

---

# Documentation System

*Release 10.2*

**The Sage Development Team**

**Dec 06, 2023**



# CONTENTS

<b>1</b>	<b>Sage Doc Builders</b>	<b>1</b>
<b>2</b>	<b>Sphinx Extensions</b>	<b>13</b>
	<b>Python Module Index</b>	<b>29</b>
	<b>Index</b>	<b>31</b>



## SAGE DOC BUILDERS

### 1.1 Sage docbuild main

This module defines the Sage documentation build command:

```
sage --docbuild [OPTIONS] DOCUMENT (FORMAT | COMMAND)
```

If **FORMAT** is given, it builds **DOCUMENT** in **FORMAT**. If **COMMAND** is given, it returns information about **DOCUMENT**.

Run `sage --docbuild` to get detailed explanations about arguments and options.

Positional arguments:

DOCUMENT	name of the document to build. It can be either one of the documents listed by <code>-D</code> or <code>'file=/path/to/FILE'</code> to build.
<code>↪documentation</code>	for this specific file.
FORMAT or COMMAND	document output format (or command)

Standard options:

<code>-h, --help</code>	show a help message and exit
<code>-H, --help-all</code>	show an extended help message and exit
<code>-D, --documents</code>	list all available DOCUMENTs
<code>-F, --formats</code>	list all output FORMATs
<code>-C DOC, --commands DOC</code>	list all COMMANDs for DOCUMENT DOC; use 'all' to list all
<code>-i, --inherited</code>	include inherited members in reference manual; may be slow,
<code>↪may fail for PDF output</code>	
<code>-u, --underscore</code>	include variables prefixed with '_' in reference manual; may be slow, may fail for PDF output
<code>-j, --mathjax, --jsmath</code>	ignored for backwards compatibility
<code>--no-plot</code>	do not include graphics auto-generated using the <code>'.. plot'</code>
<code>↪markup</code>	
<code>--include-tests-blocks</code>	include TESTS blocks in the reference manual
<code>--no-pdf-links</code>	do not include PDF links in DOCUMENT 'website'; FORMATs: html, json, pickle, web
<code>--warn-links</code>	issue a warning whenever a link is not properly resolved; equivalent to <code>'--sphinx-opts -n'</code> (sphinx option:
<code>↪nitpicky)</code>	
<code>--check-nested</code>	check picklability of nested classes in DOCUMENT 'reference'
<code>--no-prune-empty-dirs</code>	do not prune empty directories in the documentation sources
<code>-N, --no-colors</code>	do not color output; does not affect children

(continues on next page)

(continued from previous page)

```

-q, --quiet                work quietly; same as --verbose=0
-v LEVEL, --verbose LEVEL  report progress at LEVEL=0 (quiet), 1 (normal), 2
                           (info), or 3 (debug); does not affect children
-o DIR, --output DIR       if DOCUMENT is a single file ('file=...'), write output to
                           ↪ this directory

```

Advanced options:

```

-S OPTS, --sphinx-opts OPTS  pass comma-separated OPTS to sphinx-build; must
                             precede OPTS with '=', as in '-S=-q,-aE' or '-S="-q,-aE"'
-U, --update-mtimes          before building reference manual, update
                             modification times for auto-generated reST files
-k, --keep-going             do not abort on errors but continue as much as
                             possible after an error
--all-documents ARG          if ARG is 'reference', list all subdocuments of
                             en/reference. If ARG is 'all', list all main documents

```

**class** `sage_docbuild.__main__.IntersphinxCache`

Bases: `object`

Replace `sphinx.ext.intersphinx.fetch_inventory` by an in-memory cached version.

**fetch\_inventory**(*app, uri, inv*)

Return the result of `sphinx.ext.intersphinx.fetch_inventory()` from a cache if possible. Otherwise, call `sphinx.ext.intersphinx.fetch_inventory()` and cache the result.

`sage_docbuild.__main__.format_columns`(*lst, align='<', cols=None, indent=4, pad=3, width=80*)

Utility function that formats a list as a simple table and returns a Unicode string representation.

The number of columns is computed from the other options, unless it's passed as a keyword argument. For help on Python's string formatter, see

<https://docs.python.org/library/string.html#format-string-syntax>

`sage_docbuild.__main__.get_formats`()

Return a list of output formats the Sage documentation builder will accept on the command-line.

`sage_docbuild.__main__.help_commands`(*name='all'*)

Append and return a tabular list of commands, if any, the Sage documentation builder can run on the indicated document. The default is to list all commands for all documents.

`sage_docbuild.__main__.help_description`(*s="", compact=False*)

Append and return a brief description of the Sage documentation builder.

If 'compact' is False, the function adds a final newline character.

`sage_docbuild.__main__.help_documents`()

Append and return a tabular list of documents, including a shortcut 'all' for all documents, available to the Sage documentation builder.

`sage_docbuild.__main__.help_examples`(*s=""*)

Append and return some usage examples for the Sage documentation builder.

`sage_docbuild.__main__.help_formats`()

Append and return a tabular list of output formats available to the Sage documentation builder.

```
class sage_docbuild.__main__.help_message_long(option_strings, dest, nargs=None, const=None,
                                              default=None, type=None, choices=None,
                                              required=False, help=None, metavar=None)
```

Bases: [Action](#)

Print an extended help message for the Sage documentation builder and exits.

```
class sage_docbuild.__main__.help_message_short(option_strings, dest, nargs=None, const=None,
                                              default=None, type=None, choices=None,
                                              required=False, help=None, metavar=None)
```

Bases: [Action](#)

Print a help message for the Sage documentation builder.

The message includes command-line usage and a list of options. The message is printed only on the first call. If error is True during this call, the message is printed only if the user hasn't requested a list (e.g., documents, formats, commands).

```
sage_docbuild.__main__.help_usage(s="", compact=False)
```

Append and return a brief usage message for the Sage documentation builder.

If 'compact' is False, the function adds a final newline character.

```
class sage_docbuild.__main__.help_wrapper(option_strings, dest, nargs=None, const=None,
                                          default=None, type=None, choices=None, required=False,
                                          help=None, metavar=None)
```

Bases: [Action](#)

A helper wrapper for command-line options to the Sage documentation builder that print lists, such as document names, formats, and document-specific commands.

```
sage_docbuild.__main__.main()
```

```
sage_docbuild.__main__.setup_logger(verbose=1, color=True)
```

Set up a Python Logger instance for the Sage documentation builder.

The optional argument sets logger's level and message format.

EXAMPLES:

```
sage: from sage_docbuild.__main__ import setup_logger, logger
sage: setup_logger()
sage: type(logger)
<class 'logging.Logger'>
```

```
sage_docbuild.__main__.setup_parser()
```

Set up and return a command-line ArgumentParser instance for the Sage documentation builder.

## 1.2 Documentation builders

This module is the starting point for building documentation, and is responsible to figure out what to build and with which options. The actual documentation build for each individual document is then done in a subprocess call to Sphinx, see [builder\\_helper\(\)](#). Note that

- The builders are configured with `build_options.py`;
- The Sphinx subprocesses are configured in `conf.py`.

*DocBuilder* is the base class of all Builders. It has builder helpers `html()`, `latex()`, `pdf()`, `inventory()`, etc, which are invoked depending on the output type. Each type corresponds with the Sphinx builder format, except that `pdf` is Sphinx latex builder plus compiling latex to pdf. Note that Sphinx inventory builder is not native to Sphinx but provided by Sage. See `sage_docbuild/ext/inventory_builder.py`. The Sphinx inventory builder is a dummy builder with no actual output but produces doctree files in `local/share/doctree` and `inventory.inv` inventory files in `local/share/inventory`.

The reference manual is built in two passes, first by *ReferenceBuilder* with `inventory` output type and secondly with `html` output type. The *ReferenceBuilder* itself uses *ReferenceTopBuilder* and *ReferenceSubBuilder* to build subcomponents of the reference manual. The *ReferenceSubBuilder* examines the modules included in the subcomponent by comparing the modification times of the module files with the times saved in `local/share/doctree/reference.pickle` from the previous build. Then new rst files are generated for new and updated modules. See `get_new_and_updated_modules()`.

After [github issue #31948](#), when Sage is built, *ReferenceBuilder* is not used and its responsibility is now taken by the Makefile in `$SAGE_ROOT/src/doc`.

**class** `sage_docbuild.builders.AllBuilder`

Bases: `object`

A class used to build all of the documentation.

**get\_all\_documents()**

Return a list of all of the documents.

A document is a directory within one of the language subdirectories of `SAGE_DOC_SRC` specified by the global `LANGUAGES` variable.

EXAMPLES:

```
sage: from sage_docbuild.builders import AllBuilder
sage: documents = AllBuilder().get_all_documents()
sage: 'en/tutorial' in documents # optional - sage_spkg
True
sage: documents[0] == 'en/reference'
True
```

**class** `sage_docbuild.builders.DocBuilder(name, lang='en')`

Bases: `object`

INPUT:

- `name` - the name of a subdirectory in `SAGE_DOC_SRC`, such as `'tutorial'` or `'bordeaux_2008'`
- `lang` - (default `"en"`) the language of the document.

**changes**(\*args, \*\*kws)

**clean**(\*args)

**html**(\*args, \*\*kws)

**htmlhelp**(\*args, \*\*kws)

**inventory**(\*args, \*\*kws)

**json**(\*args, \*\*kws)

**latex**(\*args, \*\*kws)



**linkcheck**(\*args, \*\*kws)

**pdf**()

Build the PDF files for this document.

This is done by first (re)-building the LaTeX output, going into that LaTeX directory, and running ‘make all-pdf’ there.

