docs-python2readthedocs Documentation

Release 0.1.0

Matthew John Hayes

Dec 02, 2017

Contents

1 Introduction **3**

2 Install Sphinx **5**
2.1 Pre-Work . . . 5
2.1.1 Ensure packages are up-to-date . . . 5
2.1.2 Install Python pip . . . 5
2.2 Sphinx Installation . . . 5

3 Configure Sphinx **7**
3.1 sphinx-quickstart . . . 7

4 Autodoc Your Code **11**
4.1 Tell autodoc how to Find your Code . . . 11
4.1.1 Submodules . . . 11
4.2 One-Off Creation of RST Files . . . 11
4.3 autodoc directives . . . 12
4.4 Documenting Your Code . . . 12

5 Create Content **13**
5.1 Updating the Index . . . 13

6 reStructuredText **15**
6.1 reStructuredText Conventions . . . 15
6.1.1 Line length . . . 15
6.2 Headings . . . 15
6.2.1 H1 . . . 15
6.2.2 H2 . . . 15
6.2.3 H3 . . . 16
6.2.4 H4 . . . 16
6.2.5 H5 . . . 16
6.2.6 H6 . . . 16
6.3 Text Formatting . . . 16
6.3.1 Italics . . . 16
6.3.2 bold . . . 16
6.3.3 literal . . . 17
6.4 Lists . . . 17
6.4.1 Numbered Lists . . . 17

6.4.2 Bullet Points . . . 17
6.5 Code Blocks . . . 17
6.5.1 Command Line . . . 18
6.5.2 Python . . . 18
6.6 Hyperlinks . . . 18
6.6.1 Simple link . . . 18
6.6.2 Link with name . . . 18
6.6.3 Link to local page . . . 18
6.7 Images . . . 18

7 Local Build **19**

8 UI Tweaks **21**
8.1 Themes . . . 21
8.2 Sidebar . . . 22

9 Read the Docs Integration **23**
9.1 Integrate with ReadtheDocs . . . 23
9.1.1 Sign up with Read the Docs . . . 23
9.1.2 Set up Service on GitHub . . . 23
9.1.3 Import Project in Read the Docs . . . 24
9.1.4 Check Read the Docs Versions . . . 24
9.2 Autodoc Fix for External Module Dependencies . . . 25
9.2.1 Example of Import Problem . . . 25
9.2.2 Fixing Missing Imports with virtualenv . . . 25
9.2.3 Fixing Missing Imports with Mock . . . 26

10 Module Documentation **27**
10.1 example_module module . . . 27
10.2 example_module2 module . . . 27

11 Troubleshooting **29**
11.1 Static Page Problems . . . 29
11.1.1 Why isn't my page showing up in the contents menu? . . . 29
11.1.2 Why isn't my page loading / display correctly? . . . 29
11.2 Autodoc Problems . . . 30
11.2.1 Module files missing or incomplete . . . 30

12 Acknowledgements **31**

13 Indices and tables **33**

Python Module Index **35**

A beginners guide to integrating a Python project with Read the Docs to create great hassle-free documentation. Read More

Contents:

CHAPTER 1

Introduction

This guide is for anyone who has a Python project and wants to improve their documentation by integrating it with Read the Docs.

There are a number of advantages to having your project documentation hosted on Read the Docs:

- Your documentation is specific to your code version. Add a new feature in your develop branch, update the documentation page, commit, update develop on GitHub and within a short amount of time the documentation for the develop branch has been updated on ReadtheDocs, but critically, master branch on ReadtheDocs still shows the documentation specific to the master branch.
- You don't have to worry about hosting a website for the documentation, including all the hassles of making it searchable etc.
- You can configure automatic conversion of Docstrings from your Python code into nice looking searchable documentation pages in Read the Docs.

This guide is built entirely from a Python project on GitHub, using the techniques outlined here. The project on GitHub (https://github.com/mattjhayes/docs-python2readthedocs) can be used as an example of the configuration. It uses a webhook for auto-rebuild of Read the Docs pages on project commits.

A couple of simple Python programs to demonstrate the autodoc functionality are also included in the project.

