflow: (directive option), 73
- add_autodoc_attrgetter() (Sphinx method), 153
- add_autodocumenter() (Sphinx method), 153
- add_builder() (Sphinx method), 151
- add_config_value() (Sphinx method), 151
- add_content() (InstanceAttributeDocumenter method), 123
- add_content() (NamedTupleDocumenter method), 87
- add_content() (OverloadMixin method), 103
- add_content() (PatchedAutoSummClassDocumenter method), 127
- add_content() (PrettyGenericAliasDocumenter method), 100
- add_content() (ProtocolDocumenter method), 91
- add_content() (RegexDocumenter method), 106
- add_content() (TypedAttributeDocumenter method), 122
- add_content() (TypedDictDocumenter method), 97
- add_content() (TypeVarDocumenter method), 118
- add_crossref_type() (Sphinx method), 152
- add_css_file() (Sphinx method), 152
- add_directive() (Sphinx method), 152
- add_directive_header() (FunctionDocumenter method), 104
- add_directive_header() (GenericBasesClassDocumenter method), 99
- add_directive_header() (MethodDocumenter method), 104
- add_directive_header() (NamedTupleDocumenter method), 87
- add_directive_header() (PrettyGenericAliasDocumenter method), 100
- add_directive_header() (ProtocolDocumenter method), 92
- add_directive_header() (RegexDocumenter method), 107
- add_directive_header() (TypedAttributeDocumenter method), 122
- add_directive_header() (TypeVarDocumenter method), 119
- add_directive_header() (VariableDocumenter method), 121
- add_directive_to_domain() (Sphinx method), 152
- add_domain() (Sphinx method), 152
- add_enumerable_node() (Sphinx method), 152
- add_env_collector() (Sphinx method), 153
- add_event() (Sphinx method), 151

add_generic_role() (*Sphinx method*), 152
 add_html_math_renderer() (*Sphinx method*), 153
 add_html_theme() (*Sphinx method*), 153
 add_index_to_domain() (*Sphinx method*), 152
 add_js_file() (*Sphinx method*), 152
 add_latex_package() (*Sphinx method*), 152
 add_lexer() (*Sphinx method*), 153
 add_nbsp_substitution() (*in module sphinx_toolbox.utils*), 158
 add_node() (*Purger method*), 161
 add_node() (*Sphinx method*), 152
 add_object_type() (*Sphinx method*), 152
 add_post_transform() (*Sphinx method*), 152
 add_role() (*Sphinx method*), 152
 add_role_to_domain() (*Sphinx method*), 152
 add_source_parser() (*Sphinx method*), 153
 add_source_suffix() (*Sphinx method*), 153
 add_transform() (*Sphinx method*), 152
 all_typevars (*ToolboxConfig attribute*), 146
 allow_subclass_add() (*in module sphinx_toolbox.utils*), 158
 asset (*role*), 7
 asset_role() (*in module sphinx_toolbox.assets*), 8
 AssetNode (*class in sphinx_toolbox.assets*), 8
 assets_dir (*ToolboxConfig attribute*), 146
 automodule_add_nodocstring() (*in module sphinx_toolbox.more_autodoc.no_docstring*), 101
 autonamedtuple (*directive*), 84
 autonamedtuple_demo
 module, 86
 autoprotocol (*directive*), 89
 :member-order: (*directive option*), 89
 :noindex: (*directive option*), 89
 :show-inheritance: (*directive option*), 89
 autoprotocol_demo
 module, 90
 autoregex (*directive*), 105
 :flags: (*directive option*), 105
 :no-flags: (*directive option*), 105
 :no-type: (*directive option*), 105
 :no-value: (*directive option*), 105
 :value: (*directive option*), 105
 autotypeddict (*directive*), 93
 :alphabetical: (*directive option*), 93
 :noindex: (*directive option*), 93
 :show-inheritance: (*directive option*), 93
 autotypeddict_demo
 module, 95
 autotypevar (*directive*), 117
 :no-type: (*directive option*), 117
 :no-value: (*directive option*), 117
 :value: (*directive option*), 117
 autovisible (*directive*), 120

:no-type: (*directive option*), 120
 :no-value: (*directive option*), 120
 :type: (*directive option*), 120
 :value: (*directive option*), 120

