sphinx-collections

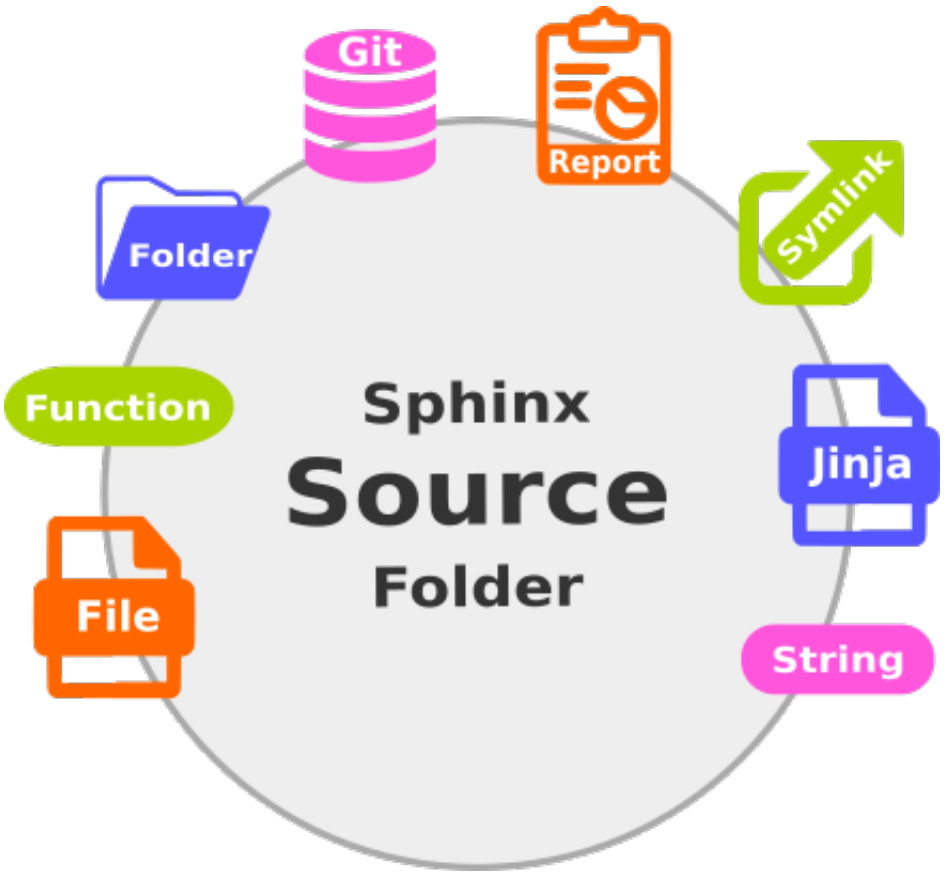
team useblocks

May 26, 2020

CONTENTS:

`Sphinx-Collections` is a Sphinx extension to collect and generate additional files from different sources. These files are added to the Sphinx Source Folder, so that Sphinx takes them into account for the overall documentation build.

`Sphinx Collections` supports multiple collections, where each collection has its own source and target folder, specific configuration and use case based driver.



A collection can be activated by default or its usage can be triggered by Sphinx tags.

Depending on the usage of a specific collection for a build, its content integration can be controlled by the if-collection:: directive .

Following use cases are supported:

- Create file with content from string
- Create file with content from function call
- Copy single file from local path
- Copy folder tree from local path
- Create a symlink to a local target

- Create a usage-report of collections
- Clone git repository
- Create multiple files based on jinja-template and specific data

`Sphinx-Collections` cares about keeping your collection folders clean before and after each build.

CHAPTER

ONE

INTRODUCTION

`Sphinx-Collections` gets completely configured by variables inside the `conf.py` file of your Sphinx project:

```
collections = {
   'my_files': {
      'driver': 'copy_folder',
      'source': '../../extra_files/'
   }
}
```

The driver `copy_folder` allows to copy local folders and their files into your Sphinx project. There are other drivers available, which support different use cases and and file locations.