EXAMPLES:

```
sage: from sage_docbuild.builders import DocBuilder
sage: b = DocBuilder('tutorial')
sage: b.pdf() #not tested
```

**pickle**(\*args, \*\*kws)

**web**(\*args, \*\*kws)

**class** sage\_docbuild.builders.**ReferenceBuilder**(name, lang='en')

Bases: *AllBuilder*

This class builds the reference manual. It uses DocBuilder to build the top-level page and ReferenceSubBuilder for each sub-component.

**get\_all\_documents**(refdir)

Return a list of all reference manual components to build.

We add a component name if it's a subdirectory of the manual's directory and contains a file named ‘index.rst’.

We return the largest component (most subdirectory entries) first since they will take the longest to build.

EXAMPLES:

```
sage: from sage_docbuild.builders import ReferenceBuilder
sage: b = ReferenceBuilder('reference')
sage: refdir = os.path.join(os.environ['SAGE_DOC_SRC'], 'en', b.name) #_
↳ optional - sage_spkg
sage: sorted(b.get_all_documents(refdir)) # optional - sage_spkg
['reference/algebras',
 'reference/arithgroup',
 ...,
 'reference/valuations']
```

**class** sage\_docbuild.builders.**ReferenceSubBuilder**(\*args, \*\*kws)

Bases: *DocBuilder*

This class builds sub-components of the reference manual. It is responsible for making sure that the auto generated reST files for the Sage library are up to date.

When building any output, we must first go through and check to see if we need to update any of the autogenerated reST files. There are two cases where this would happen:

1. A new module gets added to one of the toctrees.
2. The actual module gets updated and possibly contains a new title.

**auto\_rest\_filename**(module\_name)

Return the name of the file associated to a given module

EXAMPLES:

```
sage: from sage_docbuild.builders import ReferenceSubBuilder
sage: ReferenceSubBuilder("reference").auto_rest_filename("sage.combinat.
↪partition")
'.../en/reference/sage/combinat/partition.rst'
```

**cache\_filename()**

Return the filename where the pickle of the reference cache is stored.

**clean\_auto()**

Remove all autogenerated reST files.

**get\_all\_included\_modules()**

Return an iterator for all modules which are included in the reference manual.

**get\_all\_rst\_files(exclude\_sage=True)**

Return an iterator for all rst files which are not autogenerated.

**get\_cache()**

Retrieve the reference cache which contains the options previously used by the reference builder.

If it doesn't exist, then we just return an empty dictionary. If it is corrupted, return an empty dictionary.

**get\_modified\_modules()**

Return an iterator for all the modules that have been modified since the documentation was last built.

**get\_module\_docstring\_title(module\_name)**

Return the title of the module from its docstring.

**get\_modules(filename)**

Given a filename for a reST file, return an iterator for all of the autogenerated reST files that it includes.

**get\_new\_and\_updated\_modules()**

Return an iterator for all new and updated modules that appear in the toctrees, and remove obsolete old modules.

**get\_sphinx\_environment()**

Return the Sphinx environment for this project.

**get\_unincluded\_modules()**

Return an iterator for all the modules in the Sage library which are not included in the reference manual.

**print\_included\_modules()**

Print all of the modules that are included in the Sage reference manual.

**print\_modified\_modules()**

Print a list of all the modules that have been modified since the documentation was last built.

**print\_new\_and\_updated\_modules()**

Print all the modules that appear in the toctrees that are newly included or updated.

**print\_unincluded\_modules()**

Print all of the modules which are not included in the Sage reference manual.

**save\_cache()**

Pickle the current reference cache for later retrieval.

**update\_mtimes()**

Update the modification times for reST files in the Sphinx environment for this project.

**write\_auto\_rest\_file**(*module\_name*)

Write the autogenerated reST file for *module\_name*.

**class** `sage_docbuild.builders.ReferenceTopBuilder`(\*args, \*\*kws)

Bases: *DocBuilder*

This class builds the top-level page of the reference manual.

**pdf**()

Build top-level document.

**class** `sage_docbuild.builders.SingleFileBuilder`(*path*)

Bases: *DocBuilder*

This is the class used to build the documentation for a single user-specified file. If the file is called ‘foo.py’, then the documentation is built in DIR/foo/ if the user passes the command line option “-o DIR”, or in DOT\_SAGE/docbuild/foo/ otherwise.

**class** `sage_docbuild.builders.WebsiteBuilder`(*name*, *lang*='en')

Bases: *DocBuilder*

**clean**()

When we clean the output for the website index, we need to remove all of the HTML that were placed in the parent directory.

In addition, remove the index file installed into the root doc directory.

**html**()

After we have finished building the website index page, we copy everything one directory up, that is, to the base directory `html/en`.

In addition, an index file is installed into the root doc directory.

Thus we have three `index.html` files:

`html/en/website/index.html` (not used) `html/en/index.html` (base directory) `index.html` (root doc directory)

**pdf**()

Build the website hosting pdf docs.

`sage_docbuild.builders.build_many`(*target*, *args*, *processes*=None)

Thin wrapper around `sage_docbuild.utils.build_many` which uses the docbuild settings `NUM_THREADS` and `ABORT_ON_ERROR`.

`sage_docbuild.builders.build_other_doc`(*args*)

`sage_docbuild.builders.build_ref_doc`(*args*)

`sage_docbuild.builders.builder_helper`(*type*)

Return a function which builds the documentation for output type *type*.

`sage_docbuild.builders.get_builder`(*name*)

Return an appropriate *Builder* object for the document name.

*DocBuilder* and its subclasses do all the real work in building the documentation.

`sage_docbuild.builders.get_documents`()

Return a list of document names the Sage documentation builder will accept as command-line arguments.

## 1.3 Build options

This module defines options for building Sage documentation.

## 1.4 Sphinx build script

This is Sage's version of the `sphinx-build` script. We redirect `stdout` and `stderr` to our own logger, and remove some unwanted chatter.

**class** `sage_docbuild.sphinxbuild.SageSphinxLogger`(*stream, prefix*)

Bases: `object`

This implements the file object interface to serve as `sys.stdout/sys.stderr` replacement.

**ansi\_color** = `re.compile('\\x1b\\[[0-9;]*m')`

**ansi\_reset** = `re.compile('\\x1b\\[[39;49;00m')`

**close**()

**closed** = `False`

**encoding** = `None`

**flush**()

**isatty**()

**mode** = `'w'`

**name** = `'<log>'`

**newlines** = `None`

**prefix\_len** = `9`

**raise\_errors**()

Raise an exceptions if any errors have been found while parsing the Sphinx output.

EXAMPLES:

```
sage: from sys import stdout
sage: from sage_docbuild.sphinxbuild import SageSphinxLogger
sage: logger = SageSphinxLogger(stdout, "doctestin")
sage: logger._log_line("This is a SEVERE error\n")
[doctestin] This is a SEVERE error
sage: logger.raise_errors()
Traceback (most recent call last):
...
OSError: This is a SEVERE error
```

**softspace** = `0`

**write**(*str*)

**writelines**(*sequence*)

`sage_docbuild.sphinxbuild.runsphinx()`

`sage_docbuild.sphinxbuild.term_width_line(text)`

## 1.5 Sphinx build configuration

This file contains configuration needed to customize Sphinx input and output behavior.

**class** `sage_docbuild.conf.SagecodeTransform`(*document*, *startnode=None*)

Bases: `SphinxTransform`

Transform a code block to a live code block enabled by jupyter-sphinx.

Effectively a code block like:

EXAMPLE::

```
sage: 1 + 1
2
```

is transformed into:

EXAMPLE::

```
sage: 1 + 1
2
```

.. ONLY:: html

```
.. JUPYTER-EXECUTE::
   :hide-code:
   :hide-output:
   :raises:
   :stderr:
```

```
1 + 1
```

enabling live execution of the code.

**apply()**

**default\_priority = 170**

Numerical priority of this transform, 0 through 999 (override).

**class** `sage_docbuild.conf.Tags`

Bases: `set`

**has**(*\*args*, *\*\*kws*)

`x.__contains__(y) <==> y in x.`

`sage_docbuild.conf.add_page_context`(*app*, *pagename*, *templatename*, *context*, *doctree*)

`sage_docbuild.conf.call_intersphinx`(*app*, *env*, *node*, *contnode*)

Call intersphinx and make links between Sage manuals relative.

`sage_docbuild.conf.check_nested_class_picklability(app, what, name, obj, skip, options)`

Print a warning if pickling is broken for nested classes.

`sage_docbuild.conf.debug_inf(app, message)`

`sage_docbuild.conf.find_sage_dangling_links(app, env, node, contnode)`

Try to find dangling link in local module imports or all.py.

`sage_docbuild.conf.nitpick_patch_config(app)`

Patch the default config for nitpicky

Calling `path_config` ensure that nitpicky is not considered as a Sphinx environment variable but rather as a Sage environment variable. As a consequence, changing it doesn't force the recompilation of the entire documentation.

`sage_docbuild.conf.set_intersphinx_mappings(app, config)`

Add precompiled inventory (the `objects.inv`)

`sage_docbuild.conf.setup(app)`

`sage_docbuild.conf.skip_member(app, what, name, obj, skip, options)`

To suppress Sphinx warnings / errors, we

- Don't include [aliases of] builtins.
- Don't include the docstring for any nested class which has been inserted into its module by `sage.misc.NestedClassMetaclass` only for pickling. The class will be properly documented inside its surrounding class.
- Optionally, check whether pickling is broken for nested classes.
- Optionally, include objects whose name begins with an underscore (`'_'`), i.e., "private" or "hidden" attributes, methods, etc.

