Ansys Sphinx Theme

ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Sep 16, 2024

CONTENTS

The Ansys Sphinx Theme is a custom Ansys-branded theme for use with Sphinx, a documentation generator for creating project documentation from reStructuredText source files.

This theme is specifically tailored for documentation related to Ansys projects helping to ensure consistency in its look and feel. Various useful extensions are included in the theme to make documentation more appealing and user-friendly.

Getting started   Learn how to install the Ansys Sphinx Theme.

Getting started   User guide   Learn how to use the capabilities and features of this theme.

User guide   Examples   Explore examples that show how to integrate third-party extensions with this theme.

Examples

CHAPTER

ONE

GETTING STARTED

How to install   Learn how to download and install the theme.

Installation

1.1 Installation

There are multiple sources for installing the latest stable version of the Ansys Sphinx Theme. These include public PyPI, Ansys PyPI, and GitHub.

Public PyPI

```
python -m pip install ansys-sphinx-theme
```

Ansys PyPI

```
PIP_EXTRA_INDEX_URL="https://${PYANSYS_PYPI_PRIVATE_PAT}@pkgs.dev.azure.com/pyansys/_
↪packaging/pyansys/pypi/simple/"
python -m pip install ansys-sphinx-theme
```

GitHub

```
python -m pip install git+https://github.com/ansys/ansys-sphinx-theme.git@v1.0.9
```

CHAPTER TWO

USER GUIDE

This section outlines the fundamental configurations of the Ansys Sphinx theme and how to integrate third-party extensions with this theme to customize your documentation.

Basic configuration Configure the Ansys Sphinx Theme

Basic configuration *HTML theme options* Set theme options

HTML theme options *The linkcode extension* See how to use the `linkcode` extension with the Ansys Sphinx Theme.

The linkcode extension *Sphinx AutoAPI* See how to use Sphinx AutoAPI with the Ansys Sphinx Theme.

Sphinx AutoAPI

2.1 Basic configuration

To use the Ansys Sphinx Theme, add the following line to your project's Sphinx `conf.py` file:

```
html_theme = "ansys_sphinx_theme"
```

The Ansys Sphinx Theme provides these features:

- *PyAnsys and Ansys logos*
- *Version switcher*
- *PDF cover page*
- *Custom CSS*

2.1.1 PyAnsys and Ansys logos

The Ansys Sphinx Theme includes the PyAnsys and Ansys logos. All the logos are available in the ansys_sphinx_theme/static/ directory.

To use the logo in both dark and light modes, add the following code to the``html_theme_options`` dictionary in your project's Sphinx `conf.py` file:

Ansys logo

```
html_theme_options = {
    "logo": "ansys",
}
```

PyAnsys logo

```
html_theme_options = {
    "logo": "pyansys",
}
```

No logo

```
html_theme_options = {
    "logo": "no_logo",
}
```

Note

By default, if `ansys` logo is displayed, the logo links to the Ansys website. If the PyAnsys logo is displayed, the logo links to the PyAnsys website. If you want to change the link, you can set the `logo_link` option in the `conf.py` file.

For example:

```
html_theme_options = {
    "logo": "ansys",
    "logo_link": "https://www.example.com",
}
```

Note

If you use the `logo` option, make sure to remove the `html_logo` option from the `conf.py` file. `logo` option overrides the `html_logo` option and display the specified logo.

You can also add a custom logo by specifying the path to the logo file as specified in pydata-sphinx-theme.

For example:

```
html_theme_options = {
    "logo": {
        "image_light": "_static/logo-light.png",
        "image_dark": "_static/logo-dark.png",
    }
}
```

2.1.2 `favicon`

The `favicon` setting specifies the icon that appears in the browser tab. To use the Ansys favicon, add the following code to your project's Sphinx `conf.py` file:

```
html_favicon = ansys_favicon
```

2.1.3 Version switcher

The Ansys Sphinx Theme includes a version switcher for switching between different versions of the documentation. To show the version switcher in your documentation, add the following code to your project's Sphinx `conf.py` file:

```
from ansys_sphinx_theme import get_version_match

version = "0.1.0"
switcher_versions = get_version_match(version)
cname = "your_name"
html_theme_options = {
    "switcher": {
        "json_url": f"https://{cname}/versions.json",
        "version_match": switcher_version,
    },
}
```

The switcher requires a `versions.json` file that contains the versions of the documentation and their URLs in the given `json_url`. For more information, see PyAnsys multi-version documentation in the *PyAnsys developer's guide*.

2.1.4 PDF cover page

The Ansys Sphinx Theme includes a PDF cover page that you can customize. To customize the PDF cover page, see *PDF cover page*.

2.1.5 Custom CSS

You can add custom CSS to the Ansys Sphinx Theme by creating a directory named `_static/css` in your documentation and adding the following code to your project's Sphinx `conf.py` file:

```
html_static_path = ["_static"]
html_css_files = ["css/custom.css"]
```

Here is an example of a custom CSS file that changes the background color of the body to black and the text color to white:

```
.body {
    background-color: black;
   color: white;
}
```

2.2 PDF cover page

For generating a PDF of your documentation, Sphinx uses a default cover page. However, you can use the `generate_preamble` function in the `ansys_sphinx_theme.latex` module to create and use a custom cover page:

`ansys_sphinx_theme.latex.`**`generate_preamble`**(*title*, *watermark='watermark'*, *date=None*)

Generate the preamble for the PDF documentation.

Parameters

title
[`str`] Title of the document.

watermark
[`str`, `optional`] Name of the watermark image.

date
[`datetime`, `optional`] Date of document generation. If not provided, today's date is used.

Returns

`str`
A string representing the LaTeX source code for the preamble.

You use this function to generate the value for the `preamble` key in the `latex_elements` variable declared in the `conf.py` file:

```
latex_elements = {
    "preamble": ansys_sphinx_theme.latex.generate_preamble(
        <title_of_coverpage>,
        <watermark_for_titlepage>,
        <date_to_be_printed>,
    )
}
```

To use the logo and watermark provided by Ansys on the cover page, you must import them and then add them to the `latex_additional_files` dictionary:

```python
from ansys_sphinx_theme import (
    ansys_logo_white,
    ansys_logo_white_cropped,
    watermark,
)
```

```
latex_additional_files = [watermark, ansys_logo_white, ansys_logo_white_cropped]
```

For an example of a rendered PDF cover page, see the PDF documentation.