By default all files get copied to `_collections/` + `collection_name`, so in this example the complete path inside your documentation folder would be `_collections/my_files/`. The location can be set specific for each collection by using `target` option.

Then you can reference the copied files by using a toctree:

```
.. toctree::
   _collections/my_files/index
```

Please see the documentation of the needed Driver to know which options are available and necessary.

CHAPTER

TWO

TAG BASED COLLECTIONS

Use Sphinx tags to collect and integrate only needed data:

```
 collections = {
   'my_files': {
      'driver': 'copy',
      'source': '../../extra_files/',
      'tags': ['user_manual'],  # gets active, if "user_manual" is set as tag
      'active': False,  # by default, collection shall not be executed
   }
}
```

Then run `sphinx-build` with `-t` option:

```
sphinx-build -b html -t user_manual . _build/html
```

CHAPTER

THREE

COLLECTION BASED CONTENT

Use `if-collection` to add content to a page only, if a specified collections has been executed successfully.

```
.. if-collection:: my_test, my_data

   My Test & Data chapter
   ----------------------

    .. toctree::

      /_collections/my_test/index
      /_collections/my_data/index
```

For more information take a look into the documentation of if-collection.

CHAPTER

FOUR

MOTIVATION

This sphinx extension is based on the needs of a software development team inside a german automotive company.

The project team was searching for a practical way to support multiple sphinx-based documentations inside a mono-repository and have the possibility to merge different documentations together or to add files based on external data.

Sphinx-Collections is part of a software bundle, which was designed to support the development of ISO 26262 compliant software. Other tools are: sphinx-needs, sphinx-test-reports, tox-envreport.

4.1 Collections

Collections are defined by setting *collections* in `conf.py`:

```
collections = {
      '<COLLECTION_NAME>': {
         'driver': '<DRIVER>',
         'source': 'source path',
         '<COLLECTION_OPTION>': 'any value'
      }
   }
```

`<COLLECTION_NAME>` must be a string, which can also be used as folder name. So try to avoid special characters or other hashable objects.

`driver` must always be set and match a registered *Driver*.

Also `source` must always be specified. Its content is validated and used by the driver.

`<COLLECTION_OPTION>` should also be a string, which matches one of the options below. If it does not match, it gets ignored.

4.1.1 Collection Options

Most of the below options have a default value. Therefore setting them inside your configuration is not needed, if you are happy with the related default value.

However, options which are needed by drivers normally don't have a default value. So please take a look into the related *driver* configuration to see what is needed and supported.

driver

Specifies the driver to use. Must be a string.

If not set or driver is unknown, an exception gets thrown.

For a complete list of drivers, please see *Drivers*.

Mandatory: Yes

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
    }
}
```

source

String of the `source` to use.

Depending on the used driver, this can be a folder, file, a git repository or whatever.

Mandatory: Yes

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
    }
}
```

target

Target path, where to store the new files. Depending on the *driver* this must be a folder or a single file.

Can be an absolute path or a relative path, which starts from the folder set by *collections_target*.

If not set, the collection name is used as target name.

Example:

If *collections_target* has default value `_collections` and collection is named `my_first_collection`, then target is set to `_collections/my_first_collection` inside your documentation project.

Hint: `target` must always be somewhere inside your documentation folder (where your `conf.py` is stored). Targets outside of your documentation are not supported.

Default: collection name

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
        'target': 'custom_folder/folder_x/'
    }
}
```

active

`active` can be set to `True` or `False`. If set to `False`, the collection gets completely ignored during documentation build.

Default: `True`

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
        'active': False
    }
}
```

safe

Takes a boolean value and if it is set to `True` any problem will raise an exception and stops the build.

Default: `True`

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'safe': False,
    }
}
```

clean

If set to `False`, no clean-up is taking place before collections get executed.

Default value can be changed for all collections by setting *collections_clean*

Default: `True`

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
        'clean': False
    }
}
```

final_clean

If set to `True`, a final clean up at the end of a Sphinx build is executed.

Often used to keep your working tree clean and have collected files only during build in related folders.

Default value can be changed for all collections by setting *collections_final_clean*.