B

baseclass_is_private() (*in module sphinx_toolbox.utils*), 158
 begin_generate() (*in module sphinx_toolbox.utils*), 158
 better_header_layout() (*in module sphinx_toolbox.latex*), 62
 bold-title (*role*), 35

C

can_document_member()
 (*InstanceAttributeDocumenter class method*), 122
 can_document_member() (*NamedTupleDocumenter class method*), 87
 can_document_member()
 (*PrettyGenericAliasDocumenter class method*), 100
 can_document_member() (*ProtocolDocumenter class method*), 92
 can_document_member() (*RegexDocumenter class method*), 107
 can_document_member() (*SlotsAttributeDocumenter class method*), 123
 can_document_member() (*TypedAttributeDocumenter class method*), 122
 can_document_member() (*TypedDictDocumenter class method*), 97
 can_document_member() (*TypeVarDocumenter class method*), 118
 check() (*HTMLRegressionFixture method*), 155
 check_asset_copy() (*in module sphinx_toolbox.testing*), 155
 check_html_regression() (*in module sphinx_toolbox.testing*), 154
 Class (*class in sphinx_toolbox.more_autodoc.typehints*), 114
 cleardoublepage (*directive*), 60
 ClearDoublePageDirective (*class in sphinx_toolbox.latex*), 62
 clearpage (*directive*), 60
 ClearPageDirective (*class in sphinx_toolbox.latex*), 61
 code_repr() (*in module sphinx_toolbox.utils*), 158
 code-block (*directive*), 15
 :tab-width: (*directive option*), 15
 code-cell (*directive*), 16
 :execution-count: (*directive option*), 16
 CodeBlock (*class in sphinx_toolbox.code*), 17

CodeCell (class in *sphinx_toolbox.code*), 17
codefactor-shield (directive), 73
 :repository: (directive option), 74
 :username: (directive option), 74
CodefactorShield (class in *sphinx_toolbox.shields*), 76
collapse (directive), 21
CollapseDirective (class in *sphinx_toolbox.collapse*), 22
CollapseNode (class in *sphinx_toolbox.collapse*), 22
conda_channels (ToolboxConfig attribute), 146
conda_installation() (in module *sphinx_toolbox.installation*), 54
config (Sphinx attribute), 151
ConfigurationValue (class in *sphinx_toolbox.confval*), 26
configure() (in module *sphinx_toolbox.code*), 18
configure() (in module *sphinx_toolbox.documentation_summary*), 32
configure() (in module *sphinx_toolbox.latex*), 63
configure() (in module *sphinx_toolbox.tweaks.latex_toc*), 134
confval (directive), 25
 :default: (directive option), 25
 :noindex: (directive option), 25
 :required: (directive option), 25
 :type: (directive option), 25
confval (role), 25
connect() (Sphinx method), 153
copy_asset_files() (in module *sphinx_toolbox.code*), 18
copy_asset_files() (in module *sphinx_toolbox.installation*), 54
copy_asset_files() (in module *sphinx_toolbox.more_autodoc.regex*), 109
copy_asset_files() (in module *sphinx_toolbox.shields*), 76
copy_asset_files() (in module *sphinx_toolbox.tweaks.sphinx_panels_tabs*), 135
coveralls-shield (directive), 73
 :branch: (directive option), 73
 :repository: (directive option), 73
 :username: (directive option), 73
CoverallsShield (class in *sphinx_toolbox.shields*), 76
create_body_overloads() (OverloadMixin method), 103
create_documenter() (PatchedAutosummary method), 128

D

deco (role), 29
default_preprocessors (in module *sphinx_toolbox.more_autodoc.typehints*), 116

depart_asset_node() (in module *sphinx_toolbox.assets*), 9
depart_collapse_node() (in module *sphinx_toolbox.collapse*), 22
depart_footnote() (in module *sphinx_toolbox.latex*), 61
depart_github_object_link_node() (in module *sphinx_toolbox.github.repos_and_users*), 47
depart_iabbr_node() (in module *sphinx_toolbox.formatting*), 36
depart_issue_node() (in module *sphinx_toolbox.github.issues*), 45
deprecated (directive), 11
docstring_hooks (in module *sphinx_toolbox.more_autodoc.typehints*), 116
document_keys() (TypedDictDocumenter method), 97
document_members() (ProtocolDocumenter method), 92
document_members() (TypedDictDocumenter method), 97
documentation-summary (directive), 31
 :meta: (directive option), 31
DocumentationSummaryDirective (class in *sphinx_toolbox.documentation_summary*), 32
docutils_tab_width (ToolboxConfig attribute), 146