Otherwise, we abide by Sphinx's decision. Note: The object `obj` is excluded (included) if this handler returns `True` (`False`).

## 1.6 Utilities

**exception** `sage_docbuild.utils.RemoteException(tb: str)`

Bases: `Exception`

Raised if an exception occurred in one of the child processes.

**tb:** `str`

**class** `sage_docbuild.utils.RemoteExceptionWrapper(exc: BaseException)`

Bases: `object`

Used by child processes to capture exceptions thrown during execution and report them to the main process, including the correct traceback.

**exc:** `BaseException`

**tb:** `str`

**exception** `sage_docbuild.utils.WorkerDiedException`(*message: Optional[str]*, *original\_exception: Optional[BaseException] = None*)

Bases: `RuntimeError`

Raised if a worker process dies unexpectedly.

**original\_exception:** `Optional[BaseException]`

`sage_docbuild.utils.build_many`(*target, args, processes=None*)

Map a list of arguments in `args` to a single-argument target function `target` in parallel using `multiprocessing.cpu_count()` (or `processes` if given) simultaneous processes.

This is a simplified version of `multiprocessing.Pool.map` from the Python standard library which avoids a couple of its pitfalls. In particular, it can abort (with a `RuntimeError`) without hanging if one of the worker processes unexpectedly dies. It also has semantics equivalent to `maxtasksperchild=1`; that is, one process is started per argument. As such, this is inefficient for processing large numbers of fast tasks, but appropriate for running longer tasks (such as doc builds) which may also require significant cleanup.

It also avoids starting new processes from a pthread, which results in at least two known issues:

- On versions of Cygwin prior to 3.0.0 there were bugs in mmap handling on threads (see [github issue #27214#comment:25](#)).
- When PARI is built with multi-threading support, forking a Sage process from a thread leaves the main Pari interface instance broken (see [github issue #26608#comment:38](#)).

In the future this may be replaced by a generalized version of the more robust parallel processing implementation from `sage.doctest.forker`.

EXAMPLES:

```
sage: from sage_docbuild.utils import build_many
sage: def target(N):
.....:     import time
.....:     time.sleep(float(0.1))
.....:     print('Processed task %s' % N)
sage: _ = build_many(target, range(8), processes=8)
Processed task ...
Processed task ...
Processed task ...
Processed task ...
Processed task ...
Processed task ...
Processed task ...
Processed task ...
```

Unlike the first version of `build_many` which was only intended to get around the Cygwin bug, this version can also return a result, and thus can be used as a replacement for `multiprocessing.Pool.map` (i.e. it still blocks until the result is ready):

```
sage: def square(N):
.....:     return N * N
sage: build_many(square, range(100))
[0, 1, 4, 9, ..., 9604, 9801]
```

If the target function raises an exception in any of the workers, `build_many` raises that exception and all other results are discarded. Any in-progress tasks may still be allowed to complete gracefully before the exception is raised:

```
sage: def target(N):
.....:     import time, os, signal
.....:     if N == 4:
.....:         # Task 4 is a poison pill
.....:         1 / 0
.....:     else:
.....:         time.sleep(float(0.5))
.....:         print('Processed task %s' % N)
```

Note: In practice this test might still show output from the other worker processes before the poison-pill is executed. It may also display the traceback from the failing process on stderr. However, due to how the doctest runner works, the doctest will only expect the final exception:

```
sage: build_many(target, range(8), processes=8)
Traceback (most recent call last):
...
    raise ZeroDivisionError("rational division by zero")
ZeroDivisionError: rational division by zero
...
    raise worker_exc.original_exception
ZeroDivisionError: rational division by zero
```

Similarly, if one of the worker processes dies unexpectedly otherwise exits non-zero (e.g. killed by a signal) any in-progress tasks will be completed gracefully, but then a `RuntimeError` is raised and pending tasks are not started:

```
sage: def target(N):
.....:     import time, os, signal
.....:     if N == 4:
.....:         # Task 4 is a poison pill
.....:         os.kill(os.getpid(), signal.SIGKILL)
.....:     else:
.....:         time.sleep(float(0.5))
.....:         print('Processed task %s' % N)
sage: build_many(target, range(8), processes=8)
Traceback (most recent call last):
...
WorkerDiedException: worker for 4 died with non-zero exit code -9
```



## SPHINX EXTENSIONS

### 2.1 Inventory builder

A customized builder which only generates intersphinx “object.inv” inventory files. The documentation files are not written.

```
class sage_docbuild.ext.inventory_builder.InventoryBuilder(app: Sphinx, env: BuildEnvironment = None)
```

Bases: DummyBuilder

A customized builder which only generates intersphinx “object.inv” inventory files. The documentation files are not written.

**epilog** = 'The inventory files are in %(outdir)s.'

The message emitted upon successful build completion. This can be a printf-style template string with the following keys: `outdir`, `project`

**finish()**

Only write the inventory files.

**format** = 'inventory'

The builder’s output format, or “” if no document output is produced.

**get\_outdated\_docs()**

Return an iterable of output files that are outdated.

**get\_target\_uri**(*docname, typ=None*)

Return the target URI for a document name.

**name** = 'inventory'

The builder’s name, for the -b command line option.

```
sage_docbuild.ext.inventory_builder.setup(app)
```

## 2.2 Sage autodoc extension

This is `sphinx.ext.autodoc` extension modified for Sage objects.

The original header of `sphinx.ext.autodoc`:

Extension to create automatic documentation from code docstrings.

Automatically insert docstrings for functions, classes or whole modules into the doctree, thus avoiding duplication between docstrings and documentation for those who like elaborate docstrings.

This module is currently based on `sphinx.ext.autodoc` from Sphinx version 5.3.0. Compare against the upstream original source file `sphinx/ext/autodoc/__init__.py`.

In the source file of this module, major modifications are delimited by a pair of comment dividers. To lessen maintenance burden, we aim at reducing the modifications.

AUTHORS:

- ? (before 2011): initial version derived from `sphinx.ext.autodoc`
- Johan S. R. Nielsen: support for `_sage_argspec_`
- Simon King (2011-04): use `sageinspect`; include public cython attributes only in the documentation if they have a docstring
- Kwankyu Lee (2018-12-26, 2022-11-08): rebased on the latest `sphinx.ext.autodoc`

```
class sage_docbuild.ext.sage_autodoc.AttributeDocumenter(directive: DocumenterBridge, name: str,  
                                                         indent: str = "")
```

```
Bases:      GenericAliasMixin,      NewTypeMixin,      SlotsMixin,      TypeVarMixin,  
          RuntimeInstanceAttributeMixin,      UninitializedInstanceAttributeMixin,  
          NonDataDescriptorMixin, DocstringStripSignatureMixin, ClassLevelDocumenter
```

Specialized Documenter subclass for attributes.

```
add_content(more_content)
```

```
add_directive_header(sig)
```

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

```
document_members(all_members=False)
```

```
get_attribute_comment(parent, attrname)
```

```
get_doc()
```

```
get_real_modname()
```

```
import_object(raiseerror=False)
```

```
static is_function_or_method(obj)
```

```
member_order = 60
```

order if `autodoc_member_order` is set to 'groupwise'

```
objtype = 'attribute'
```

name by which the directive is called (auto...) and the default generated directive name

```

option_spec: Dict[str, Callable[[str], Any]] = {'annotation': <function
annotation_option>, 'no-value': <function bool_option>, 'noindex': <function
bool_option>}

priority = 10
    priority if multiple documenters return True from can_document_member

should_suppress_value_header()

update_annotations(parent)
    Update __annotations__ to support type_comment and so on.
class sage_docbuild.ext.sage_autodoc.ClassDocumenter(*args: Any)
    Bases: DocstringSignatureMixin, ModuleLevelDocumenter
    Specialized Documenter subclass for classes.
    add_content(more_content)
    add_directive_header(sig)
    classmethod can_document_member(member, membername, isattr, parent)
    document_members(all_members=False)
    format_args(**kwargs)
    format_signature(**kwargs)
    generate(more_content=None, real_modname=None, check_module=False, all_members=False)
    get_canonical_fullname()
    get_doc()
    get_object_members(want_all)
    get_overloaded_signatures()
    get_variable_comment()
    import_object(raiseerror=False)
    member_order = 20
        order if autodoc_member_order is set to 'groupwise'
    objtype = 'class'
        name by which the directive is called (auto...) and the default generated directive name
    option_spec: Dict[str, Callable[[str], Any]] = {'class-doc-from': <function
class_doc_from_option>, 'exclude-members': <function exclude_members_option>,
'inherited-members': <function inherited_members_option>, 'member-order':
<function member_order_option>, 'members': <function members_option>, 'noindex':
<function bool_option>, 'private-members': <function members_option>,
'show-inheritance': <function bool_option>, 'special-members': <function
members_option>, 'undoc-members': <function bool_option>}

```

```
class sage_docbuild.ext.sage_autodoc.ClassLevelDocumenter(directive: DocumenterBridge, name: str,
                                                         indent: str = "")
```

Bases: *Documenter*

Specialized Documenter subclass for objects on class level (methods, attributes).

**resolve\_name**(*modname*, *parents*, *path*, *base*)

```
class sage_docbuild.ext.sage_autodoc.DataDocumenter(directive: DocumenterBridge, name: str, indent:
                                                    str = "")
```

Bases: *GenericAliasMixin*, *NewTypeMixin*, *TypeVarMixin*, *UninitializedGlobalVariableMixin*, *ModuleLevelDocumenter*

Specialized Documenter subclass for data items.