Default: `True`

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
        'final_clean': False
    }
}
```

tags

List of tags, which trigger an activation of the collection. Should be used together with *active* set to `False`, otherwise the collection gets always executed.

```
collections = {
    'my_collection': {
        'driver': 'copy_folder',
        'source': 'source path',
        'active': False,
        'tags': ['my_collection', 'dummy']
    }
}
```

Use `-t tag` option of `sphinx-build` command to trigger related collections.

```
sphinx-build -b html -t dummy . _build/html
```

4.1.2 Driver Options

Options for drivers are also stored directly with the configuration for collections.

Please take a look into the specific *Driver* to get information about its additional configuration possibilities.

4.2 Drivers

Drivers represents the technical function, which gets configured by the configuration given by a collection.

Each collection must reference a single driver, which cares about:

- Initial clean up
- Configured execution
- Final clean up

`Sphinx-Collections` already provides some major drivers, which support different use cases.

4.2.1 copy_file

Copies a a single from `source` to `project`. Both should should have a valid file name in it.

```
 collections = {
   'my_files: {
      'driver': 'copy_file',
      'source': '../extra_files/my_file.txt',
      'target': 'my_data/new_data.txt'
      }
   }
}
```

Clean up behavior

During clean up the target file gets deleted.

4.2.2 copy_folder

Copies a folder tree from `source` into your documentation project:

```
collections = {
    'my_files': {
        'driver': 'copy_folder',
        'source': '../../extra_files/',
        'target': 'my_data/',
        'ignore': ['*.dat', '.exe'],
    }
}
```

Options

ignore

List of file matches, which shall get ignored from copy.

This variable is internally given to shutil.ignore_patterns. So it must follow its syntax rules.

Clean up behavior

During clean up the target folder gets deleted.

4.2.3 function

Executes a function referenced by `source` and writes its return value into a file specified by `target`.

```
 def my_own_data(config):
     string = 'This data gets written into {}'.format(config['target'])

     return string

 collections = {
   'my_files: {
      'driver': 'function',
      'source': my_own_data,
      'target': 'my_data/my_file.txt'
      'write_result': True
      }
   }
}
```

The specified function gets 1 argument during the call: A dictionary which contains the complete configuration of the collection.

If return value is not None, the returned data is written to the file specified by `target`.

Options

write_result

If `write_result` is False, no data is written by the driver. But this could be done by the function itself.

Default: `True`

Clean up behavior

The target folder/file gets deleted.

4.2.4 git

`git` driver clones a given repository into a target folder.

URL must be given by `source` parameter.

```
collections = {
    'my_files': {
        'driver': 'git',
        'source': 'https://github.com/useblocks/sphinx_dummy.git',
    }
}
```

`Sphinx-Collections` will clone the given repository into `_collections/my_files/`.

Hint: `Git` binary must be installed on the system, so that the used Python library `gitpython` can use it.

Clean up behavior

During clean up the local repository clone gets deleted.

4.2.5 jinja

Creates one or multiple files based on a given Jinja template in `source`.

`data` must be a dictionary and its keys can be used as identifier in the template.

Example for single data element

template file

Path: `templates/say_hello.rst.template`

```
Hey Hooo!

It's {{name}} from {{city}}!
```

conf.py file

```
collections = [
    'jinja_test': {
        'driver': 'jinja',
        'source': 'templates/say_hello.rst.template',
        'target': 'my_jinja_test.rst',
        'data': {
            'name': 'Max',
            'city': 'Munich'
        },
        'active': True,
    },
]
```

The values inside `{{..}}` get replaced by the related value from the data dictionary.

If `multiple_files` is set to `True`, `data` must be a list and the driver gets executed for each element (dict) in this list.

To get also a new file for each element of the list, you can use Jinja syntax also in `target` and `source`.

Example for multiple data element

template file

Path: `templates/say_hello.rst.template`

```
Hey Hooo!