E

escape_trailing__() (in module *sphinx_toolbox.utils*), 159
events (Sphinx attribute), 151
Example (class in *sphinx_toolbox.more_autodoc.generic_bases*), 99
extensions (directive), 51
 :first: (directive option), 51
 :import-name: (directive option), 51
 :no-postamble: (directive option), 51
 :no-preamble: (directive option), 51
ExtensionsDirective (class in *sphinx_toolbox.installation*), 52

F

filter_members() (NamedTupleDocumenter method), 88
filter_members() (ProtocolDocumenter method), 92
filter_members() (TypedDictDocumenter method), 97
filter_members_warning() (in module *sphinx_toolbox.utils*), 159
flag() (in module *sphinx_toolbox.utils*), 159
flake8-codes (directive), 33
Flake8CodesDirective (class in *sphinx_toolbox.flake8*), 33
Flake8PreCommitDirective (class in *sphinx_toolbox.pre_commit*), 66

format_annotation() (in module *sphinx_toolbox.more_autodoc.typehints*), 116
 format_default() (ConfigurationValue static method), 28
 format_required() (ConfigurationValue static method), 26
 format_signature() (FunctionDocumenter method), 103
 format_signature() (MethodDocumenter method), 104
 format_signature() (ProtocolDocumenter method), 92
 format_signature() (TypedDictDocumenter method), 97
 format_type() (ConfigurationValue static method), 26
 frobnicate() (Frobnicator method), 90
 Function (class in *sphinx_toolbox.more_autodoc.typehints*), 114
 FunctionDocumenter (class in *sphinx_toolbox.more_autodoc.overloads*), 103

G

GenericBasesClassDocumenter (class in *sphinx_toolbox.more_autodoc.generic_bases*), 99
 get_all_type_hints() (in module *sphinx_toolbox.more_autodoc.typehints*), 116
 get_doc() (SlotsAttributeDocumenter method), 123
 get_doc() (TypedAttributeDocumenter method), 122
 get_doc() (TypeVarDocumenter method), 119
 get_documenter() (in module *sphinx_toolbox.more_autosummary*), 129
 get_first_matching() (in module *sphinx_toolbox.utils*), 159
 get_issue_title() (in module *sphinx_toolbox.github.issues*), 45
 get_outdated_docnames() (Purger method), 161
 get_repo_details() (GitHubBackedShield method), 75
 get_table() (PatchedAutosummary method), 129
 get_variable_type() (in module *sphinx_toolbox.more_autodoc.variables*), 124
 github:issue (role), 39
 github:org (role), 40
 github:pull (role), 40
 github:repo (role), 40
 github:user (role), 40
 GITHUB_COM (in module *sphinx_toolbox.utils*), 159
 github_installation() (in module *sphinx_toolbox.installation*), 54
 github_issues_url (ToolboxConfig attribute), 146
 github_pull_url (ToolboxConfig attribute), 146
 github_repository (ToolboxConfig attribute), 146
 github_source_url (ToolboxConfig attribute), 147

github_url (ToolboxConfig attribute), 147
 github_username (ToolboxConfig attribute), 147
 github-shield (directive), 72
 :branch: (directive option), 72
 :commits-since: (directive option), 73
 :contributors: (directive option), 73
 :last-commit: (directive option), 73
 :license: (directive option), 73
 :repository: (directive option), 72
 :top-language: (directive option), 73
 :username: (directive option), 72
 GitHubActionsShield (class in *sphinx_toolbox.shields*), 75
 GitHubBackedShield (class in *sphinx_toolbox.shields*), 75
 GitHubDomain (class in *sphinx_toolbox.github*), 41
 GitHubObjectLinkNode (class in *sphinx_toolbox.github.repos_and_users*), 45
 GitHubShield (class in *sphinx_toolbox.shields*), 75