**add\_content**(*more\_content*)

**add\_directive\_header**(*sig*)

**classmethod can\_document\_member**(*member*, *membername*, *isattr*, *parent*)

**document\_members**(*all\_members=False*)

**get\_doc**()

**get\_module\_comment**(*attrname*)

**get\_real\_modname**()

**import\_object**(*raiseerror=False*)

**member\_order** = 40

order if autodoc\_member\_order is set to 'groupwise'

**objtype** = 'data'

name by which the directive is called (auto...) and the default generated directive name

**option\_spec**: **Dict**[**str**, **Callable**[[**str**], **Any**]] = {'annotation': <function annotation\_option>, 'no-value': <function bool\_option>, 'noindex': <function bool\_option>}

**priority** = -10

priority if multiple documenters return True from can\_document\_member

**should\_suppress\_value\_header**()

**update\_annotations**(*parent*)

Update \_\_annotations\_\_ to support type\_comment and so on.

```
class sage_docbuild.ext.sage_autodoc.DataDocumenterMixinBase
```

Bases: *object*

**config**: **Config** = None

**env**: **BuildEnvironment** = None

**modname**: **str** = None

**object**: **Any** = None

```

objpath: List[str] = None

parent: Any = None

should_suppress_directive_header()
    Check directive header should be suppressed.

should_suppress_value_header()
    Check :value: header should be suppressed.

update_content(more_content)
    Update docstring for the NewType object.

```

**class** sage\_docbuild.ext.sage\_autodoc.**DecoratorDocumenter**(*directive: DocumenterBridge, name: str, indent: str = ""*)

Bases: *FunctionDocumenter*

Specialized Documenter subclass for decorator functions.

```

format_args(**kwargs)

objtype = 'decorator'
    name by which the directive is called (auto...) and the default generated directive name

priority = -1
    priority if multiple documenters return True from can_document_member

```

**class** sage\_docbuild.ext.sage\_autodoc.**DocstringSignatureMixin**

Bases: object

Mixin for FunctionDocumenter and MethodDocumenter to provide the feature of reading the signature from the docstring.

```

format_signature(**kwargs)

get_doc()

```

**class** sage\_docbuild.ext.sage\_autodoc.**DocstringStripSignatureMixin**

Bases: *DocstringSignatureMixin*

Mixin for AttributeDocumenter to provide the feature of stripping any function signature from the docstring.

```

format_signature(**kwargs)

```

**class** sage\_docbuild.ext.sage\_autodoc.**Documenter**(*directive: DocumenterBridge, name: str, indent: str = ""*)

Bases: object

A Documenter knows how to autodocument a single object type. When registered with the AutoDirective, it will be used to document objects of that type when needed by autodoc.

Its *objtype* attribute selects what auto directive it is assigned to (the directive name is 'auto' + objtype), and what directive it generates by default, though that can be overridden by an attribute called *directivetype*.

A Documenter has an *option\_spec* that works like a docutils directive's; in fact, it will be used to parse an auto directive's options that matches the Documenter.

```

add_content(more_content)
    Add content from docstrings, attribute documentation and user.

```

**add\_directive\_header**(*sig*)

Add the directive header and options to the generated content.

**add\_line**(*line, source, \*lineno*)

Append one line of generated reST to the output.

**classmethod can\_document\_member**(*member, membername, isattr, parent*)

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

**check\_module**()

Check if *self.object* is really defined in the module given by *self.modname*.

**content\_indent** = ' '

indentation by which to indent the directive content

**document\_members**(*all\_members=False*)

Generate reST for member documentation.

If *all\_members* is True, document all members, else those given by *self.options.members*.

**property documenters:** `Dict[str, Type[Documenter]]`

Returns registered Documenter classes

**filter\_members**(*members, want\_all*)

Filter the given member list.

Members are skipped if

- they are private (except if given explicitly or the private-members option is set)
- they are special methods (except if given explicitly or the special-members option is set)
- they are undocumented (except if the undoc-members option is set)

The user can override the skipping decision by connecting to the `autodoc-skip-member` event.

**format\_args**(*\*\*kwargs*)

Format the argument signature of *self.object*.

Should return None if the object does not have a signature.

**format\_name**()

Format the name of *self.object*.

This normally should be something that can be parsed by the generated directive, but doesn't need to be (Sphinx will display it unparsed then).

**format\_signature**(*\*\*kwargs*)

Format the signature (arguments and return annotation) of the object.

Let the user process it via the `autodoc-process-signature` event.

**generate**(*more\_content=None, real\_modname=None, check\_module=False, all\_members=False*)

Generate reST for the object given by *self.name*, and possibly for its members.

If *more\_content* is given, include that content. If *real\_modname* is given, use that module name to find attribute docs. If *check\_module* is True, only generate if the object is defined in the module name it is imported from. If *all\_members* is True, document all members.

**get\_attr**(*obj, name, \*defargs*)

`getattr()` override for types such as Zope interfaces.

**get\_doc()**

Decode and return lines of the docstring(s) for the object.

When it returns None, autodoc-process-docstring will not be called for this object.

**get\_object\_members(want\_all)**

Return (*members\_check\_module*, *members*) where *members* is a list of (*membername*, *member*) pairs of the members of *self.object*.

If *want\_all* is True, return all members. Else, only return those members given by *self.options.members* (which may also be None).

**get\_real\_modname()**

Get the real module name of an object to document.

It can differ from the name of the module through which the object was imported.

**get\_sourcename()****import\_object(raiseerror=False)**

Import the object given by *self.modname* and *self.objpath* and set it as *self.object*.

Returns True if successful, False if an error occurred.

**member\_order = 0**

order if autodoc\_member\_order is set to 'groupwise'

**objtype = 'object'**

name by which the directive is called (auto...) and the default generated directive name

**option\_spec: Dict[str, Callable[[str], Any]] = {'noindex': <function bool\_option>}****parse\_name()**

Determine what module to import and what attribute to document.

Returns True and sets *self.modname*, *self.objpath*, *self.fullname*, *self.args* and *self.retann* if parsing and resolving was successful.

**priority = 0**

priority if multiple documenters return True from *can\_document\_member*

**process\_doc(docstrings)**

Let the user process the docstrings before adding them.

**resolve\_name(modname, parents, path, base)**

Resolve the module and name of the object to document given by the arguments and the current module/class.

Must return a pair of the module name and a chain of attributes; for example, it would return ('zipfile', ['ZipFile', 'open']) for the *zipfile.ZipFile.open* method.

**sort\_members(documenters, order)**

Sort the given member list.

**titles\_allowed = False**

true if the generated content may contain titles

**class sage\_docbuild.ext.sage\_autodoc.ExceptionDocumenter(\*args: Any)**

Bases: *ClassDocumenter*

Specialized *ClassDocumenter* subclass for exceptions.

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

member_order = 10
    order if autodoc_member_order is set to 'groupwise'

objtype = 'exception'
    name by which the directive is called (auto...) and the default generated directive name

priority = 10
    priority if multiple documenters return True from can_document_member
```

**class** sage\_docbuild.ext.sage\_autodoc.**FunctionDocumenter**(*directive: DocumenterBridge, name: str,*  
*indent: str = ""*)

Bases: *DocstringSignatureMixin, ModuleLevelDocumenter*

Specialized Documenter subclass for functions.

```
add_directive_header(sig)

annotate_to_first_argument(func, typ)
    Annotate type hint to the first argument of function if needed.

classmethod can_document_member(member, membername, isattr, parent)

document_members(all_members=False)

format_args(**kwargs)

format_signature(**kwargs)

member_order = 30
    order if autodoc_member_order is set to 'groupwise'

merge_default_value(actual, overload)
    Merge default values of actual implementation to the overload variants.

objtype = 'function'
    name by which the directive is called (auto...) and the default generated directive name
```

**class** sage\_docbuild.ext.sage\_autodoc.**GenericAliasMixin**

Bases: *DataDocumenterMixinBase*

Mixin for DataDocumenter and AttributeDocumenter to provide the feature for supporting GenericAliases.

```
should_suppress_directive_header()

update_content(more_content)
```

sage\_docbuild.ext.sage\_autodoc.**INSTANCEATTR** = <object object>

The base class of the class hierarchy.

When called, it accepts no arguments and returns a new featureless instance that has no instance attributes and cannot be given any.

**class** sage\_docbuild.ext.sage\_autodoc.**MethodDocumenter**(*directive: DocumenterBridge, name: str,*  
*indent: str = ""*)

Bases: *DocstringSignatureMixin, ClassLevelDocumenter*

Specialized Documenter subclass for methods (normal, static and class).



```

add_directive_header(sig)

annotate_to_first_argument(func, typ)
    Annotate type hint to the first argument of function if needed.

classmethod can_document_member(member, membername, isattr, parent)

directivetype = 'method'

document_members(all_members=False)

format_args(**kwargs)

get_doc()

import_object(raiseerror=False)

member_order = 50
    order if autodoc_member_order is set to 'groupwise'

merge_default_value(actual, overload)
    Merge default values of actual implementation to the overload variants.

objtype = 'method'
    name by which the directive is called (auto...) and the default generated directive name

priority = 1
    priority if multiple documenters return True from can_document_member

class sage_docbuild.ext.sage_autodoc.ModuleDocumenter(*args: Any)
    Bases: Documenter
    Specialized Documenter subclass for modules.

    add_content(more_content)

    add_directive_header(sig)

    classmethod can_document_member(member, membername, isattr, parent)

    content_indent = ''
        indentation by which to indent the directive content

    get_module_members()
        Get members of target module.

    get_object_members(want_all)

    import_object(raiseerror=False)

    objtype = 'module'
        name by which the directive is called (auto...) and the default generated directive name

```

```
option_spec: Dict[str, Callable[[str], Any]] = {'deprecated': <function
bool_option>, 'exclude-members': <function exclude_members_option>,
'ignore-module-all': <function bool_option>, 'imported-members': <function
bool_option>, 'inherited-members': <function inherited_members_option>,
'member-order': <function member_order_option>, 'members': <function
members_option>, 'no-value': <function bool_option>, 'noindex': <function
bool_option>, 'platform': <function identity>, 'private-members': <function
members_option>, 'show-inheritance': <function bool_option>, 'special-members':
<function members_option>, 'synopsis': <function identity>, 'undoc-members':
<function bool_option>}
```

```
parse_name()
```

```
resolve_name(modname, parents, path, base)
```

```
sort_members(documenters, order)
```

```
titles_allowed = True
```

true if the generated content may contain titles

```
class sage_docbuild.ext.sage_autodoc.ModuleLevelDocumenter(directive: DocumenterBridge, name:
                                                         str, indent: str = "")
```

Bases: [Documenter](#)

Specialized Documenter subclass for objects on module level (functions, classes, data/constants).