CHAPTER 2

Install Sphinx

Sphinx is a tool for generation of HTML (and other formats) documentation from reStructuredText

This guide is for installing Sphinx on Ubuntu.

2.1 Pre-Work

2.1.1 Ensure packages are up-to-date

```
sudo apt-get update
sudo apt-get upgrade
```

2.1.2 Install Python pip

```
sudo apt-get install python-pip
```

2.2 Sphinx Installation

Install Sphinx:

```
sudo pip install Sphinx
```

CHAPTER 3

Configure Sphinx

This guide is for configuring Sphinx on Ubuntu.

In the root directory of your project (replace <PROJECT_NAME> where needed), create a folder for the documentation (if it doesn't already exist) called docs:

```
cd
cd <PROJECT_NAME>
mkdir docs
cd docs
```

3.1 sphinx-quickstart

The sphinx-quickstart script does a one-time set-up for the project. If you haven't already configured Sphinx for the project then run it with:

```
sphinx-quickstart
```

Accept the default for root path:

```
Enter the root path for documentation.
> Root path for the documentation [.]:
```

Override the default to have separate source and build directories:

```
You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]: y
```

Accept the default for name prefix:

```
> Name prefix for templates and static dir [_]:
```

Enter the name of your project, and your name:

```
The project name will occur in several places in the built documentation.
> Project name: <PROJECT_NAME>
> Author name(s): <AUTHOR_NAME>
```

Enter version and release numbers for the project:

```
Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version: <X.Y>
> Project release [0.2]: <X.Y.Z>
```

Choose language (English is default):

```
> Project language [en]:
```

You need to make a decision about the file suffix for your restructuredText. ReadtheDocs recommend using .txt extension (see: http://documentation-style-guide-sphinx.readthedocs.io/), however I personally prefer to use the .rst extension so that it is clear what format the files are in. Your choice:

```
The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:
```

Accept the default for epub:

```
Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/n) [n]:
```

Choose to enable autodoc if you have Python code to auto-document:

```
Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/n) [n]: y
```

Accept defaults, apart from Windows (unless you need it):

```
> doctest: automatically test code snippets in doctest blocks (y/n) [n]:
> intersphinx: link between Sphinx documentation of different projects (y/n) [n]:
> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]:
> coverage: checks for documentation coverage (y/n) [n]:
> imgmath: include math, rendered as PNG or SVG images (y/n) [n]:
> mathjax: include math, rendered in the browser by MathJax (y/n) [n]:
> ifconfig: conditional inclusion of content based on config values (y/n) [n]:
> viewcode: include links to the source code of documented Python objects (y/n) [n]:
> githubpages: create .nojekyll file to publish the document on GitHub pages (y/n)␣
↪[n]:

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. `make html' instead of invoking sphinx-build
directly.
> Create Makefile? (y/n) [y]:
> Create Windows command file? (y/n) [y]: n
```

Output:

```
Creating file ./source/conf.py.
Creating file ./source/index.rst.
Creating file ./Makefile.

Finished: An initial directory structure has been created.

You should now populate your master file ./source/index.rst and create
other documentation source files. Use the Makefile to build the docs,
like so:
make builder
where "builder" is one of the supported builders, e.g. html, latex or
linkcheck.
```

A directory structure like this will have been created:

```
+-- docs
¦   +-- build
¦   +-- Makefile
¦   +-- source
¦       +-- conf.py
¦       +-- index.rst
¦       +-- _static
¦       +-- _templates
```

The initial configuration of Sphinx is now complete, keep reading as there are more tasks that still need to be done.

CHAPTER 4

Autodoc Your Code

The Sphinx autodoc extension (see http://www.sphinx-doc.org/en/stable/ext/autodoc.html) converts docstrings from your Python code into the final documentation format at Sphinx build-time.