It's {{name}} from {{city}}!
```

conf.py file

```
collections = [
    'jinja_test': {
        'driver': 'jinja',
        'source': 'templates/say_hello.rst.template',
        'target': 'my_jinja_test_for_{{name|lower}}.rst',
```

(continues on next page)

(continued from previous page)

```
        'data': [
            {
                'name': 'Max',
                'city': 'Munich'
            },
            {
                'name': 'Sandra',
                'city': 'Barcelone'
            },
        ],
        'active': True,
    },
]
```

This example would create two files: `my_jinja_test_for_max.rst` and `my_jinja_test_for_sandra.rst`

4.2.6 report

Creates a collection report in file specified by `target`.

Please be sure to specify this report as one of the latest collections, otherwise other collections have not been executed before this report gets generated.

```
 collections = {
   'my_collection_report: {
      'driver': 'report',
      'target': 'reports/collections.rst'
      }
   }
}
```

Hint: The `executed` option in the report for the current report-collection is always `False`, as this options gets changed **after** the report was successfully generated.

The following template is used to build the report:

```
Collection Report
=================

{%  for collection in collections %}
{{ collection.name }}
{{ "-" * collection.name|length }}
**Active**: {{ collection.active }}

**Executed**: {{ collection.executed }}

**Source**: {{ collection.config['source'] }}

**Target**: {{ collection.config['target'] }}

.. code-block:: text
```

(continues on next page)

(continued from previous page)

```
   {{ collection.config }}


{% endfor %}
```

Example Report

This is the report of the latest run for this documentation.

Collection Report

driver_test

Active: False

Executed: False

Source: ../tests/dummy/

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/driver_test

```
{'driver': 'my_driver', 'source': '../tests/dummy/', 'active': False, 'name': 'driver_
↪test', 'confdir': '/home/docs/checkouts/readthedocs.org/user_builds/sphinx-
↪collections/checkouts/latest/docs', 'target': '/home/docs/checkouts/readthedocs.org/
↪user_builds/sphinx-collections/checkouts/latest/docs/_collections/driver_test',
↪'safe': True}
```

copy_folder_test

Active: False

Executed: False

Source: ../tests/dummy/

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/copy_folder_test

```
{'driver': 'copy_folder', 'source': '../tests/dummy/', 'ignore': ['*.dat'], 'active':␣
↪False, 'name': 'copy_folder_test', 'confdir': '/home/docs/checkouts/readthedocs.org/
↪user_builds/sphinx-collections/checkouts/latest/docs', 'target': '/home/docs/
↪checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_
↪collections/copy_folder_test', 'safe': True}
```

copy_file_test

Active: False

Executed: False

Source: ../tests/dummy/dummy.rst

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/dummy_new.rst

```
{'driver': 'copy_file', 'source': '../tests/dummy/dummy.rst', 'target': '/home/docs/
↪checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_
↪collections/dummy_new.rst', 'active': False, 'name': 'copy_file_test', 'confdir': '/
↪home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/
↪docs', 'safe': True}
```

string_test

Active: False

Executed: False

Source: Take **this**!!!

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/dummy_string.rst

```
{'driver': 'string', 'source': 'Take **this**!!!', 'target': '/home/docs/checkouts/
↪readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/
↪dummy_string.rst', 'active': False, 'name': 'string_test', 'confdir': '/home/docs/
↪checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs',
↪'safe': True}
```

function_test

Active: False

Executed: False

Source: <function my_func at 0x7fe3b7984488>

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/dummy_function.rst

```
{'driver': 'function', 'source': <function my_func at 0x7fe3b7984488>, 'target': '/
↪home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/
↪docs/_collections/dummy_function.rst', 'active': False, 'name': 'function_test',
↪'confdir': '/home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/
↪checkouts/latest/docs', 'safe': True}
```

report

Active: True

Executed: False

Source:

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/doc_collection_report

```
{'driver': 'report', 'target': '/home/docs/checkouts/readthedocs.org/user_builds/
↪sphinx-collections/checkouts/latest/docs/_collections/doc_collection_report.rst',
↪'active': True, 'name': 'report', 'confdir': '/home/docs/checkouts/readthedocs.org/
↪user_builds/sphinx-collections/checkouts/latest/docs', 'safe': True}
```

symlink_test

Active: False

Executed: False

Source: ../tests/dummy/

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/symlink_test

