ORMIR Standard Operating Procedure for Producing Package Documentation using Sphinx

By Donnie Cameron, 2024

1. Introduction:

The purpose of this Standard Operating Procedure (SOP) is to outline the steps and guidelines for creating Google style package documentation using Sphinx. Sphinx is a popular documentation generation tool widely used in the software development community. By following this SOP, you can produce high-quality, well-structured documentation that adheres to Google's documentation style guidelines. Note that we use the Google style because it is easy to read both in the code and in the generated docs.

See this page for a quick tour of how to use Sphinx.

2. Prerequisites:

Before proceeding with the documentation production process, ensure that the following prerequisites are met:

- Sphinx is installed on your system.

- The package code is well-documented with appropriate docstrings following the Google style guidelines. See here for details: https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html.

- A code repository is available, containing the package source code.

3. Setup and Configuration:

3.1. Install Sphinx:

If Sphinx is not already installed, install it using the package manager of your choice (e.g., pip). Run the following command:

`pip install sphinx`

3.2. Create a Documentation Directory:

Create a new directory specifically for your documentation files. For example, create a directory named "docs" at the root of your project. Specify separate source and build directories for the docs when prompted.

3.3. Initialize Sphinx:

Open a terminal or command prompt and navigate to the "docs" directory. Then run the following command to initialise Sphinx:

`sphinx-quickstart`

4. Configuration:

4.1. Edit 'conf.py':

A Python file called 'conf.py' should have appeared in the "docs/source" directory. Modify the following settings as necessary:

- Set the 'extensions' variable to include 'sphinx.ext.autodoc' and 'sphinx.ext.napoleon' extensions.

- Specify the location of the package source code using the 'sys.path' variable.

- Set the 'html_theme' variable to 'sphinx_rtd_theme', for example, for a modern, clean look.

4.2. Add 'index.rst':

Create an 'index.rst' file in the "docs" directory. This file will serve as the main entry point for your documentation. This should contain your main table of contents (`toctree::`) and preamble text.

5. Documenting Your Package:

5.1. Create Documentation Files:

For each module or class you want to document, create a corresponding '.rst' file in the "docs" directory. Use a meaningful name that reflects the module or class being documented.

See the following resources for tips on how to write documents in the restructured text format:

- http://docutils.sourceforge.net/rst.html
- http://docutils.sourceforge.net/docs/user/rst/quickref.html
- http://docutils.sourceforge.net/docs/user/rst/cheatsheet.txt

5.2. Write Documentation:

Open each '.rst' file and start documenting the module or class. Follow the Google-style guidelines for documenting your code, including proper headings, descriptions, parameters, return values, and examples. Refer to the Sphinx documentation for detailed formatting instructions.

Rather than adding .rst files on a module-by-module or function-by-function basis, you can use Sphinx's autodoc functionality to pull in documentation from docstrings in a semi-automatic manner.

Add `.. automodule::` and `.. autoclass::` commands to your .rst files to pull in documentation from modules and classes respectively. Add `:members:` on the next line to recurse through all member functions, etc.

6. Building the Documentation:

6.1. Build HTML Output:

In the terminal or command prompt, navigate to the "docs" directory. Run the following command to build the HTML output:

`make html`

6.2. Verify Build Success:

After the build process completes, check for any errors or warnings in the output. Resolve any issues before proceeding.

7. Review and Publish:

7.1. Review Documentation:

Open the generated HTML documentation in a web browser and review the content. Ensure that it follows the Google style guidelines and meets the desired quality standards. Make any necessary revisions or improvements to the docstrings and .rst files accordingly.

7.2. Publish the Documentation:

Upload the generated HTML documentation to a suitable hosting platform or - ideally - integrate it into your project's website using GitHub pages. Ensure that the documentation remains easily accessible and up-to-date. The 'build' folder can be added to gitignore.

8. Maintenance and Updates:

Keep the documentation up-to-date with any changes or additions to your package. Update the docstrings in the code

9. Conclusion:

Following this SOP will enable you to produce Google style package documentation using Sphinx. By documenting your package effectively, you enhance its usability and promote collaboration within the ORMIR community. Regularly update and maintain the documentation to ensure its accuracy and relevance over time.

Note: This SOP provides a general framework for producing Google style package documentation using Sphinx. We recommend you refer to the Sphinx (https://pythonhosted.org/an_example_pypi_project/sphinx.html) and Google (https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html) documentation for additional guidance and best practices.