H

html_themes (Sphinx attribute), 151
 HTMLRegexParser (class in *sphinx_toolbox.more_autodoc.regex*), 108
 HTMLRegressionFixture (class in *sphinx_toolbox.testing*), 155

I

iabbr (role), 35
 import_by_name() (PatchedAutosummary method), 129
 import_parent() (InstanceAttributeDocumenter method), 123
 installation (directive), 49
 :conda: (directive option), 49
 :conda-channels: (directive option), 49
 :conda-name: (directive option), 49
 :github: (directive option), 50
 :pypi: (directive option), 49
 :pypi-name: (directive option), 49
 InstallationDirective (class in *sphinx_toolbox.installation*), 52
 InstanceAttributeDocumenter (class in *sphinx_toolbox.more_autodoc.variables*), 122
 InvalidOptionError, 145
 is_executive() (Employee method), 86
 is_namedtuple() (in module *sphinx_toolbox.utils*), 159
 issue (role), 57
 issue_role() (in module *sphinx_toolbox.github.issues*), 44
 IssueNode (class in *sphinx_toolbox.github.issues*), 42
 IssueNodeWithName (class in *sphinx_toolbox.github.issues*), 42

ItalicAbbreviation (class in *sphinx_toolbox.formatting*), 36

ItalicAbbreviationNode (class in *sphinx_toolbox.formatting*), 36

L

`latex:cleardoublepage` (directive), 60

`latex:clearpage` (directive), 60

`latex:samepage` (directive), 60

`latex:vspace` (directive), 60

`latex_depart_iabbr_node()` (in module *sphinx_toolbox.formatting*), 37

`latex_textcolor()` (in module *sphinx_toolbox.more_autodoc.regex*), 109

`latex_visit_iabbr_node()` (in module *sphinx_toolbox.formatting*), 36

LaTeXDomain (class in *sphinx_toolbox.latex*), 62

LaTeXRegexParser (class in *sphinx_toolbox.more_autodoc.regex*), 108

M

`maintained-shield:` (directive), 72

MaintainedShield (class in *sphinx_toolbox.shields*), 75

`make_github_url()` (in module *sphinx_toolbox.utils*), 160

`make_installation_instructions()` (in module *sphinx_toolbox.installation*), 55

`make_rest_example()` (in module *sphinx_toolbox.rest_example*), 70

`make_wikipedia_link()` (in module *sphinx_toolbox.wikipedia*), 82

MethodDocumenter (class in *sphinx_toolbox.more_autodoc.overloads*), 104

MissingOptionError, 145

module

autonamedtuple_demo, 86

autoprotocol_demo, 90

autotypeddict_demo, 95

sphinx_toolbox, 5

sphinx_toolbox.__init__, 143

sphinx_toolbox.assets, 7

sphinx_toolbox.changeset, 11

sphinx_toolbox.code, 15

sphinx_toolbox.collapse, 21

sphinx_toolbox.config, 145

sphinx_toolbox.confval, 25

sphinx_toolbox.decorators, 29

sphinx_toolbox.documentation_summary, 31

sphinx_toolbox.flake8, 33

sphinx_toolbox.formatting, 35

sphinx_toolbox.github, 39

sphinx_toolbox.github.issues, 42

sphinx_toolbox.github.repos_and_users, 45

sphinx_toolbox.installation, 52

sphinx_toolbox.issues, 57

sphinx_toolbox.latex, 59

sphinx_toolbox.more_autodoc, 83

sphinx_toolbox.more_autodoc.augment_defaults, 83

sphinx_toolbox.more_autodoc.autonamedtuple, 84

sphinx_toolbox.more_autodoc.autoprotocol, 88

sphinx_toolbox.more_autodoc.autotypeddict, 93

sphinx_toolbox.more_autodoc.generic_bases, 98

sphinx_toolbox.more_autodoc.genericalias, 99

sphinx_toolbox.more_autodoc.no_docstring, 100