```
{'driver': 'symlink', 'source': '../tests/dummy/', 'active': False, 'name': 'symlink_
↪test', 'confdir': '/home/docs/checkouts/readthedocs.org/user_builds/sphinx-
↪collections/checkouts/latest/docs', 'target': '/home/docs/checkouts/readthedocs.org/
↪user_builds/sphinx-collections/checkouts/latest/docs/_collections/symlink_test',
↪'safe': True}
```

jinja_test

Active: False

Executed: False

Source: examples/jinja_template.rst.temp

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/my_jinja_test_{{nam

```
{'driver': 'jinja', 'source': 'examples/jinja_template.rst.temp', 'target': '/home/
↪docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/
↪_collections/my_jinja_test_{{name}}.rst', 'data': {'name': 'me', 'city': 'munich'},
↪'active': False, 'name': 'jinja_test', 'confdir': '/home/docs/checkouts/readthedocs.
↪org/user_builds/sphinx-collections/checkouts/latest/docs', 'safe': True}
```

jinja_test_multiple

Active: False

Executed: False

Source: examples/jinja_template.rst.temp

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/my_jinja_test_{{nam

```
{'driver': 'jinja', 'source': 'examples/jinja_template.rst.temp', 'target': '/home/
↪docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/
↪_collections/my_jinja_test_{{name|lower}}.rst', 'multiple_files': True, 'data': [{
↪'name': 'Marco', 'city': 'Munich'}, {'name': 'Daniel', 'city': 'Soest'}], 'active':␣
↪False, 'name': 'jinja_test_multiple', 'confdir': '/home/docs/checkouts/readthedocs.
↪org/user_builds/sphinx-collections/checkouts/latest/docs', 'safe': True}
```

git_test

Active: False

Executed: False

Source: https://github.com/useblocks/sphinx_dummy.git

Target: /home/docs/checkouts/readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/git_test

```
{'driver': 'git', 'source': 'https://github.com/useblocks/sphinx_dummy.git', 'active
↪': False, 'name': 'git_test', 'confdir': '/home/docs/checkouts/readthedocs.org/user_
↪builds/sphinx-collections/checkouts/latest/docs', 'target': '/home/docs/checkouts/
↪readthedocs.org/user_builds/sphinx-collections/checkouts/latest/docs/_collections/
↪git_test', 'safe': True}
```

Clean up behavior

During clean up the target folders, which contains the report, gets deleted.

4.2.7 string

Copies a string defined in `source` into a file specified by `target`.

```
collections = {
  'my_files: {
     'driver': 'string',
     'source': 'Awesome, this is nice',
     'target': 'my_data/my_file.txt'
     }
  }
}
```

You can also use more complex strings by assigning them to a variable.

```
my_string = """
Headline
========

Ohh **awesome**!

Multiline!

.. codeblock:: rst

   Works also
   ----------
"""

collections = {
  'my_files: {
     'driver': 'string',
     'source': my_string,
     'target': 'my_data/my_file.txt'
     }
  }
}
```

Clean up behavior

During clean up the created target file gets deleted.

4.2.8 symlink

Creates a symlink (symbolic link) from `target` to `source`.

`target` must be a folder inside your documentation project.

Please note, config option `source` is used as the name where the symlink shows to, as this is our data source for additional sphinx files. And `target` is the path where the symlink starts, because this defines the place where the extra sphinx files shall be stored.

If executed on Windows 10, special privileges may be needed for creating symlinks. Please see os.symlink() details for constraints.

This symlink driver can deal with links to folders and files.

```
 collections = {
   'my_files: {
      'driver': 'symlink',
      'source': '../extra_files/',
      'target': 'my_data/'
      }
   }
}
```

Clean up behavior

During clean up the symlink gets unlinked/removed.

4.2.9 Own drivers

You can specify own drivers directly inside your `conf.py` file.

Using own drivers instead of e.g. a pure function call has several advantages:

- Configuration handling.
- Correct and easy logging.
- Executed during correct Sphinx phases.
- Integrated clean-up.
- Report capabilities.