2.3 HTML theme options

In the Sphinx configuration (conf.py) file in the doc directory, you can use the html_theme_options dictionary to customize the Ansys Sphinx theme.

2.3.1 Show breadcrumbs

Showing breadcrumbs at the top of your documentation pages makes navigation easier. Breadcrumbs are shown by setting "show_breadcrumbs": True. To add additional *root* breadcrumbs, "additional_breadcrumbs" is set to a list of tuples in this form: ("Link text", "url").

This html_theme_options dictionary show breadcrumbs, including a root breadcrumb for the documentation landing page for the Ansys repository:

```
html_theme_options = {
    "github_url": "https://github.com/ansys/ansys-sphinx-theme",
    "show_prev_next": False,
    "show_breadcrumbs": True,
    "additional_breadcrumbs": [
        ("PyAnsys", "https://docs.pyansys.com/"),
    ],
}
```

When you are on the landing page for your documentation, the breadcrumb shows the title for this page. However, Sphinx cannot access this title from other documentation pages. Thus, after html_theme_options dictionary, you must set html_short_title to the display text to use for this breadcrumb.

To ensure a consistent user experience, always set the html_short_title (or optionally html_title if html_short_title is not used) to the library name.

For example, in the conf.py file for the Ansys Sphinx Theme, this line is added after the html_theme_options dictionary:

```
html_short_title = html_title = "Ansys Sphinx Theme"
```

If you want the title for your documentation's main index.rst file to show the version, include |version| in the title:

```
html_short_title = html_title = "Ansys Sphinx Theme |version|"
```

2.3.2 Add and hide icons in the navigation bar

The navigation bar shows two icons on the right by default. The first is for switching between light and dark modes. The second is for going to the library's GitHub repository.

- For comprehensive information on adding custom link behavior, see Add custom attributes to icon links in the PyData Theme documentation.
- For comprehensive information on how to use Font Awesome to add icons, see How To Add Icons in the Font Awesome documentation.

The following sections explain how to add icons and hide icons.

Add icons

In the `conf.py` file, the `html_theme_options` dictionary has a child `icon_links` dictionary. To add icons to the navigation bar, add them to the `icon_links` dictionary. For each icon to add, specify its `name`, the associated `url`, the `icon`, and the `type`.

This example adds an icon for sending an email:

```
html_theme_options = {
 "icon_links": [
     dict(name="Mail", url="mailto:me", icon="fas fa-envelope")
 ],
 ...
}
```

Hide icons

To hide icons so that they do not show in the navigation bar, add their names to the `hidden_icons` dictionary:

```
html_theme_options = {
    "hidden_icons": ["GitHub"],
    ...
}
```

If you want to hide all icons, use the `show_icons` Boolean variable:

```
html_theme_options = {
    "show_icons": False,
    ...
}
```

2.3.3 Use MeiliSearch

MeiliSearch is an open source search engine that allows developers to easily integrate search functionality into their applications.

To use MeiliSearch in your documentation, in the `conf.py` file, a child dictionary named `use_meilisearch``is added to the ``html_theme_options` dictionary.

This dictionary contains these keys, in the order given:

1. `host`: Host name of your MeiliSearch instance. If no value is provided, the default public host for PyAnsys is used: `https://backend.search.pyansys.com` on port `7700`. If added security is needed, you can use the `os.getenv()` function to set the instance using an environment variable.
2. `api_key`: API key for your MeiliSearch instance. If no value is provided, the default public API key for PyAnsys is used. If added security is needed, you can use the `os.getenv()` function to set the key using an environment variable.
3. `index_uids`: Dictionary that provides the mapping between the unique identifier (UID) of an index and its corresponding user-friendly name. Each key-value pair in the dictionary represents an index, with the key being the index UID and the value being the index name. The index UID points to an index on the server.

Here is an example of how to configure MeiliSearch for use in the `conf.py` file:

```python
import os

use_meilisearch = {
    "host": os.getenv("MEILISEARCH_HOST_NAME", ""),
    "api_key": os.getenv("MEILISEARCH_API_KEY", ""),
    "index_uids": {
        "index-uid of current project": "index name to display",
        "another-index-uid": "index name to display",
    },
}
```

If your project features multiple documentation versions, it's crucial to adapt the `index_uids` mapping to accommodate different versions. To ensure seamless search integration across versions, use the following format to dynamically generate version-specific index `UIDs`:

```python
from ansys_sphinx_theme import convert_version_to_pymeilisearch

use_meilisearch = {
    "api_key": os.getenv("MEILISEARCH_PUBLIC_API_KEY", ""),
    "index_uids": {
        f"ansys-sphinx-theme-v{convert_version_to_pymeilisearch(__version__)}": "ansys-
↪sphinx-theme",
    },
}
```

Here is an example configuration of how to configure MeiliSearch in the `conf.py` file for the Ansys Sphinx Theme:

```python
import os

html_theme_options = {
    "use_meilisearch": {
        "index_uids": {
            "ansys-sphinx-theme-sphinx-docs": "ansys-sphinx-theme",
            "pyansys-docs-all-public": "PyAnsys",
        },
    },
}
```

With these options set, MeiliSearch is available for performing searches of your documentation.

Note

If you do not set the `use_meilisearch` dictionary, the Ansys Sphinx Theme uses the default search functionality inherited from the PyData Sphinx Theme.

2.3.4 Cheat sheets

If a cheat sheet has been created for your PyAnsys library, with `quarto`, you can add it to the left navigation pane of your documentation.

In the `html_theme_options` dictionary, you add a child dictionary named `cheatsheet` that contain these keys, in the order given:

1. `file`: File name including the extension of the cheat sheet. If the file is inside a directory, include the directory name relative to the root of the documentation. For example, if the cheat sheet is in the `getting_started` directory, the file name is `getting_started/cheat_sheet.qmd`.
2. `title`: Title of the cheat sheet to be displayed in the left navigation pane.
3. `pages`: List of names for the pages to include the cheat sheet on. If no value is provided, the cheat sheet is displayed only on the main `index.html` file.