sphinx_toolbox.more_autodoc.overloads, 101

sphinx_toolbox.more_autodoc.regex, 109

sphinx_toolbox.more_autodoc.sourcelink, 110

sphinx_toolbox.more_autodoc.typehints, 111

sphinx_toolbox.more_autodoc.typevars, 117

sphinx_toolbox.more_autodoc.variables, 120

sphinx_toolbox.more_autosummary, 125

sphinx_toolbox.pre_commit, 65

sphinx_toolbox.rest_example, 69

sphinx_toolbox.shields, 75

sphinx_toolbox.sidebar_links, 77

sphinx_toolbox.source, 79

sphinx_toolbox.testing, 149

sphinx_toolbox.tweaks, 131

sphinx_toolbox.tweaks.footnote_symbols, 131

sphinx_toolbox.tweaks.latex_layout, 132

sphinx_toolbox.tweaks.latex_toc, 132

sphinx_toolbox.tweaks.param_dash, 134

sphinx_toolbox.tweaks.sphinx_panels_tabs, 135

sphinx_toolbox.tweaks.tabsize, 137

sphinx_toolbox.utils, 157

sphinx_toolbox.wikipedia, 81

Module (class in

sphinx_toolbox.more_autodoc.typehints), 113

`my_decorator()` (in module

sphinx_toolbox.decorators), 29

MyClass (class in *sphinx_toolbox.tweaks.param_dash*), 134

N

namedtuple (role), 84

NamedTupleDocumenter (class in *sphinx_toolbox.more_autodoc.autonamedtuple*), 87

no_docstring_process_docstring() (in module *sphinx_toolbox.more_autodoc.no_docstring*), 101

no_formatting() (in module *sphinx_toolbox.more_autodoc.regex*), 109

no_unbound_typevars (ToolboxConfig attribute), 147

NoMatchError, 160

O

ObjectAlias (class in *sphinx_toolbox.more_autodoc.typehints*), 113

options (InstallationDirective attribute), 53

OptionSpec (in module *sphinx_toolbox.utils*), 160

output-cell (directive), 16

OutputCell (class in *sphinx_toolbox.code*), 17

OverloadMixin (class in *sphinx_toolbox.more_autodoc.overloads*), 103

P

parse_hooks() (in module *sphinx_toolbox.pre_commit*), 67

parse_parameters() (in module *sphinx_toolbox.utils*), 160

parse_pattern() (RegexParser method), 107

parse_regex_flags() (in module *sphinx_toolbox.more_autodoc.regex*), 109

PatchedAutosummary (class in *sphinx_toolbox.more_autosummary*), 127

PatchedAutoSummClassDocumenter (class in *sphinx_toolbox.more_autosummary*), 127

PatchedAutoSummModuleDocumenter (class in *sphinx_toolbox.more_autosummary*), 127

pre-commit (directive), 65

:args: (directive option), 65

:hooks: (directive option), 65

:rev: (directive option), 65

pre-commit:flake8 (directive), 66

:flake8-version: (directive option), 66

:plugin-name: (directive option), 66

pre-commit-ci-shield (directive), 74

:branch: (directive option), 74

:repository: (directive option), 74

:username: (directive option), 74

pre-commit-shield (directive), 74

PreCommitCIShield (class in *sphinx_toolbox.shields*), 76

PreCommitDirective (class in *sphinx_toolbox.pre_commit*), 66

PreCommitShield (class in *sphinx_toolbox.shields*), 76

Preprocessor (in module *sphinx_toolbox.more_autodoc.typehints*), 116

PrettyGenericAliasDocumenter (class in *sphinx_toolbox.more_autodoc.genericalias*), 100

process_docstring() (in module *sphinx_toolbox.more_autodoc.typehints*), 114

process_documenter_options() (in module *sphinx_toolbox.more_autodoc.augment_defaults*), 83