```
resolve_name(modname, parents, path, base)
```

```
class sage_docbuild.ext.sage_autodoc.NewTypeAttributeDocumenter(directive: DocumenterBridge,
                                                                name: str, indent: str = "")
```

Bases: [AttributeDocumenter](#)

Specialized Documenter subclass for NewTypes.

Note: This must be invoked before MethodDocumenter because NewType is a kind of function object.

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

```
directivetype = 'attribute'
```

```
objtype = 'newvarattribute'
```

name by which the directive is called (auto...) and the default generated directive name

```
priority = 2
```

priority if multiple documenters return True from can\_document\_member

```
class sage_docbuild.ext.sage_autodoc.NewTypeDataDocumenter(directive: DocumenterBridge, name:
                                                            str, indent: str = "")
```

Bases: [DataDocumenter](#)

Specialized Documenter subclass for NewTypes.

Note: This must be invoked before FunctionDocumenter because NewType is a kind of function object.

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

```
directivetype = 'data'
```

**objtype** = 'newtypedata'

name by which the directive is called (auto...) and the default generated directive name

**priority** = 1

priority if multiple documenters return True from can\_document\_member

**class** sage\_docbuild.ext.sage\_autodoc.**NewTypeMixin**

Bases: *DataDocumenterMixinBase*

Mixin for DataDocumenter and AttributeDocumenter to provide the feature for supporting NewTypes.

**should\_suppress\_directive\_header**()

**update\_content**(*more\_content*)

**class** sage\_docbuild.ext.sage\_autodoc.**NonDataDescriptorMixin**

Bases: *DataDocumenterMixinBase*

Mixin for AttributeDocumenter to provide the feature for supporting non data-descriptors.

---

**Note:** This mix-in must be inherited after other mix-ins. Otherwise, docstring and :value: header will be suppressed unexpectedly.

---

**get\_doc**()

**import\_object**(*raiseerror=False*)

**should\_suppress\_value\_header**()

**class** sage\_docbuild.ext.sage\_autodoc.**ObjectMember**(*name: str, obj: Any, \*\*kwargs: Any*)

Bases: tuple

A member of object.

This is used for the result of Documenter.get\_object\_members() to represent each member of the object.

---

**Note:** An instance of this class behaves as a tuple of (name, object) for compatibility to old Sphinx. The behavior will be dropped in the future. Therefore extensions should not use the tuple interface.

---

**class** sage\_docbuild.ext.sage\_autodoc.**Options**

Bases: dict

A dict/attribute hybrid that returns None on nonexisting keys.

**copy**()

**class** sage\_docbuild.ext.sage\_autodoc.**PropertyDocumenter**(*directive: DocumenterBridge, name: str, indent: str = ""*)

Bases: *DocstringStripSignatureMixin, ClassLevelDocumenter*

Specialized Documenter subclass for properties.

**add\_directive\_header**(*sig*)

**classmethod** **can\_document\_member**(*member, membername, isattr, parent*)

**document\_members**(*all\_members=False*)

**get\_real\_modname()**

**import\_object(raiseerror=False)**

Check the existence of uninitialized instance attribute when failed to import the attribute.

**member\_order = 60**

order if autodoc\_member\_order is set to 'groupwise'

**objtype = 'property'**

name by which the directive is called (auto...) and the default generated directive name

**priority = 11**

priority if multiple documenters return True from can\_document\_member

**class sage\_docbuild.ext.sage\_autodoc.RuntimeInstanceAttributeMixin**

Bases: *DataDocumenterMixinBase*

Mixin for AttributeDocumenter to provide the feature for supporting runtime instance attributes (that are defined in `__init__()` methods with doc-comments).

Example:

**class Foo:**

**def \_\_init\_\_(self):**

self.attr = None #: This is a target of this mix-in.

**RUNTIME\_INSTANCE\_ATTRIBUTE**

**get\_doc()**

**import\_object(raiseerror=False)**

Check the existence of runtime instance attribute after failing to import the attribute.

**is\_runtime\_instance\_attribute(parent)**

Check the subject is an attribute defined in `__init__()`.

**is\_runtime\_instance\_attribute\_not\_commented(parent)**

Check the subject is an attribute defined in `__init__()` without comment.

**should\_suppress\_value\_header()**

**class sage\_docbuild.ext.sage\_autodoc.SlotsMixin**

Bases: *DataDocumenterMixinBase*

Mixin for AttributeDocumenter to provide the feature for supporting `__slots__`.

**get\_doc()**

**import\_object(raiseerror=False)**

**isslotsattribute()**

Check the subject is an attribute in `__slots__`.

**should\_suppress\_value\_header()**

**class sage\_docbuild.ext.sage\_autodoc.TypeVarMixin**

Bases: *DataDocumenterMixinBase*

Mixin for DataDocumenter and AttributeDocumenter to provide the feature for supporting TypeVars.

**get\_doc()**

**should\_suppress\_directive\_header()**

**update\_content**(*more\_content*)

**class** sage\_docbuild.ext.sage\_autodoc.UninitializedGlobalVariableMixin

Bases: [DataDocumenterMixinBase](#)

Mixin for DataDocumenter to provide the feature for supporting uninitialized (type annotation only) global variables.

**get\_doc()**

**import\_object**(*raiseerror=False*)

**should\_suppress\_value\_header()**

**class** sage\_docbuild.ext.sage\_autodoc.UninitializedInstanceAttributeMixin

Bases: [DataDocumenterMixinBase](#)

Mixin for AttributeDocumenter to provide the feature for supporting uninitialized instance attributes (PEP-526 styled, annotation only attributes).

Example:

```
class Foo:
    attr: int #: This is a target of this mix-in.
```

**get\_doc()**

**import\_object**(*raiseerror=False*)

Check the existence of uninitialized instance attribute when failed to import the attribute.

**is\_uninitialized\_instance\_attribute**(*parent*)

Check the subject is an annotation only attribute.

**should\_suppress\_value\_header()**

sage\_docbuild.ext.sage\_autodoc.**annotation\_option**(*arg*)

sage\_docbuild.ext.sage\_autodoc.**autodoc\_attrgetter**(*app, obj, name, \*defargs*)

Alternative getattr() for types

sage\_docbuild.ext.sage\_autodoc.**between**(*marker, what=None, keepempty=False, exclude=False*)

Return a listener that either keeps, or if *exclude* is True excludes, lines between lines that match the *marker* regular expression. If no line matches, the resulting docstring would be empty, so no change will be made unless *keepempty* is true.

If *what* is a sequence of strings, only docstrings of a type in *what* will be processed.

sage\_docbuild.ext.sage\_autodoc.**bool\_option**(*arg*)

Used to convert flag options to auto directives. (Instead of directives.flag(), which returns None).

sage\_docbuild.ext.sage\_autodoc.**class\_doc\_from\_option**(*arg*)

Used to convert the :class-doc-from: option to autoclass directives.

`sage_docbuild.ext.sage_autodoc.cut_lines(pre, post=0, what=None)`

Return a listener that removes the first *pre* and last *post* lines of every docstring. If *what* is a sequence of strings, only docstrings of a type in *what* will be processed.

Use like this (e.g. in the `setup()` function of `conf.py`):

```
from sphinx.ext.autodoc import cut_lines
app.connect('autodoc-process-docstring', cut_lines(4, what=['module']))
```

This can (and should) be used in place of `automodule_skip_lines`.

`sage_docbuild.ext.sage_autodoc.exclude_members_option(arg)`

Used to convert the `:exclude-members:` option.

`sage_docbuild.ext.sage_autodoc.getdoc(obj, *args, **kwargs)`

`sage_docbuild.ext.sage_autodoc.identity(x)`

`sage_docbuild.ext.sage_autodoc.inherited_members_option(arg)`

Used to convert the `:inherited-members:` option to auto directives.

`sage_docbuild.ext.sage_autodoc.member_order_option(arg)`

Used to convert the `:member-order:` option to auto directives.

`sage_docbuild.ext.sage_autodoc.members_option(arg)`

Used to convert the `:members:` option to auto directives.

`sage_docbuild.ext.sage_autodoc.merge_members_option(options)`

Merge `:private-members:` and `:special-members:` options to the `:members:` option.