Here is an example of how to add the `cheatsheet` dictionary to the *html_theme_options*` dictionary:

```
html_theme_options = (
    {
        "cheatsheet": {
            "file": "<file name including the extension of the cheat sheet>",
            "pages": "<list of names for the pages to include the cheat sheet on>",  #␣
↪Optional
        },
    },
)
```

Here is an example of how to show a thumbnail of a PyMAPDL cheat sheet in the left navigation pane of its main `index.rst` file and the `learning.rst` file in its "Getting started" section:

```
html_theme_options = (
    {
        "cheatsheet": {
            "file": "getting_started/cheat_sheet.qmd",
            "pages": ["index", "getting_started/learning"],
        },
    },
)
```

Note

To use this feature, you must have the *quarto <https://quarto.org/>* package installed. To create thumbnails of generated PDF files, the theme is using *pdf2image*. So you should have the `poppler` package installed in your system. For more information, see the pdf2image documentation.

2.4 The linkcode extension

The linkcode extension automatically adds *source* links to the documentation for Python, C, C++, and JavaScript objects. It allows you to link to the source code hosted on GitHub.

To use the linkcode extension, you must add it to the extensions variable in your project's Sphinx conf.py file:

```
extensions = ["ansys_sphinx_theme.extension.linkcode"]
```

2.4.1 Configuration options

The Linkcode extension provides a way to configure its behavior by using certain options within your conf.py file. Depending on your preferred approach, you can utilize the direct configuration options or the html_context dictionary to streamline your settings.

If both sets of configuration options are given, the direct configuration options (that is, link_code_library, link_code_source, link_code_branch) has precedence over the corresponding settings in the html_context dictionary.

Direct configuration options

- link_code_library : The user/repository name where the source code is hosted. For example, ansys/ansys-sphinx-theme.
- link_code_source (str, optional, default: ''): The relative path of the source code file within the repository. For example, src.
- link_code_branch (str, optional, default: 'main'): The GitHub branch. It can be a specific version like main or dev.

If the link_code_source and link_code_branch options are not provided in the configuration, the following default values are used:

- link_code_source: An empty string (''). This links to the root of the repository.
- link_code_branch: main. This is the default branch name used if no branch is specified.

```
# Example of setting direct configuration in example
link_code_library = "username/repo-name"
link_code_source = "src"
link_code_branch = "dev"
```

Using html_context dictionary

You also have the option to centralize your GitHub-related configuration by incorporating it directly into the html_context dictionary within your conf.py file. This approach allows you to minimize redundancy and manage the GitHub-related information more effectively for the extension:

```
# Example of setting GitHub-related configuration in conf.py
html_context = {
    "github_user": "<your-github-org>",
    "github_repo": "<your-github-repo>",
    "github_version": "<your-branch>",
```

(continues on next page)

(continued from previous page)

```
    "source_path": "<path-from-root-to-your-source_file>",
}
```

With this setup, you can fine-tune your configuration according to your preferences and requirements, enhancing the integration of the `linkcode` extension into your documentation.

2.5 Sphinx AutoAPI

To use Sphinx AutoAPI with the Ansys Sphinx Theme, you must add `ansys_sphinx_theme.extension.autoapi` to the `extensions` list in your `conf.py` file and set the `ansys_sphinx_theme_autoapi` theme options in the `html_theme_options` dictionary.

- `project`: The name of the project.
- `output`: The path to the directory where the generated files are placed. By default, this is set to the `api` directory.
- `templates`: The path to the directory containing the custom templates for `sphinx-autoapi`. By default, this is set to the `autoapi_templates` directory in the theme package.
- `directory`: The path to the directory containing the source code with respect to the `conf.py` file. By default, this is set to the `src/ansys` directory.
- `use_implicit_namespaces`: If set to `True`, the autoapi extension use *implicit namespaces*. By default, this is set to `True`.
- `keep_files`: If set to `True`, the autoapi extension keeps the generated files. By default, this is set to `True`.
- `own_page_level`: The level of the page where the autoapi extension places the content of the class. By default, this is set to `class`.
- `type`: The type of the autoapi extension. By default, this is set to `python`.
- `options`: The options to pass to the autoapi extension. By default, this is set to `["members", "undoc-members", "show-inheritance", "show-module-summary", "special-members"]`.
- `class_content`: The content of the class. By default this is set to `class`.
- `ignore`: The list of directories to ignore. By default, this is empty.
- `add_toctree_entry`: If set to `True`, the autoapi extension adds the generated files to the TOC tree. By default, this is set to `False`.
- `package_depth`: The depth of the package. By default, this is set to `3`. This is the `namespace` depth of the package. For example, if the package is `ansys`, the depth is `1`. If the package is `ansys.foo`, the depth is `2`.
- `member_order`: The order to document members. By default, this is set to `bysource`. Other options include `alphabetical`, which orders members by their name (case sensitive), or `groupwise`, which orders members by their type and alphabetically.

All these options can be set in the `conf.py` file of your Sphinx project.

```
html_theme_options = {
    "ansys_sphinx_theme_autoapi": {
        "project": "My Project",
        "output": "api",
        "directory": "src/ansys",
        "use_implicit_namespaces": True,
        "keep_files": True,
```

(continues on next page)

(continued from previous page)

```
        "own_page_level": "class",
        "type": "python",
        "options": [
            "members",
            "undoc-members",
            "show-inheritance",
            "show-module-summary",
            "special-members",
        ],
        "class_content": "class",
        "ignore": [],
        "add_toctree_entry": False,
        "package_depth": 3,
        "member_order": "bysource",
    }
}
```

You need to add `ansys_sphinx_theme.extension.autoapi` to the `extensions` list in your `conf.py` file:

```
extensions = [
    "ansys_sphinx_theme.extension.autoapi",
]
```

The complete configuration for `sphinx-autoapi` in your `conf.py` file should look like this:

```
html_theme_options = {
    "ansys_sphinx_theme_autoapi": {
        "project": "My Project",
        "output": "api",
        "use_implicit_namespaces": True,
        "directory": "src/ansys",
        "keep_files": True,
        "own_page_level": "class",
        "type": "python",
        "options": [
            "members",
            "undoc-members",
            "show-inheritance",
            "show-module-summary",
            "special-members",
        ],
        "class_content": "class",
    }
}

extensions = [
    "ansys_sphinx_theme.extension.autoapi",
]
```

CHAPTER

THREE

EXAMPLES

This section provides examples that show how the Ansys Sphinx Theme renders different Sphinx extensions and documentation components.

3.1 Sphinx extensions

These examples show how the Ansys Sphinx Theme renders different Sphinx extensions.

Sphinx design   Examples of how the Sphinx design is rendered.

Sphinx design   Jupyter notebook - nbsphinx   Examples of how notebooks are rendered.

Jupyter notebook - nbsphinx   Sphinx gallery   Examples of how the Sphinx gallery is rendered.