process_github_option() (SidebarLinksDirective method), 78

process_link() (PyDecoXRefRole method), 30

process_overload_signature() (FunctionDocumenter method), 104

process_overload_signature() (MethodDocumenter method), 104

process_overload_signature() (OverloadMixin method), 103

process_signature() (in module *sphinx_toolbox.more_autodoc.typehints*), 114

Prompt (class in *sphinx_toolbox.code*), 18

protocol (role), 89

ProtocolDocumenter (class in *sphinx_toolbox.more_autodoc.autoprotocol*), 91

pull (role), 57

pull_role() (in module *sphinx_toolbox.github.issues*), 44

purge_nodes() (Purger method), 161

Purger (class in *sphinx_toolbox.utils*), 161

PyDecoXRefRole (class in *sphinx_toolbox.decorators*), 30

pypi_installation() (in module *sphinx_toolbox.installation*), 55

pypi-shield (directive), 72

:downloads: (directive option), 72

:implementations: (directive option), 72

:license: (directive option), 72

:project: (directive option), 72

:py-versions: (directive option), 72

:version: (directive option), 72

:wheel: (directive option), 72

PyPIShield (class in *sphinx_toolbox.shields*), 75

R

regex (role), 106

RegexDocumenter (class in *sphinx_toolbox.more_autodoc.regex*), 106

RegexParser (class in *sphinx_toolbox.more_autodoc.regex*), 107

register() (Sources method), 54

`register_confval()` (in module `sphinx_toolbox.confval`), 28
`registry` (Sphinx attribute), 151
`remove_html_footer()` (in module `sphinx_toolbox.testing`), 154
`remove_html_link_tags()` (in module `sphinx_toolbox.testing`), 154
`replace_unknown_unicode()` (in module `sphinx_toolbox.latex`), 62
`repository_role()` (in module `sphinx_toolbox.github.repos_and_users`), 46
`require_sphinx()` (Sphinx method), 153
`requires-io-shield` (directive), 73
 :branch: (directive option), 73
 :repository: (directive option), 73
 :username: (directive option), 73
`RequiresIOShield` (class in `sphinx_toolbox.shields`), 75
`resolve_type()` (TypeVarDocumenter method), 118
`rest_example_purger` (in module `sphinx_toolbox.rest_example`), 70
`rest-example` (directive), 69
 :dedent: (directive option), 69
 :emphasize-lines: (directive option), 69
 :force: (directive option), 69
 :tab-width: (directive option), 69
`reSTExampleDirective` (class in `sphinx_toolbox.rest_example`), 70
`rst_prolog` (ToolboxConfig attribute), 147
`rtfd-shield` (directive), 72
 :project: (directive option), 72
 :target: (directive option), 72
 :version: (directive option), 72
`RTFDShield` (class in `sphinx_toolbox.shields`), 75
`run()` (CodeBlock method), 17
`run()` (CodeCell method), 17
`run()` (CollapseDirective method), 22
`run()` (ConfigurationValue method), 26
`run()` (DocumentationSummaryDirective method), 32
`run()` (ExtensionsDirective method), 52
`run()` (InstallationDirective method), 53
`run()` (ItalicAbbreviation method), 36
`run()` (reSTExampleDirective method), 70
`run()` (SidebarLinksDirective method), 78
`run_generic()` (InstallationDirective method), 53
`run_html()` (InstallationDirective method), 53
`run_setup()` (in module `sphinx_toolbox.testing`), 153

S

`samepage` (directive), 60
`SamepageDirective` (class in `sphinx_toolbox.latex`), 61
`set_translator()` (Sphinx method), 151
`setup()` (in module `sphinx_toolbox.__init__`), 143
`setup()` (in module `sphinx_toolbox.assets`), 9