```
from sphinxcontrib.collections.drivers import Driver
from sphinxcontrib.collections.api import register_driver


class myDriver(Driver):
    def run(self):
        self.info('Run for source {}'.format(self.config['source']))

    def clean(self):
```

(continues on next page)

(continued from previous page)

```
        self.info('Clean')

register_driver('my_driver', myDriver)

collections = {
    'my_driver_test': {
        'driver': 'my_driver',
        'source': '../tests/dummy/',
        'active': True,
    },
```

If you have created an awesome driver, please consider to provide it to `Sphinx-Collections` by creating a PR on our github project . This would help our little Sphinx community a lot. Thanks!

Driver class

class sphinxcontrib.collections.drivers.**Driver**(*collection*, *config=None*)

__init__(*collection*, *config=None*)
Initialize self. See help(type(self)) for accurate signature.

clean()
Cares about cleaning up the working space from actions performed in run().

Gets called normally at the beginning and add the end of collection handling.

Must be implement by the parent driver class.

debug(*message*)
Writes a log message of level DEBUG.

Sets collection and driver information as prefix in front of the message

Parameters **message** – string

Returns None

error(*message*, *e=None*)
Raises exception, if driver is in safe mode. Otherwise just a log message gets printed.

Parameters

- **message** – String
- **e** – Traceback object

Returns None

get_path(*path*)
Returns absolute path. If path is given as relative path, the absolute path is calculated taking documentation confdir as base folder.

Returns path string

get_source_path()
Returns absolute source path. If source was configured as relative path, the absolute path is calculated taking documentation confdir as base folder.

Returns path string

info(*message*)
Writes a log message of level INFO.

Sets collection and driver information as prefix in front of the message

Parameters **message** – string

Returns None

run()
Is the main routine for the driver.

Must be implement by the parent driver class.

4.3 Directives

4.3.1 if_collection

Content is added to rst only, if named collection got executed correctly and is therefore available:

```
.. if-collection:: my_test, my_data

   .. toctree::

      /_collections/my_test/index
      /_collections/my_data/index
```

Takes one single argument, which is a comma-separated list of collection names. If one of these collections was executed correctly the data from the content part is added and parsed.

In all other cases nothing is added to the page.

Behaves like Sphinx own only directive, but content gets removed already during read-in phase and is therefore not parsed by Sphinx, if collection was not executed. This avoids a lot of trouble and warnings.

Hint: If you use `if-collection` to add entries to a toctree or set any other value, which Sphinx stores in its internal cache (pickled environment), please use `sphinx-build` with option `-E` to avoid caching. Otherwise a tag or collection change may not get recognized by Sphinx, if related `rst` file got no updates and is therefore taken from cache.

There is a abbreviation available to save some characters: `..  ifc::` :

```
.. ifc:: my_test

   Awesome!
```

4.4 Configuration

4.4.1 collections

Takes a dictionary, which configures the *Collections*.

```
collections = {
  'my_files': {
     'driver': 'copy',
     'source': '../../extra_files/'
  }
}
```

See *Collections* for details.

Default: `{}`

4.4.2 collections_target

Defines the default storage location for all collections.

If a relative path is set, this path is relative to the documentation folder (the one which contains your `conf.py`).

Can be set individually for each collection by using *target*.

Default: `_collections`

4.4.3 collections_clean

If `True` all configured target locations get wiped out at the beginning.

The related driver of the collection decides, if and clean is needed and how it must be performed.

If you use nested collections, e.g `_collections/collection_A/collection_B` the outer clean routine of a collection (here `collection_A`) deletes also the content of other collections (here `collection_B`).

Can be overwritten for each collection be setting *clean*.

Default: `True`

4.4.4 collections_final_clean

If `True` all collections start their clean-up routine after a Sphinx build is done. Normally It doesn't matter if the build was an success or stopped.

Works similar to *collections_clean*, but at the end of the build instead before.

Can be overwritten for each collection be setting *final_clean*.

This final clean up is normally active to keep your working tree clean and get no unnecessary files into git or any other solution.

Default: `True`

4.5 Changelog

4.5.1 0.0.1 (not released yet)

- Initial version with basic feature set.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

Symbols

C

D

E

G

I

M

R

S