```
sage_docbuild.ext.sage_autodoc.py_ext_sig_re = re.compile('^ ([\\w.]+::)? # explicit
module name\\n ([\\w.]+\\.)? # module and/or class name(s)\\n (\\w+) \\s* # thing name\\n
(?: \\((.*)\\) , re.VERBOSE)
```

extended signature RE: with explicit module name separated by `::`.

`sage_docbuild.ext.sage_autodoc.setup(app)`

## 2.3 Sage multidocs extension

The goal of this extension is to manage a multi-documentation in Sphinx. To be able to compile Sage’s huge documentation in parallel, the documentation is cut into a bunch of independent documentations called “sub-docs”, which are compiled separately. There is a master document which points to all the sub-docs. The intersphinx extension ensures that the cross-link between the sub-docs are correctly resolved. However some work is needed to build a global index. This is the goal of the `multidocs` extension.

More precisely this extension ensures the correct merging of

- the todo list if this extension is activated
- the python indexes
- the list of python modules
- the javascript index
- the citations

`sage_docbuild.ext.multidocs.citation_dir(app)`

`sage_docbuild.ext.multidocs.fetch_citation(app, env)`

Fetch the global citation index from the refman to allow for cross references.

`sage_docbuild.ext.multidocs.fix_path_html(app, pagename, templatenam, ctx, event_arg)`

Fix the context so that the files

- `search.html`
- `genindex.html`
- `py-modindex.html`

point to the right place, that is in `reference/` instead of `reference/subdocument`.

`sage_docbuild.ext.multidocs.get_env(app, curdoc)`

Get the environment of a sub-doc from the pickle

`sage_docbuild.ext.multidocs.get_js_index(app, curdoc)`

Get the JS index of a sub-doc from the file

`sage_docbuild.ext.multidocs.init_subdoc(app)`

Init the merger depending on if we are compiling a sub-doc or the master doc itself.

`sage_docbuild.ext.multidocs.merge_environment(app, env)`

Merge the following attributes of the sub-docs environment into the main environment:

- `titles` – Titles
- `todo_all_todos` – todo's
- `indexentries` – global python index
- `all_docs` – needed by the js index
- `citations` – citations
- `domaingroup['py']['modules']` – list of python modules

`sage_docbuild.ext.multidocs.merge_js_index(app)`

Merge the JS indexes of the sub-docs into the main JS index

`sage_docbuild.ext.multidocs.setup(app)`

`sage_docbuild.ext.multidocs.write_citations(app, citations)`

Pickle the citation in a file.





## PYTHON MODULE INDEX

### S

- `sage_docbuild.__main__`, 1
- `sage_docbuild.build_options`, 8
- `sage_docbuild.builders`, 3
- `sage_docbuild.conf`, 9
- `sage_docbuild.ext.inventory_builder`, 13
- `sage_docbuild.ext.multidocs`, 26
- `sage_docbuild.ext.sage_autodoc`, 14
- `sage_docbuild.sphinxbuild`, 8
- `sage_docbuild.utils`, 10



## INDEX

### A

`add_content()` (*sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* method), 14  
`add_content()` (*sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* method), 15  
`add_content()` (*sage\_docbuild.ext.sage\_autodoc.DataDocumenter* method), 16  
`add_content()` (*sage\_docbuild.ext.sage\_autodoc.Documenter* method), 17  
`add_content()` (*sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter* method), 21  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* method), 14  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* method), 15  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.DataDocumenter* method), 16  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.Documenter* method), 17  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter* method), 20  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.MethodDocumenter* method), 20  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter* method), 21  
`add_directive_header()` (*sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter* method), 23  
`add_line()` (*sage\_docbuild.ext.sage\_autodoc.Documenter* method), 18  
`add_page_context()` (in module *sage\_docbuild.conf*), 9  
`AllBuilder` (class in *sage\_docbuild.builders*), 4  
`annotate_to_first_argument()` (*sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter* method), 20  
`annotate_to_first_argument()` (*sage\_docbuild.ext.sage\_autodoc.MethodDocumenter* method), 21  
`annotation_option()` (in module *sage\_docbuild.ext.sage\_autodoc*), 25  
`ansi_color()` (*sage\_docbuild.sphinxbuild.SageSphinxLogger* attribute), 8  
`ansi_reset()` (*sage\_docbuild.sphinxbuild.SageSphinxLogger* attribute), 8  
`apply()` (*sage\_docbuild.conf.SagecodeTransform* method), 9  
`AttributeDocumenter` (class in *sage\_docbuild.ext.sage\_autodoc*), 14  
`auto_rest_filename()` (*sage\_docbuild.builders.ReferenceSubBuilder* method), 5  
`autodoc_attrgetter()` (in module *sage\_docbuild.ext.sage\_autodoc*), 25

### B

`between()` (in module *sage\_docbuild.ext.sage\_autodoc*), 25  
`bool_option()` (in module *sage\_docbuild.ext.sage\_autodoc*), 25  
`build_many()` (in module *sage\_docbuild.builders*), 7  
`build_many()` (in module *sage\_docbuild.utils*), 11  
`build_other_doc()` (in module *sage\_docbuild.builders*), 7  
`build_ref_doc()` (in module *sage\_docbuild.builders*), 7  
`builder_helper()` (in module *sage\_docbuild.builders*), 7

### C

`cache_filename()` (*sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
`call_intersphinx()` (in module *sage\_docbuild.conf*), 9  
`can_document_member()` (*sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* class method), 14

`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter class method), 15  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.DataDocumenter class method), 16  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.Documenter class method), 18  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.ExceptionDocumenter class method), 19  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter class method), 20  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.MethodDocumenter class method), 21  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter class method), 21  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.NewTypeAttributeDocumenter class method), 22  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.NewTypeDataDocumenter class method), 22  
`can_document_member()` (sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter class method), 23  
`changes()` (sage\_docbuild.builders.DocBuilder method), 4  
`check_module()` (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
`check_nested_class_picklability()` (in module sage\_docbuild.conf), 9  
`citation_dir()` (in module sage\_docbuild.ext.multidocs), 26  
`class_doc_from_option()` (in module sage\_docbuild.ext.sage\_autodoc), 25  
`ClassDocumenter` (class in sage\_docbuild.ext.sage\_autodoc), 15  
`ClassLevelDocumenter` (class in sage\_docbuild.ext.sage\_autodoc), 15  
`clean()` (sage\_docbuild.builders.DocBuilder method), 4  
`clean()` (sage\_docbuild.builders.WebsiteBuilder method), 7  
`clean_auto()` (sage\_docbuild.builders.ReferenceSubBuilder method), 6  
`close()` (sage\_docbuild.sphinxbuild.SageSphinxLogger method), 8  
`closed` (sage\_docbuild.sphinxbuild.SageSphinxLogger attribute), 8  
`config` (sage\_docbuild.ext.sage\_autodoc.DataDocumenterMixinBase attribute), 16  
`content_indent` (sage\_docbuild.ext.sage\_autodoc.Documenter attribute), 18  
`content_indent` (sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter attribute), 21  
`copy()` (sage\_docbuild.ext.sage\_autodoc.Options method), 23  
`cut_lines()` (in module sage\_docbuild.ext.sage\_autodoc), 25

## D

`DataDocumenter` (class in sage\_docbuild.ext.sage\_autodoc), 16  
`DataDocumenterMixinBase` (class in sage\_docbuild.ext.sage\_autodoc), 16  
`debug_inf()` (in module sage\_docbuild.conf), 10  
`DecoratorDocumenter` (class in sage\_docbuild.ext.sage\_autodoc), 17  
`default_priority` (sage\_docbuild.conf.SagecodeTransform attribute), 9  
`directivetype` (sage\_docbuild.ext.sage\_autodoc.MethodDocumenter attribute), 21  
`directivetype` (sage\_docbuild.ext.sage\_autodoc.NewTypeAttributeDocumenter attribute), 22  
`directivetype` (sage\_docbuild.ext.sage\_autodoc.NewTypeDataDocumenter attribute), 22  
`DocBuilder` (class in sage\_docbuild.builders), 4  
`DocstringSignatureMixin` (class in sage\_docbuild.ext.sage\_autodoc), 17  
`DocstringStripSignatureMixin` (class in sage\_docbuild.ext.sage\_autodoc), 17  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter method), 14  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter method), 15  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.DataDocumenter method), 16  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter method), 20  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.MethodDocumenter method), 21  
`document_members()` (sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter method), 23  
`Documenter` (class in sage\_docbuild.ext.sage\_autodoc), 17  
`documenters` (sage\_docbuild.ext.sage\_autodoc.Documenter property), 18

## E

`encoding` (sage\_docbuild.sphinxbuild.SageSphinxLogger attribute), 8

env (sage\_docbuild.ext.sage\_autodoc.DataDocumenterMixin attribute), 16  
 epilog (sage\_docbuild.ext.inventory\_builder.InventoryBuilder attribute), 13  
 exc (sage\_docbuild.utils.RemoteExceptionWrapper attribute), 10  
 ExceptionDocumenter (class in sage\_docbuild.ext.sage\_autodoc), 19  
 exclude\_members\_option() (in module sage\_docbuild.ext.sage\_autodoc), 26  
**F**  
 fetch\_citation() (in module sage\_docbuild.ext.multidocs), 26  
 fetch\_inventory() (sage\_docbuild.\_\_main\_\_.IntersphinxCache method), 2  
 filter\_members() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 find\_sage\_dangling\_links() (in module sage\_docbuild.conf), 10  