Sphinx-Gallery   Sphinx autoapi   Examples of how Sphinx AutoAPI documentation is rendered.

Sphinx AutoAPI

3.2 Documentation components

These examples show how the Ansys Sphinx Theme renders different documentation components.

Tables   Examples of tables with JavaScript and RST rendered using the *ansys-sphinx-theme*.

Table   Admonitions   Examples of how admonitions are rendered.

Admonitions

3.2.1 Sphinx design

This example shows how the Ansys Sphinx Theme renders documentation components using the `sphinx-design` extension. For comprehensive information, see its documentation.

Credits and acknowledgments

The examples presented below are sourced from the Sphinx Design project Copyright (c) by Chris Sewell (2023), available at sphinx design examples. This project is licensed under the MIT License.

Badge-basic

```
:bdg:`plain badge`

:bdg-primary:`primary`, :bdg-primary-line:`primary-line`

:bdg-secondary:`secondary`, :bdg-secondary-line:`secondary-line`

:bdg-success:`success`, :bdg-success-line:`success-line`

:bdg-info:`info`, :bdg-info-line:`info-line`

:bdg-warning:`warning`, :bdg-warning-line:`warning-line`

:bdg-danger:`danger`, :bdg-danger-line:`danger-line`

:bdg-light:`light`, :bdg-light-line:`light-line`

:bdg-dark:`dark`, :bdg-dark-line:`dark-line`
```

This directive renders as follows:

plain badge

primary, primary-line

secondary, secondary-line

success, success-line

info, info-line

warning, warning-line

danger, danger-line

light, light-line

dark, dark-line

Badge-link

```
:bdg-link-primary:`https://example.com`

:bdg-link-primary-line:`explicit title <https://example.com>`
```

This directive renders as follows:

https://example.com

explicit title

Button-link

```
.. button-link:: https://example.com

.. button-link:: https://example.com

    Button text

.. button-link:: https://example.com
    :color: primary
    :shadow:

.. button-link:: https://example.com
    :color: primary
    :outline:

.. button-link:: https://example.com
    :color: secondary
    :expand:
```

This directive renders as follows:

https://example.com

Button text

https://example.com

https://example.com

https://example.com

Card-basic

```
.. card:: Card Title

    Card content
```

This directive renders as follows:

Card Title   Card content

Card-carousel

```
.. card-carousel:: 2

    .. card:: card 1

        content

    .. card:: card 2

        Longer
```

(continues on next page)

(continued from previous page)

```
        content

    .. card:: card 3

    .. card:: card 4

    .. card:: card 5

    .. card:: card 6
```

This directive renders as follows:

card 1   content

card 2   Longer

content

card 3   card 4   card 5   card 6

Card-head-foot

```
.. card:: Card Title

    Header
    ^^^
    Card content
    +++
    Footer
```

This directive renders as follows:

Header

Card Title   Card content

Footer

Card-images

```
.. grid:: 2 3 3 4

    .. grid-item::

        .. card:: Title
            :img-background: images/particle_background.jpg
            :class-card: sd-text-black
            :img-alt: my text

            Text

    .. grid-item-card:: Title
```

(continues on next page)

(continued from previous page)

```
        :img-top: images/particle_background.jpg
        :img-alt:

        Header
        ^^^
        Content
        +++
        Footer

    .. grid-item-card:: Title
        :img-bottom: images/particle_background.jpg

        Header
        ^^^
        Content
        +++
        Footer
```

This directive renders as follows:

Title Text

Header

Title Content

Footer

Header

Title Content

Footer

Card-link

```
.. _cards-clickable:

Clickable cards
...............

Using the ``link`` and ``link-type`` options, you can turn an entire card into a␣
↪clickable link. Try hovering over then clicking on the cards below:

.. card:: Clickable Card (external)
    :link: https://example.com

    The entire card can be clicked to navigate to https://example.com.

.. card:: Clickable Card (external)
    :link: https://example.com
    :link-alt: example.com

    The entire card can be clicked to navigate to https://example.com.

.. card:: Clickable Card (internal)
    :link: cards-clickable
    :link-type: ref

    The entire card can be clicked to navigate to the ``cards-clickable`` reference card-
↪link.txt.

.. card:: Clickable Card (internal)
    :link: cards-clickable
    :link-type: ref
    :link-alt: clickable cards

    The entire card can be clicked to navigate to the ``cards-clickable`` reference card-
↪link.txt.
```

This directive renders as follows:

Clickable cards

Using the `link` and `link-type` options, you can turn an entire card into a clickable link. Try hovering over then clicking on the cards below:

Clickable Card (external) The entire card can be clicked to navigate to https://example.com.

https://example.com Clickable Card (external) The entire card can be clicked to navigate to https://example.com.

example.com Clickable Card (internal) The entire card can be clicked to navigate to the `cards-clickable` reference card-link.txt.

Clickable cards Clickable Card (internal) The entire card can be clicked to navigate to the `cards-clickable` reference card-link.txt.

clickable cards

Card-title-link

```
.. _card-title-link.txt:
.. card:: Card Title https://example.com :ref:`link <card-title-link.txt>`

    Card content
```

This directive renders as follows:

Card Title https://example.com *link* Card content

Div-basic

```
.. div:: sd-text-center sd-font-italic sd-text-primary

    Some CSS styled text
```

This directive renders as follows:

Some CSS styled text

Dropdown-basic

```
.. dropdown::

    Dropdown content

.. dropdown:: Dropdown title

    Dropdown content

.. dropdown:: Open dropdown
    :open:

    Dropdown content
```

This directive renders as follows:

Dropdown content

Dropdown title

Dropdown content

Open dropdown

Dropdown content

Dropdown-options

```
.. dropdown:: Title
    :name: dropdown-options.txt
    :color: info
    :icon: alert
    :margin: 1
    :class-container: class-container
    :class-title: class-title
    :class-body: class-body

    Dropdown content

:ref:`dropdown-options.txt`, :ref:`text <dropdown-options.txt>`
```

This directive renders as follows:

Title

Dropdown content

Title, *text*

Grid-basic

```
.. grid:: 1 2 3 4
    :outline:

    .. grid-item::

        A

    .. grid-item::

        B

    .. grid-item::

        C

    .. grid-item::

        D
```

This directive renders as follows:

A

B

C

D

Grid-card-columns

```
.. grid:: 2

    .. grid-item-card::
        :columns: auto

        A

    .. grid-item-card::
        :columns: 12 6 6 6

        B

    .. grid-item-card::
        :columns: 12

        C
```

This directive renders as follows:

A

B

C

Grid-card

```
.. grid:: 2

    .. grid-item-card::  Title 1

        A

    .. grid-item-card::  Title 2

        B
```

This directive renders as follows:

Title 1   A

Title 2   B

Grid-gutter

```
.. grid:: 2
    :gutter: 1

    .. grid-item-card::

        A

    .. grid-item-card::

        B

.. grid:: 2
    :gutter: 3 3 4 5

    .. grid-item-card::

        A

    .. grid-item-card::

        B
```

This directive renders as follows:

A

B

A

B

Grid-nested

```
.. grid:: 1 1 2 2
    :gutter: 1

    .. grid-item::

        .. grid:: 1 1 1 1
            :gutter: 1

            .. grid-item-card:: Item 1.1

                Multi-line

                content

            .. grid-item-card:: Item 1.2

                Content
```

(continues on next page)

(continued from previous page)

```
    .. grid-item::

        .. grid:: 1 1 1 1
            :gutter: 1

            .. grid-item-card:: Item 2.1

                Content

            .. grid-item-card:: Item 2.2

                Content

            .. grid-item-card:: Item 2.3

                Content
```

This directive renders as follows:

Item 1.1 Multi-line

content

Item 1.2 Content

Item 2.1 Content

Item 2.2 Content

Item 2.3 Content

Icon-fontawesome

```
An icon :fas:`spinner;sd-bg-primary sd-bg-text-primary`, some more text.
```

This directive renders as follows:

An icon , some more text.

Icon-material-design

```
- A regular icon: :material-regular:`data_exploration;2em`, some more text
- A coloured regular icon: :material-regular:`settings;3em;sd-text-success`, some more␣
↪text.
- A coloured outline icon: :material-outlined:`settings;3em;sd-text-success`, some more␣
↪text.
- A coloured sharp icon: :material-sharp:`settings;3em;sd-text-success`, some more text.
- A coloured round icon: :material-round:`settings;3em;sd-text-success`, some more text.
- A coloured two-tone icon: :material-twotone:`settings;3em;sd-text-success`, some more␣
↪text.
- A fixed size icon: :material-regular:`data_exploration;24px`, some more text.
```

This directive renders as follows:

- A regular icon: , some more text
- A coloured regular icon: , some more text.
- A coloured outline icon: , some more text.
- A coloured sharp icon: , some more text.
- A coloured round icon: , some more text.
- A coloured two-tone icon: , some more text.
- A fixed size icon: , some more text.

Icon-octicon

```
A coloured icon: :octicon:`report;1em;sd-text-info`, some more text.
```

This directive renders as follows:

A coloured icon: , some more text.

Tab-basic

```
.. tab-set::

    .. tab-item:: Label1

        Content 1

    .. tab-item:: Label2

        Content 2
```

This directive renders as follows:

Label1

Content 1

Label2

Content 2

Tab-code-set

```
.. tab-set-code::

    .. literalinclude:: ./snippet.py
        :language: python

    .. code-block:: javascript

        a = 1;
```

This directive renders as follows:

PYTHON

```python
# Copyright (C) 2021 - 2024 ANSYS, Inc. and/or its affiliates.
# SPDX-License-Identifier: MIT
#
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

"""Sample functions for ansys-sphinx-theme."""


def func(arg1, arg2):
    """Summary line <should be one one line>.

    Extended description of function.  Can span multiple lines and
    provides a general overview of the function.

    .. warning::
       Use the ``.. warning::`` directive within the doc-string for any
       warnings you would like to explicitly state.  For example, if
       this method will be deprecated in the next release.

    Parameters
```

(continues on next page)

(continued from previous page)

```
    ----------
    arg1 : int
        Description of arg1.
    arg2 : str
        Description of arg2.

    Returns
    -------
    bool
        Description of return value.

    Examples
    --------
    >>> func(1, 'foo')
    True

    """
    return True
```

JAVASCRIPT

```
a = 1;
```

Tab-options

```
.. tab-set::
    :class: class-set

    .. tab-item:: **Label**
        :name: tab-options.txt
        :selected:
        :class-container: class-container
        :class-label: class-label
        :class-content: class-content

        Content

:ref:`tab-options.txt`, :ref:`text <tab-options.txt>`
```

This directive renders as follows:

Label

Content

Label, *text*

Tab-sync

```
.. tab-set::
    :sync-group: category

    .. tab-item:: Label1
        :sync: key1

        Content 1

    .. tab-item:: Label2
        :sync: key2

        Content 2

.. tab-set::
    :sync-group: category

    .. tab-item:: Label1
        :sync: key1

        Content 1

    .. tab-item:: Label2
        :sync: key2

        Content 2
```

This directive renders as follows:

Label1

Content 1

Label2

Content 2

Label1

Content 1

Label2

Content 2

3.2.2 Jupyter notebook - nbsphinx

This example shows how the Ansys Sphinx Theme renders a Jupyter notebook using the `nbspinx` extension.

Download this example as a `Jupyter notebook`.

Nbsphinx example

This example renders a Jupyter notebook using the `nbsphinx` extension.

Plot a simple sphere using PyVista.

[1]:
```
import pyvista as pv

pv.set_jupyter_backend('html')

sphere = pv.Sphere()
sphere.plot()
```

```
EmbeddableWidget(value='<iframe srcdoc="<!DOCTYPE html>\n<html>\n  <head>\n    <meta␣
↪http-equiv=&quot;Content-...
```

[2]:
```
plotter = pv.Plotter(notebook=True)
plotter.add_mesh(sphere, color='white', show_edges=True)
plotter.title = '3D Sphere Visualization'
plotter.show()
```

```
EmbeddableWidget(value='<iframe srcdoc="<!DOCTYPE html>\n<html>\n  <head>\n    <meta␣
↪http-equiv=&quot;Content-...
```

Render equations using the IPython `math` module.

[3]:
```
from IPython.display import Math
eq = Math(r'\int\limits_{-\infty}^\infty f(x) \delta(x - x_0) dx = f(x_0)')
eq
```

[3]:

$$\int\limits_{-\infty}^\infty f(x) \delta(x - x_0) dx = f(x_0)$$

[4]:
```python
from IPython.display import Latex
Latex(r'This is a \LaTeX{} equation: $a^2 + b^2 = c^2$')
```

[4]: This is a LaTeX equation: $a^2 + b^2 = c^2$

Render a table in markdown.