`setup()` (in module `sphinx_toolbox.changeset`), 12
`setup()` (in module `sphinx_toolbox.code`), 18
`setup()` (in module `sphinx_toolbox.collapse`), 23
`setup()` (in module `sphinx_toolbox.confval`), 28
`setup()` (in module `sphinx_toolbox.decorators`), 30
`setup()` (in module `sphinx_toolbox.documentation_summary`), 32
`setup()` (in module `sphinx_toolbox.flake8`), 33
`setup()` (in module `sphinx_toolbox.formatting`), 37
`setup()` (in module `sphinx_toolbox.github`), 42
`setup()` (in module `sphinx_toolbox.installation`), 55
`setup()` (in module `sphinx_toolbox.issues`), 58
`setup()` (in module `sphinx_toolbox.latex`), 63
`setup()` (in module `sphinx_toolbox.more_autodoc.augment_defaults`), 83
`setup()` (in module `sphinx_toolbox.more_autodoc.autonamedtuple`), 88
`setup()` (in module `sphinx_toolbox.more_autodoc.autoprotocol`), 92
`setup()` (in module `sphinx_toolbox.more_autodoc.autotypeddict`), 98
`setup()` (in module `sphinx_toolbox.more_autodoc.generic_bases`), 99
`setup()` (in module `sphinx_toolbox.more_autodoc.genericalias`), 100
`setup()` (in module `sphinx_toolbox.more_autodoc.no_docstring`), 101
`setup()` (in module `sphinx_toolbox.more_autodoc.overloads`), 105
`setup()` (in module `sphinx_toolbox.more_autodoc.regex`), 110
`setup()` (in module `sphinx_toolbox.more_autodoc.sourcelink`), 111
`setup()` (in module `sphinx_toolbox.more_autodoc.typehints`), 116
`setup()` (in module `sphinx_toolbox.more_autodoc.typevars`), 119
`setup()` (in module `sphinx_toolbox.more_autodoc.variables`), 124
`setup()` (in module `sphinx_toolbox.more_autosummary`), 129
`setup()` (in module `sphinx_toolbox.pre_commit`), 67
`setup()` (in module `sphinx_toolbox.rest_example`), 70
`setup()` (in module `sphinx_toolbox.shields`), 76
`setup()` (in module `sphinx_toolbox.sidebar_links`), 78
`setup()` (in module `sphinx_toolbox.source`), 80
`setup()` (in module

sphinx_toolbox.tweaks.footnote_symbols), 131
 setup() (in module *sphinx_toolbox.tweaks.latex_layout*), 132
 setup() (in module *sphinx_toolbox.tweaks.latex_toc*), 134
 setup() (in module *sphinx_toolbox.tweaks.param_dash*), 135
 setup() (in module *sphinx_toolbox.tweaks.sphinx_panels_tabs*), 135
 setup() (in module *sphinx_toolbox.tweaks.tabsize*), 137
 setup() (in module *sphinx_toolbox.wikipedia*), 82
 setup_extension() (*Sphinx* method), 153
 SetupFunc (in module *sphinx_toolbox.utils*), 162
 Shield (class in *sphinx_toolbox.shields*), 75
 shield_default_option_spec (in module *sphinx_toolbox.shields*), 75
 SHIELDS_IO (in module *sphinx_toolbox.shields*), 75
 sidebar-links (directive), 77
 :caption: (directive option), 77
 :github: (directive option), 77
 :pypi: (directive option), 77
 SidebarLinksDirective (class in *sphinx_toolbox.sidebar_links*), 78
 SlotsAttributeDocumenter (class in *sphinx_toolbox.more_autodoc.variables*), 123
 sort_members() (*NamedTupleDocumenter* method), 88
 sort_members() (*TypedDictDocumenter* method), 98
 source (role), 79
 source_link_target (*ToolboxConfig* attribute), 147
 source_role() (in module *sphinx_toolbox.source*), 80
 sourcecode (directive), 15
 sourcelinks_process_docstring() (in module *sphinx_toolbox.more_autodoc.sourcelink*), 111
 Sources (class in *sphinx_toolbox.installation*), 53
 sources (in module *sphinx_toolbox.installation*), 55
 span() (in module *sphinx_toolbox.more_autodoc.regex*), 109
 Sphinx (class in *sphinx_toolbox.testing*), 149
 sphinx_toolbox
 module, 5
 sphinx_toolbox.__init__
 module, 143
 sphinx_toolbox.assets
 module, 7
 sphinx_toolbox.changeset
 module, 11
 sphinx_toolbox.code
 module, 15
 sphinx_toolbox.collapse
 module, 21
 sphinx_toolbox.config
 module, 145
 sphinx_toolbox.confval
 module, 25
 sphinx_toolbox.decorators
 module, 29
 sphinx_toolbox.documentation_summary
 module, 31
 sphinx_toolbox.flake8
 module, 33
 sphinx_toolbox.formatting
 module, 35
 sphinx_toolbox.github
 module, 39
 sphinx_toolbox.github.issues
 module, 42
 sphinx_toolbox.github.repos_and_users
 module, 45
 sphinx_toolbox.installation
 module, 52
 sphinx_toolbox.issues
 module, 57
 sphinx_toolbox.latex
 module, 59
 sphinx_toolbox.more_autodoc
 module, 83
 sphinx_toolbox.more_autodoc.augment_defaults
 module, 83
 sphinx_toolbox.more_autodoc.autonamedtuple
 module, 84
 sphinx_toolbox.more_autodoc.autoprotocol
 module, 88
 sphinx_toolbox.more_autodoc.autotypeddict
 module, 93
 sphinx_toolbox.more_autodoc.generic_bases
 module, 98
 sphinx_toolbox.more_autodoc.genericalias
 module, 99
 sphinx_toolbox.more_autodoc.no_docstring
 module, 100
 sphinx_toolbox.more_autodoc.overloads
 module, 101
 sphinx_toolbox.more_autodoc.regex
 module, 109
 sphinx_toolbox.more_autodoc.sourcelink
 module, 110
 sphinx_toolbox.more_autodoc.typehints
 module, 111
 sphinx_toolbox.more_autodoc.typevars
 module, 117
 sphinx_toolbox.more_autodoc.variables
 module, 120
 sphinx_toolbox.more_autosummary
 module, 125
 sphinx_toolbox.pre_commit
 module, 65
 sphinx_toolbox.rest_example