 finish() (sage\_docbuild.ext.inventory\_builder.InventoryBuilder method), 13  
 fix\_path\_html() (in module sage\_docbuild.ext.multidocs), 27  
 flush() (sage\_docbuild.sphinxbuild.SageSphinxLogger method), 8  
 format (sage\_docbuild.ext.inventory\_builder.InventoryBuilder attribute), 13  
 format\_args() (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter method), 15  
 format\_args() (sage\_docbuild.ext.sage\_autodoc.DecoratorDocumenter method), 17  
 format\_args() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 format\_args() (sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter method), 20  
 format\_args() (sage\_docbuild.ext.sage\_autodoc.MethodDocumenter method), 21  
 format\_columns() (in module sage\_docbuild.\_\_main\_\_), 2  
 format\_name() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 format\_signature() (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter method), 15  
 format\_signature() (sage\_docbuild.ext.sage\_autodoc.DocstringSignatureMixin method), 17  
 format\_signature() (sage\_docbuild.ext.sage\_autodoc.DocstringSignatureMixin method), 17  
 format\_signature() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 format\_signature() (sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter method), 20  
 FunctionDocumenter (class in sage\_docbuild.ext.sage\_autodoc), 20  
 generate() (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter method), 15  
 generate() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 GenericAliasMixin (class in sage\_docbuild.ext.sage\_autodoc), 20  
 get\_all\_documents() (sage\_docbuild.builders.AllBuilder method), 4  
 get\_all\_documents() (sage\_docbuild.builders.ReferenceBuilder method), 5  
 get\_all\_included\_modules() (sage\_docbuild.builders.ReferenceSubBuilder method), 6  
 get\_all\_rst\_files() (sage\_docbuild.builders.ReferenceSubBuilder method), 6  
 get\_attr() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 get\_builder\_attribute\_comment() (sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter method), 14  
 get\_builder() (in module sage\_docbuild.builders), 7  
 get\_cache() (sage\_docbuild.builders.ReferenceSubBuilder method), 6  
 get\_canonical\_fullname() (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter method), 15  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter method), 14  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter method), 15  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.DataDocumenter method), 16  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.DocstringSignatureMixin method), 17  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.Documenter method), 18  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.MethodDocumenter method), 21  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.NonDataDescriptorMixin method), 23  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.RuntimeInstanceAttributeMixin method), 24  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.SlotsMixin method), 24  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.TypeVarMixin method), 24  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.UninitializedGlobalVariableMixin method), 25  
 get\_doc() (sage\_docbuild.ext.sage\_autodoc.UninitializedInstanceAttributeMixin method), 25  
 get\_documents() (in module sage\_docbuild.builders), 4

7  
 get\_env() (in module *sage\_docbuild.ext.multidocs*), 27  
 get\_formats() (in module *sage\_docbuild.\_\_main\_\_*), 2  
 get\_js\_index() (in module *sage\_docbuild.ext.multidocs*), 27  
 get\_modified\_modules() (in module *sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
 get\_module\_comment() (in module *sage\_docbuild.ext.sage\_autodoc.DataDocumenter* method), 16  
 get\_module\_docstring\_title() (in module *sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
 get\_module\_members() (in module *sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter* method), 21  
 get\_modules() (in module *sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
 get\_new\_and\_updated\_modules() (in module *sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
 get\_object\_members() (in module *sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* method), 15  
 get\_object\_members() (in module *sage\_docbuild.ext.sage\_autodoc.Documenter* method), 19  
 get\_object\_members() (in module *sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter* method), 21  
 get\_outdated\_docs() (in module *sage\_docbuild.ext.inventory\_builder.InventoryBuilder* method), 13  
 get\_overloaded\_signatures() (in module *sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* method), 15  
 get\_real\_modname() (in module *sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* method), 14  
 get\_real\_modname() (in module *sage\_docbuild.ext.sage\_autodoc.DataDocumenter* method), 16  
 get\_real\_modname() (in module *sage\_docbuild.ext.sage\_autodoc.Documenter* method), 19  
 get\_real\_modname() (in module *sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter* method), 23  
 get\_sourcename() (in module *sage\_docbuild.ext.sage\_autodoc.Documenter* method), 19  
 get\_sphinx\_environment() (in module *sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
 get\_target\_uri() (in module *sage\_docbuild.ext.inventory\_builder.InventoryBuilder* method), 13  
 get\_unincluded\_modules() (in module *sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
 get\_variable\_comment() (in module *sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* method), 15  
 getdoc() (in module *sage\_docbuild.ext.sage\_autodoc*), 26  
 H  
 has() (in module *sage\_docbuild.conf.Tags* method), 9  
 help\_commands() (in module *sage\_docbuild.\_\_main\_\_*), 2  
 help\_description() (in module *sage\_docbuild.\_\_main\_\_*), 2  
 help\_documents() (in module *sage\_docbuild.\_\_main\_\_*), 2  
 help\_examples() (in module *sage\_docbuild.\_\_main\_\_*), 2  
 help\_formats() (in module *sage\_docbuild.\_\_main\_\_*), 2  
 help\_message\_long (in module *sage\_docbuild.\_\_main\_\_*), 2  
 help\_message\_short (in module *sage\_docbuild.\_\_main\_\_*), 3  
 help\_usage() (in module *sage\_docbuild.\_\_main\_\_*), 3  
 help\_wrapper (in module *sage\_docbuild.\_\_main\_\_*), 3  
 html() (in module *sage\_docbuild.builders.DocBuilder* method), 4  
 html() (in module *sage\_docbuild.builders.WebsiteBuilder* method), 7  
 htmlhelp() (in module *sage\_docbuild.builders.DocBuilder* method), 4  
 I  
 identity() (in module *sage\_docbuild.ext.sage\_autodoc*), 26  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* method), 14  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* method), 15  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.DataDocumenter* method), 16  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.Documenter* method), 19  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.MethodDocumenter* method), 21  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter* method), 21  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.NonDataDescriptorM* method), 23  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter* method), 24  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.RuntimeInstanceAttri* method), 24  
 import\_object() (in module *sage\_docbuild.ext.sage\_autodoc.SlotsMixin* method), 24



[import\\_object\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.UninitializedInstanceAttribute* method), 25  
[import\\_object\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.UninitializedInstanceAttribute* method), 25  
[inherited\\_members\\_option\(\)](#) (in module *sage\_docbuild.ext.sage\_autodoc*), 26  
[init\\_subdoc\(\)](#) (in module *sage\_docbuild.ext.multidocs*), 27  
[INSTANCEATTR](#) (in module *sage\_docbuild.ext.sage\_autodoc*), 20  
[IntersphinxCache](#) (class in *sage\_docbuild.\_\_main\_\_*), 2  
[inventory\(\)](#) (*sage\_docbuild.builders.DocBuilder* method), 4  
[InventoryBuilder](#) (class in *sage\_docbuild.ext.inventory\_builder*), 13  
[is\\_function\\_or\\_method\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* static method), 14  
[is\\_runtime\\_instance\\_attribute\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.RuntimeInstanceAttribute* method), 24  
[is\\_runtime\\_instance\\_attribute\\_not\\_commented\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.RuntimeInstanceAttribute* method), 24  
[is\\_uninitialized\\_instance\\_attribute\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.UninitializedInstanceAttribute* method), 25  
[isatty\(\)](#) (*sage\_docbuild.sphinxbuild.SageSphinxLogger* method), 8  
[isslotsattribute\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.SlotsMixin* method), 24

**J**

[json\(\)](#) (*sage\_docbuild.builders.DocBuilder* method), 4

**L**

[latex\(\)](#) (*sage\_docbuild.builders.DocBuilder* method), 4  
[linkcheck\(\)](#) (*sage\_docbuild.builders.DocBuilder* method), 4

**M**

[main\(\)](#) (in module *sage\_docbuild.\_\_main\_\_*), 3  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* attribute), 14  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.ClassDocumenter* attribute), 15  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.DataDocumenter* attribute), 16  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.Documenter* attribute), 19  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.ExceptionDocumenter* attribute), 20

[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter* attribute), 20  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.MethodDocumenter* attribute), 21  
[member\\_order](#) (*sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter* attribute), 24  
[member\\_order\\_option\(\)](#) (in module *sage\_docbuild.ext.sage\_autodoc*), 26  