This is an example to render the table inside the notebook

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

Render a data frame

[5]:
```python
import pandas as pd

# Create a dictionary of data
data = {
    'A': [True, False, True, False],
    'B': [False, True, False, True],
    'C': [True, True, False, False],
}

# Create DataFrame from the dictionary
df = pd.DataFrame(data)

# Display the DataFrame
df.head()
```

[5]:
```
       A      B      C
0   True  False   True
1  False   True   True
2   True  False  False
3  False   True  False
```

3.2.3 Sphinx-Gallery

This example shows how the Ansys Sphinx Theme renders examples using Sphinx-Gallery.

Sphinx-Gallery

This example shows how to add a new example when using Sphinx-Gallery.

To use Sphinx-Gallery, first install the package with this command:

```
pip install sphinx-gallery
```

Then, add the package to the `extensions` variable in your Sphinx `conf.py` file:

```
extensions = [
    "sphinx_gallery.gen_gallery",
]
```

Plot a simple sphere using PyVista

This code plots a simple sphere using PyVista.

```
import pyvista as pv

pv.set_jupyter_backend("html")

sphere = pv.Sphere()
sphere.plot()
```

Plot a simple sphere using PyVista with a plotter

```
plotter = pv.Plotter(notebook=True)
plotter.add_mesh(sphere, color="white", show_edges=True)
plotter.title = "3D Sphere Visualization"
plotter.show()
```

Render equations using IPython `math`

This example shows how to render equations using the IPython `math` module.

```
from IPython.display import Math, display

# LaTeX formatted equation
equation = r"\int\limits_{-\infty}^\infty f(x) \delta(x - x_0) \, dx = f(x_0)"
# Display the equation
display(Math(equation))
```

```
from IPython.display import Latex

Latex(r"This is a \LaTeX{} equation: $a^2 + b^2 = c^2$")
```

Render a table in markdown

This is an example to render a table inside the markdown with Sphinx-Gallery.

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

Render a table using pandas

```python
import pandas as pd

# Create a dictionary of data
data = {
    "A": [True, False, True, False],
    "B": [False, True, False, True],
    "C": [True, True, False, False],
}

# Create DataFrame from the dictionary
df = pd.DataFrame(data)

# Display the DataFrame
df.head()
```

3.2.4 Sphinx AutoAPI

This example shows how the Ansys Sphinx Theme renders API documentation using the Sphinx AutoAPI extension.

API reference

This section describes ansys_sphinx_theme endpoints, their capabilities, and how to interact with them programmatically.

The `examples` package

Summary

Submodules

sample_func	Sample functions for ansys-sphinx-theme.
samples	Sample classes and functions for ansys-sphinx-theme.
type_hint_example	Module containing an example function using type hinting.

The `sample_func.py` module

Summary

Functions

func	Summary line <should be one one line>.

examples.sample_func.func

`examples.sample_func.`**`func`**(*arg1*, *arg2*)

Summary line <should be one one line>.

Extended description of function. Can span multiple lines and provides a general overview of the function.

Warning

Use the `.. warning::` directive within the doc-string for any warnings you would like to explicitly state. For example, if this method will be deprecated in the next release.

Parameters

arg1
[`int`] Description of arg1.

arg2
[`str`] Description of arg2.

Returns

bool
Description of return value.

Examples

```
>>> func(1, 'foo')
True
```

Description

Sample functions for ansys-sphinx-theme.

The `samples.py` module

Summary

Classes

ExampleClass	The summary line for a class docstring should fit on one line.
Complex	Custom implementation of a complex number.

`ExampleClass`

class examples.samples.**ExampleClass**(*param1*, *param2*, *param3=0*)

The summary line for a class docstring should fit on one line.

Attributes should be documented inline with the attribute's declaration.

Properties created with the `@property` decorator should be documented in the property's getter method.

Parameters

param1
[`str`] Description of *param1*.

param2
[`list of str`] Description of *param2*. Multiple lines are supported.

param3
[`int, optional`] Description of *param3*.

Examples

An example of how to initialize this class should be given.

```
>>> from ansys_sphinx_theme import samples
>>> example = samples.ExampleClass('mystr', ['apple', 'orange'], 3)
```

Overview

Methods

example_method	Class methods are similar to regular functions.

Properties

readonly_property	Properties should be documented in their getter method.
readwrite_property	Set or return the readwrite property.

Attributes

attr1
attr2
attr3
attr4
attr5

Special methods

__special__	By default special members with docstrings are not included.
__special_without_docstring__	

Import detail

```
from examples.samples import ExampleClass
```

Property detail

property ExampleClass.**readonly_property: str**

Properties should be documented in their getter method.

Examples

```
>>> example.readonly_property
"readonly_property"
```

property ExampleClass.**readwrite_property**

Set or return the readwrite property.

Properties with both a getter and setter should only be documented in their getter method.

If the setter method contains notable behavior, it should be mentioned here.

Examples

```
>>> example.readwrite_property
"readwrite_property"
```

```
>>> example.readwrite_property = 'hello world'
>>> example.readwrite_property
'hello world'
```

Attribute detail

`ExampleClass.`**`attr1`**

`ExampleClass.`**`attr2`**

`ExampleClass.`**`attr3`**

`ExampleClass.`**`attr4 = ['attr4']`**

`ExampleClass.`**`attr5 = None`**

Method detail

`ExampleClass.`**`example_method`**(*param1*, *param2*)

Class methods are similar to regular functions.

Parameters

param1
[`str`] The first parameter.

param2
[`str`] The second parameter.

Returns

bool
`True` if successful, `False` otherwise.

Notes

Do not include the `self` parameter in the `Parameters` section.

Examples

```
>>> example.example_method('foo', 'bar')
True
```

`ExampleClass.__special__()`

By default special members with docstrings are not included.

Special members are any methods or attributes that start with and end with a double underscore. Any special member with a docstring will be included in the output, if `napoleon_include_special_with_doc` is set to True.

This behavior can be enabled by changing the following setting in Sphinx's conf.py:

```
napoleon_include_special_with_doc = True
```

`ExampleClass.__special_without_docstring__()`

Complex

class `examples.samples.Complex`(*real*, *imag=0.0*)

Bases: `object`

Custom implementation of a complex number.

Parameters

real
[`float`] Real component of the complex number.

imag
[`float`, `optional`] Imaginary component of the complex number.