- module, 69
- sphinx_toolbox.shields
 - module, 75
- sphinx_toolbox.sidebar_links
 - module, 77
- sphinx_toolbox.source
 - module, 79
- sphinx_toolbox.testing
 - module, 149
- sphinx_toolbox.tweaks
 - module, 131
- sphinx_toolbox.tweaks.footnote_symbols
 - module, 131
- sphinx_toolbox.tweaks.latex_layout
 - module, 132
- sphinx_toolbox.tweaks.latex_toc
 - module, 132
- sphinx_toolbox.tweaks.param_dash
 - module, 134
- sphinx_toolbox.tweaks.sphinx_panels_tabs
 - module, 135
- sphinx_toolbox.tweaks.tabsize
 - module, 137
- sphinx_toolbox.utils
 - module, 157
- sphinx_toolbox.wikipedia
 - module, 81

T

TerminalRegexParser (class in *sphinx_toolbox.more_autodoc.regex*), 108

ToolboxConfig (class in *sphinx_toolbox.config*), 145

type_template (in module *sphinx_toolbox.more_autodoc.variables*), 123

typed_flag_regex (in module *sphinx_toolbox.utils*), 162

typed_param_regex (in module *sphinx_toolbox.utils*), 162

TypedAttributeDocumenter (class in *sphinx_toolbox.more_autodoc.variables*), 121

typeddict (role), 93

TypedDictDocumenter (class in *sphinx_toolbox.more_autodoc.autotypeddict*), 96

TypeVarDocumenter (class in *sphinx_toolbox.more_autodoc.typevars*), 118

U

unknown_module_warning() (in module *sphinx_toolbox.utils*), 163

unskip_typevars() (in module *sphinx_toolbox.more_autodoc.typevars*), 119

untyped_param_regex (in module *sphinx_toolbox.utils*), 163

use_package() (in module *sphinx_toolbox.latex*), 61

user_role() (in module *sphinx_toolbox.github.repos_and_users*), 46

V

validate_config() (in module *sphinx_toolbox.config*), 147

validate_config() (in module *sphinx_toolbox.github*), 41

VariableDocumenter (class in *sphinx_toolbox.more_autodoc.variables*), 121

versionadded (directive), 11

VersionChange (class in *sphinx_toolbox.changeset*), 12

versionchanged (directive), 11

versionremoved (directive), 11

visit_asset_node() (in module *sphinx_toolbox.assets*), 8

visit_collapse_node() (in module *sphinx_toolbox.collapse*), 22

visit_footnote() (in module *sphinx_toolbox.latex*), 61

visit_github_object_link_node() (in module *sphinx_toolbox.github.repos_and_users*), 47

visit_iabbr_node() (in module *sphinx_toolbox.formatting*), 36

visit_issue_node() (in module *sphinx_toolbox.github.issues*), 45

visit_prompt_html() (in module *sphinx_toolbox.code*), 18

visit_prompt_latex() (in module *sphinx_toolbox.code*), 18

VSpaceDirective (class in *sphinx_toolbox.latex*), 62

W

wikipedia (role), 81

wikipedia_lang (ToolboxConfig attribute), 147