[members\\_option\(\)](#) (in module *sage\_docbuild.ext.sage\_autodoc*), 26  
[merge\\_default\\_value\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter* method), 20  
[merge\\_default\\_value\(\)](#) (*sage\_docbuild.ext.sage\_autodoc.MethodDocumenter* method), 21  
[merge\\_environment\(\)](#) (in module *sage\_docbuild.ext.multidocs*), 27  
[merge\\_js\\_index\(\)](#) (in module *sage\_docbuild.ext.multidocs*), 27  
[merge\\_members\\_option\(\)](#) (in module *sage\_docbuild.ext.sage\_autodoc*), 26  
[MixinBase](#) (class in *sage\_docbuild.ext.sage\_autodoc*), 20  
[mode](#) (*sage\_docbuild.sphinxbuild.SageSphinxLogger* attribute), 8  
[module\\_name](#) (*sage\_docbuild.ext.sage\_autodoc.DataDocumenter* attribute), 16

[ModuleDocumenter](#) (class in *sage\_docbuild.\_\_main\_\_*), 1  
[sage\\_docbuild.build\\_options](#), 8  
[sage\\_docbuild.builders](#), 3  
[sage\\_docbuild.conf](#), 9  
[sage\\_docbuild.ext.inventory\\_builder](#), 13  
[sage\\_docbuild.ext.multidocs](#), 26  
[sage\\_docbuild.ext.sage\\_autodoc](#), 14  
[sage\\_docbuild.sphinxbuild](#), 8  
[sage\\_docbuild.utils](#), 10

[ModuleDocumenter](#) (class in *sage\_docbuild.ext.sage\_autodoc*), 21  
[ModuleLevelDocumenter](#) (class in *sage\_docbuild.ext.sage\_autodoc*), 22

**N**

[name](#) (*sage\_docbuild.ext.inventory\_builder.InventoryBuilder* attribute), 13  
[name](#) (*sage\_docbuild.sphinxbuild.SageSphinxLogger* attribute), 8  
[newlines](#) (*sage\_docbuild.sphinxbuild.SageSphinxLogger* attribute), 8  
[NewTypeAttributeDocumenter](#) (class in *sage\_docbuild.ext.sage\_autodoc*), 22  
[NewTypeDataDocumenter](#) (class in *sage\_docbuild.ext.sage\_autodoc*), 22

NewTypeMixin (class in attribute), 17  
     sage\_docbuild.ext.sage\_autodoc), 23  
 nitpick\_patch\_config() (in module method), 19  
     sage\_docbuild.conf), 10  
 NonDataDescriptorMixin (class in method), 22  
     sage\_docbuild.ext.sage\_autodoc), 23  
 pdf() (sage\_docbuild.builders.DocBuilder method), 5  
 pdf() (sage\_docbuild.builders.ReferenceTopBuilder method), 7  
 pdf() (sage\_docbuild.builders.WebsiteBuilder method), 7  
 object (sage\_docbuild.ext.sage\_autodoc.DataDocumenterMixinBase attribute), 16  
 ObjectMember (class in pickle() (sage\_docbuild.builders.DocBuilder method), 5  
     sage\_docbuild.ext.sage\_autodoc), 23  
 objpath (sage\_docbuild.ext.sage\_autodoc.DataDocumenterMixinBase attribute), 16  
 objtype (sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter attribute), 14  
 objtype (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter attribute), 15  
 objtype (sage\_docbuild.ext.sage\_autodoc.DataDocumenter attribute), 16  
 objtype (sage\_docbuild.ext.sage\_autodoc.DecoratorDocumenter attribute), 17  
 objtype (sage\_docbuild.ext.sage\_autodoc.Documenter attribute), 19  
 objtype (sage\_docbuild.ext.sage\_autodoc.ExceptionDocumenter attribute), 20  
 objtype (sage\_docbuild.ext.sage\_autodoc.FunctionDocumenter attribute), 20  
 objtype (sage\_docbuild.ext.sage\_autodoc.MethodDocumenter attribute), 21  
 objtype (sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter attribute), 21  
 objtype (sage\_docbuild.ext.sage\_autodoc.NewTypeAttributeDocumenter attribute), 22  
 objtype (sage\_docbuild.ext.sage\_autodoc.NewTypeDataDocumenter attribute), 22  
 objtype (sage\_docbuild.ext.sage\_autodoc.PropertyDocumenter attribute), 24  
 option\_spec (sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter attribute), 14  
 option\_spec (sage\_docbuild.ext.sage\_autodoc.ClassDocumenter attribute), 15  
 option\_spec (sage\_docbuild.ext.sage\_autodoc.DataDocumenter attribute), 16  
 option\_spec (sage\_docbuild.ext.sage\_autodoc.Documenter attribute), 19  
 option\_spec (sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter attribute), 21  
 Options (class in sage\_docbuild.ext.sage\_autodoc), 23  
 original\_exception (sage\_docbuild.utils.WorkerDiedException attribute), 11

## R

raise\_errors() (sage\_docbuild.sphinxbuild.SageSphinxLogger method), 8  
 parent (sage\_docbuild.ext.sage\_autodoc.DataDocumenterMixinBase attribute), 16



[ReferenceBuilder](#) (class in [sage\\_docbuild.builders](#)), 5  
[ReferenceSubBuilder](#) (class in [sage\\_docbuild.builders](#)), 5  
[ReferenceTopBuilder](#) (class in [sage\\_docbuild.builders](#)), 7  
[RemoteException](#), 10  
[RemoteExceptionWrapper](#) (class in [sage\\_docbuild.utils](#)), 10  
[resolve\\_name\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.ClassLevelDocumenter](#) method), 16  
[resolve\\_name\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.Documenter](#) method), 19  
[resolve\\_name\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.ModuleDocumenter](#) method), 22  
[resolve\\_name\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.ModuleLevelDocumenter](#) method), 22  
[runspphinx\(\)](#) (in module [sage\\_docbuild.sphinxbuild](#)), 8  
[RUNTIME\\_INSTANCE\\_ATTRIBUTE](#) ([sage\\_docbuild.ext.sage\\_autodoc.RuntimeInstanceAttributeMixin](#) attribute), 24  
[RuntimeInstanceAttributeMixin](#) (class in [sage\\_docbuild.ext.sage\\_autodoc](#)), 24  
**S**  
[sage\\_docbuild.\\_\\_main\\_\\_](#) module, 1  
[sage\\_docbuild.build\\_options](#) module, 8  
[sage\\_docbuild.builders](#) module, 3  
[sage\\_docbuild.conf](#) module, 9  
[sage\\_docbuild.ext.inventory\\_builder](#) module, 13  
[sage\\_docbuild.ext.multidocs](#) module, 26  
[sage\\_docbuild.ext.sage\\_autodoc](#) module, 14  
[sage\\_docbuild.sphinxbuild](#) module, 8  
[sage\\_docbuild.utils](#) module, 10  
[SagecodeTransform](#) (class in [sage\\_docbuild.conf](#)), 9  
[SageSphinxLogger](#) (class in [sage\\_docbuild.sphinxbuild](#)), 8  
[save\\_cache\(\)](#) ([sage\\_docbuild.builders.ReferenceSubBuilder](#) method), 6  
[set\\_intersphinx\\_mappings\(\)](#) (in module [sage\\_docbuild.conf](#)), 10  
[setup\(\)](#) (in module [sage\\_docbuild.conf](#)), 10  
[setup\(\)](#) (in module [sage\\_docbuild.ext.inventory\\_builder](#)), 13  
[setup\(\)](#) (in module [sage\\_docbuild.ext.multidocs](#)), 27  
[setup\(\)](#) (in module [sage\\_docbuild.ext.sage\\_autodoc](#)), 26  
[setup\\_logger\(\)](#) (in module [sage\\_docbuild.\\_\\_main\\_\\_](#)), 3  
[setup\\_parser\(\)](#) (in module [sage\\_docbuild.\\_\\_main\\_\\_](#)), 3  
[should\\_suppress\\_directive\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.DataDocumenterMixinBase](#) method), 17  
[should\\_suppress\\_directive\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.GenericAliasMixin](#) method), 20  
[should\\_suppress\\_directive\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.NewTypeMixin](#) method), 23  
[should\\_suppress\\_directive\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.TypeVarMixin](#) method), 25  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.AttributeDocumenter](#) method), 15  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.DataDocumenter](#) method), 16  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.DataDocumenterMixinBase](#) method), 17  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.NonDataDescriptorMixin](#) method), 23  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.RuntimeInstanceAttributeMixin](#) method), 24  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.SlotsMixin](#) method), 24  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.UninitializedGlobalVariableMixin](#) method), 25  
[should\\_suppress\\_value\\_header\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.UninitializedInstanceAttributeMixin](#) method), 25  
[SingleFileBuilder](#) (class in [sage\\_docbuild.builders](#)), 7  
[skip\\_member\(\)](#) (in module [sage\\_docbuild.conf](#)), 10  
[SlotsMixin](#) (class in [sage\\_docbuild.ext.sage\\_autodoc](#)), 24  
[softspace](#) ([sage\\_docbuild.sphinxbuild.SageSphinxLogger](#) attribute), 8  
[sort\\_members\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.Documenter](#) method), 19  
[sort\\_members\(\)](#) ([sage\\_docbuild.ext.sage\\_autodoc.ModuleDocumenter](#) method), 22  
**T**  
[Tags](#) (class in [sage\\_docbuild.conf](#)), 9

`tb` (*sage\_docbuild.utils.RemoteException* attribute), 10  
`tb` (*sage\_docbuild.utils.RemoteExceptionWrapper* attribute), 10  
`term_width_line()` (in module *sage\_docbuild.sphinxbuild*), 9  
`titles_allowed` (*sage\_docbuild.ext.sage\_autodoc.Documenter* attribute), 19  
`titles_allowed` (*sage\_docbuild.ext.sage\_autodoc.ModuleDocumenter* attribute), 22  
`TypeVarMixin` (class in *sage\_docbuild.ext.sage\_autodoc*), 24

## U

`UninitializedGlobalVariableMixin` (class in *sage\_docbuild.ext.sage\_autodoc*), 25  
`UninitializedInstanceAttributeMixin` (class in *sage\_docbuild.ext.sage\_autodoc*), 25  
`update_annotations()` (*sage\_docbuild.ext.sage\_autodoc.AttributeDocumenter* method), 15  
`update_annotations()` (*sage\_docbuild.ext.sage\_autodoc.DataDocumenter* method), 16  
`update_content()` (*sage\_docbuild.ext.sage\_autodoc.DataDocumenterMixinBase* method), 17  
`update_content()` (*sage\_docbuild.ext.sage\_autodoc.GenericAliasMixin* method), 20  
`update_content()` (*sage\_docbuild.ext.sage\_autodoc.NewTypeMixin* method), 23  
`update_content()` (*sage\_docbuild.ext.sage\_autodoc.TypeVarMixin* method), 25  
`update_mtimes()` (*sage\_docbuild.builders.ReferenceSubBuilder* method), 6

## W

`web()` (*sage\_docbuild.builders.DocBuilder* method), 5  
`WebsiteBuilder` (class in *sage\_docbuild.builders*), 7  
`WorkerDiedException`, 10  
`write()` (*sage\_docbuild.sphinxbuild.SageSphinxLogger* method), 8  
`write_auto_rest_file()` (*sage\_docbuild.builders.ReferenceSubBuilder* method), 6  
`write_citations()` (in module *sage\_docbuild.ext.multidocs*), 27  
`writelines()` (*sage\_docbuild.sphinxbuild.SageSphinxLogger* method), 8