Examples

```
>>> my_num = Complex(real=1, imag=-1.0)
>>> my_num
(1.0 + 1.0j)
```

Overview

Properties

real	Real component of this complex number.
imag	Real component of this complex number.
abs	Return the absolute value of this number.

Special methods

`__add__`	Add two complex numbers.
`__sub__`	Subtract two complex numbers.
`__mul__`	Multiply two complex numbers.
`__truediv__`	Divide two complex numbers.
`__repr__`	Return repr(self).

Import detail

```
from examples.samples import Complex
```

Property detail

property `Complex.`**real**

Real component of this complex number.

Examples

```
>>> my_num = Complex(real=1, imag=-1.0)
>>> my_num.real
1.0
```

property `Complex.`**imag**

Real component of this complex number.

Examples

```
>>> my_num = Complex(real=1, imag=-1.0)
>>> my_num.imag
-1.0
```

Set the imaginary component

```
>>> my_num.imag = 2.0
>>> my_num.imag
2.0
```

property `Complex.`**abs**

Return the absolute value of this number.

Examples

```
>>> my_num = Complex(real=1, imag=1.0)
>>> my_num.abs
```

Method detail

`Complex.`**`__add__`**`(`*other*`)`

Add two complex numbers.

`Complex.`**`__sub__`**`(`*other*`)`

Subtract two complex numbers.

`Complex.`**`__mul__`**`(`*other*`)`

Multiply two complex numbers.

`Complex.`**`__truediv__`**`(`*other*`)`

Divide two complex numbers.

`Complex.`**`__repr__`**`()`

Return repr(self).

Description

Sample classes and functions for ansys-sphinx-theme.

The `type_hint_example.py` module

Summary

Functions

type_hint_func	Summary containing the function description.

examples.type_hint_example.type_hint_func

`examples.type_hint_example.`**`type_hint_func`**`(`*param1: int = 1, param2: str = 'test', param3: int | float = 1*`) → bool`

Summary containing the function description.

Extended description of the function. Can span multiple lines and provides a general overview of the function.

Parameters

param1
Description of an integer parameter.

param2
Description of a string parameter.

param3
Parameter that can be either int or float using Union (typing).

Returns

bool
Description of the returned value.

Examples

```
>>> func(1, 'foo', 1)
True
```

Description

Module containing an example function using type hinting.

Description

A sub-package containing various examples for checking their rendering.

3.2.5 Table

The table directive with ansys sphinx theme allows for rendering of tables. There are different types of tables, such as the data table, longtable-centered, and table-centered, each serving different purposes.

Normal table

Table 1: Truth table for "not"

A	B	C
False	True False	False True
False	True False	False True
False	True True False	False True

Data table

This is an example of a data table that can be rendered using the table directive. It consists of three columns representing the variables A, B, and A and B respectively. Each row represents a different combination of True and False for variables A and B. The *datatable* class can be used to style the data table.

```
.. table::
   :class: datatable

   =====  =====  =======
   A      B      A and B
   =====  =====  =======
   False  False  False
   True   False  False
   False  True   False
   True   True   True
   =====  =====  =======
```

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

Longtable-centered

The longtable-centered class can be used to create a table that spans multiple pages and is centered horizontally. This is useful for tables that have a large number of rows or columns. Here is an example of a longtable-centered:

```
.. table::
   :class: longtable-centered

   +---------------------------+-------------------+
   |                           | MAPDL Command     |
   +===========================+===================+
   | **GUI commands**          | * ``*ASK``        |
   |                           |                   |
   | **GUI commands**          | * ``*ASK``        |
   |                           |                   |
   | **GUI commands**          | * ``*ASK``        |
   +---------------------------+-------------------+
```

	MAPDL Command
GUI commands **GUI commands** **GUI commands**	• `*ASK` • `*ASK` • `*ASK`

Table-centered

The table-centered class can be used to create a table that is horizontally centered. This is useful for tables that have only a few columns. Here is an example of a table-centered:

```
.. table::
    :class: table-centered

    +----------------------------+--------------------+
    |                            | MAPDL Command      |
    +============================+====================+
    | **GUI commands**           | * ``*ASK``         |
    +----------------------------+--------------------+
    | **GUI commands**           | * ``*ASK``         |
    +----------------------------+--------------------+
    | **GUI commands**           | * ``*ASK``         |
    +----------------------------+--------------------+
```

	MAPDL Command
GUI commands	• `*ASK`
GUI commands	• `*ASK`
GUI commands	• `*ASK`

3.2.6 Admonitions

Admonitions are specially formatted blocks that highlight important information in the documentation. This page shows how the Ansys Sphinx Theme renders admonitions.

The examples shown below are derived from the PyData Sphinx Theme project, copyrighted by pandas, 2018. Original examples can be found in Examples in the PyData Sphinx Theme documentation. These examples are licensed under the BSD 3-Clause License.

The examples included below in this page were originally copyrighted and licensed as follows:

- Copyright (c) 2021, Pradyun Gedam. Licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (SPDX-License-Identifier: CC-BY-SA-4.0), as specified by the PyData Sphinx Theme in their examples.

Admonitions

Sphinx provides several different types of admonitions.

`topic`

This is a topic.

This is what admonitions are a special case of, according to the docutils documentation.

`admonition`

The one with the custom titles

It's got a certain charm to it.

`attention`

Attention

Climate change is real.

`caution`

Caution

Cliff ahead: Don't drive off it.

`danger`

Danger

Mad scientist at work!

`error`

Error

Does not compute.

`hint`

Hint

Insulators insulate, until they are subject to ______ voltage.

`important`

Important

Tech is not neutral, nor is it apolitical.

`note`

Note

This is a note.

`seealso`

See also

Other relevant information.

`tip`

Tip

25% if the service is good.

`todo`

Todo

This needs the `sphinx.ext.todo` extension.

`warning`

Warning

Reader discretion is strongly advised.

`versionadded`

Added in version v0.1.1: Here’s a version added message.

`versionchanged`

Changed in version v0.1.1: Here’s a version changed message.

`deprecated`

Deprecated since version v0.1.1: Here’s a deprecation message.

CHAPTER

FOUR

RELEASE NOTES

This document contains the release notes for the project.

4.1 1.0.9 - 2024-09-16

4.1.1 Added

- feat: add member_order to autoapi #495

4.1.2 Fixed

- fix: `autoapi` relative directory path wrt `tox` env #494

4.2 1.0.8 - 2024-09-03

4.2.1 Fixed

- fix: Align jupyter cell output #489
- fix: the download in sphinx gallery #490

4.3 1.0.7 - 2024-08-23

4.3.1 Fixed

- fix: autoapi extension #472
- fix: admonitions styles for `topic` admonition #477

4.4 1.0.6 - 2024-08-23

4.4.1 Fixed

- fix: download icon with sphinx-gallery and nbsphinx #471
- feat: add different width for different media for main content #473
- fix: the scrollbar on sidebar #474

4.4.2 Documentation

- chore: update CHANGELOG for v1.0.5 #470

4.5 1.0.5 - 2024-08-16

4.5.1 Fixed

- feat: add default logo links for Ansys and PyAnsys logos #469

4.5.2 Dependencies

- build(deps): bump nbsphinx from 0.9.4 to 0.9.5 #465

4.6 1.0.4 - 2024-08-13

4.6.1 Fixed

- fix: tables and cell output #460

4.6.2 Dependencies

- ci: bump ansys/actions from 6 to 7 #457
- build(deps): bump numpydoc from 1.7.0 to 1.8.0 #459

4.7 1.0.3 - 2024-08-09

4.7.1 Fixed

- fix: minor style changes #452
- fix: downgrade the autoapi and keep `autoapi` toctree to `True` by default #453
- fix: *pygment_styles* with dark and light theme and dark theme table #454

4.8 1.0.2 - 2024-08-08

4.8.1 Changed

- maint: update ansys actions #449

4.8.2 Fixed

- fix: sphinx design image background #450

4.9 1.0.1 - 2024-08-08

4.9.1 Fixed

- fix: stable docs indexing package name #446

4.10 1.0.0 - 2024-08-08

4.10.1 Added

- fix: update the github icon #401
- feat: add default logo and update logo option with theme #425
- feat: add quarto cheat sheet extension with cheat sheet option #428

4.10.2 Changed

- chore: update CHANGELOG for v0.16.2 #381
- chore: update CHANGELOG for v0.16.3 #389
- chore: update CHANGELOG for v0.16.4 #390
- chore: update CHANGELOG for v0.16.5 #394
- chore: update CHANGELOG for v0.16.6 #402

4.10.3 Fixed

- fix: Align cheat sheet center #382
- fix: reformat the style files #406
- fix: reformat the table styles #408
- fix: reformat navigation bar and background #409
- fix: *primary* ,`secondary` sidebars and links #411
- fix: sphinx design reformat #412

- fix: update the breadcrumbs #419
- fix: admonitions style #424
- fix: sidebar borders and overflow #427
- fix: search bar styles #429
- fix: updated the logo options #431
- fix: add dropdown styles for the header navigation bar #437
- fix: dark theme variables #438
- fix: sphinx card *box shadow* on focus #439
- fix: focus links with keyboard #440
- fix: search bar style for dark theme, icons links #442

4.10.4 Dependencies

- build(deps-dev): update pydata-sphinx-theme requirement from <0.15,>=0.14 to >=0.15 #336
- chore: version 0.17.dev0 #386
- chore(deps): bump requests from 2.32.2 to 2.32.3 #391
- docs: reformat the documentation #396
- chore(deps): bump sphinx-autoapi from 3.1.1 to 3.1.2 #405
- build(deps): bump pyvista[jupyter] from 0.43.10 to 0.44.0 #413
- build(deps): bump jupytext from 1.16.2 to 1.16.3 #415
- build(deps): bump sphinx from 7.3.7 to 7.4.4 #416
- build(deps): bump sphinx from 7.4.4 to 7.4.5 #417
- build(deps): bump sphinx from 7.4.5 to 7.4.6 #418
- build(deps): bump sphinx-autoapi from 3.1.2 to 3.2.0 #420
- build(deps): bump sphinx-gallery from 0.16.0 to 0.17.0 #421
- build(deps): bump pyvista[jupyter] from 0.44.0 to 0.44.1 #422
- build(deps): bump sphinx from 7.4.6 to 7.4.7 #423
- build(deps): bump sphinx-autoapi from 3.2.0 to 3.2.1 #426
- build(deps): bump sphinx-notfound-page from 1.0.2 to 1.0.3 #432
- build(deps): bump jupytext from 1.16.3 to 1.16.4 #433
- build(deps): bump sphinx-notfound-page from 1.0.3 to 1.0.4 #434
- build(deps): bump sphinx-design from 0.6.0 to 0.6.1 #435
- build(deps): bump sphinx from 7.4.7 to 8.0.2 #436
- build(deps): bump sphinx-gallery from 0.17.0 to 0.17.1 #441

4.10.5 Miscellaneous

- refactor: remove function duplicate #407
- docs: Update *mail id* in README.rst #414

4.11 0.16.6 - 2024-06-18

4.11.1 Fixed

- fix: wrong env var name for PACKAGE_NAME #395

4.12 0.16.5 - 2024-05-31

4.12.1 Fixed

- fix: sphinx design card font size #393

4.13 0.16.4 - 2024-05-29

4.13.1 Added

- feat: adapt package to general PyAnsys repository layout #387

4.13.2 Dependencies

- chore(deps): bump sphinx-design from 0.5.0 to 0.6.0 #383
- chore(deps): bump sphinx-notfound-page from 1.0.1 to 1.0.2 #384
- chore(deps): bump sphinx-autoapi from 3.1.0 to 3.1.1 #385

4.14 0.16.3 - 2024-05-29

4.14.1 Fixed

- fix: update the sphinx design style to disable display of name #388

4.15 0.16.2 - 2024-05-22

4.15.1 Changed

- chore: update CHANGELOG for v0.16.1 #379

4.15.2 Miscellaneous

- docs: update changelog_template.jinja #380

4.16 0.16.1 - 2024-05-22

4.16.1 Added

- feat: add nerd fonts for `autoapi` templates icon #362
- feat: add the changelog action #370
- feat: add autoapi extension #372

4.16.2 Fixed

- fix: add changelog action in ci-cd #378

4.16.3 Dependencies

- chore(deps): bump requests from 2.31.0 to 2.32.1 #374
- maint: update the sphinx-autoapi version #375
- chore(deps): bump sphinx-notfound-page from 1.0.0 to 1.0.1 #376
- chore(deps): bump requests from 2.32.1 to 2.32.2 #377

PYTHON MODULE INDEX

INDEX