This is very useful, but may not work out of the box. Here are some steps to set it up properly:

4.1 Tell autodoc how to Find your Code

Autodoc probably can't find your code without a little help. Edit the docs/source/conf.py file. Uncomment:

```
import os
import sys
```

Uncomment and edit this line (adjust path as appropriate):

```
sys.path.insert(0, os.path.abspath('../../<PROJECT_NAME>'))
```

4.1.1 Submodules

If you have submodules, then you may need to use this path instead:

```
sys.path.insert(0, os.path.abspath('../..'))
```

4.2 One-Off Creation of RST Files

There is a script that you can run to create a directive file per Python module. You should only run this command once to set up the *.rst files.

In the docs directory, run this command to create rst files that document your python modules (Note that the -f option tells it to overwrite existing files):

```
sphinx-apidoc -f -o source/ ../<PROJECT_NAME>/
```

You should see rst files created in the docs/source/ folder

4.3 autodoc directives

The reStructuredText files for your Python modules in docs/source do not contain the docstrings. Instead they just contain directives on how to build the corresponding page.

They contain reStructuredText with directives to build the documentation from a particular Python module in your project. Example:

```
example_module module
=====================

.. automodule:: example_module
    :members:
    :undoc-members:
    :show-inheritance:
```

Example from this project, showing source RST and Python with resulting HTML:

reStructuredText: example_module.rst

Python: example_module.py

Auto-generated HTML: example_module.html

Here are some additional directives that you may wish to add include:

- Include private members, i.e. ones that start with an underscore

```
:private-members:
```

- Include special members, i.e. ones that start and end with two underscores, such as __init__

```
:special-members:
```

Example using these extra directives:

reStructuredText: example_module2.rst

Python: example_module2.py

Auto-generated HTML: example_module2.html

4.4 Documenting Your Code

While it is possible to use reStructuredText in the docstrings of your Python code, the author prefers to stay with plain text. Plain text docstrings still produce great HTML pages with autodoc. Ultimately, it is your choice.

CHAPTER 5

Create Content

You should consider creating project documentation in addition to the auto-generated module documentation. While, it's good surfacing your docstrings as nicely formatted pages, you should still have some general pages that introduce your project and add extra context such as diagrams.

5.1 Updating the Index

The file docs/source/index.rst is the landing page for your projects documentation.

Initially it will look something like this:

```
Welcome to <PROJECT_NAME>'s documentation!
==========================================================

Contents:

.. toctree::
   :maxdepth: 2

Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

The rst files for autodoc are in the docs/source directory so it is a good idea for reasons of tidiness and to avoid name collisions to create a subdirectory for your content.

In this example, there is a subdirectory called *userguide*

Add the names of your additional RST files, without file extension, one line below the ':maxdepth: 2'. Prefix with the subdirectory if using, example:

```
userguide/introduction
```

Be sure to preserve the 3-space indent.

See: Example

CHAPTER 6

reStructuredText

There are a lot of guides on how reStructuredText works, and this is not a substitute for them. It is just a brief sample of common formatting options that work with Read the Docs for those in a hurry.

6.1 reStructuredText Conventions

Here are some basic pointers on how to create documentation pages in reStructuredText.

6.1.1 Line length

The length of a line in reStructuredText shouldn't be more than 79 characters

6.2 Headings

Headings are:

6.2.1 H1

A row of #'s above and below the line of text. There should only one H1 in the document. Example:

```
###############
Heading Level 1
###############
```

6.2.2 H2

A row of *'s above and below the line of text. Example:

```
***************
Heading Level 2
***************
```

6.2.3 H3

A row of =’s below the line of text. Example:

```
Heading Level 3
===============
```

6.2.4 H4

A row of -‘s below the line of text. Example:

```
Heading Level 4
---------------
```

6.2.5 H5

A row of ^’s below the line of text. Example:

```
Heading Level 5
^^^^^^^^^^^^^^^
```

6.2.6 H6

A row of “‘s below the line of text. Example:

```
Heading Level 6
"""""""""""""""
```

6.3 Text Formatting

6.3.1 Italics

Surround word(s) with single *asterisks*:

```
*italics*
```

6.3.2 bold

Surround word(s) with double **asterisks**:

```
**bold**
```

6.3.3 literal

Surroud word(s) with double backticks:

```
``double back-quotes``
```

6.4 Lists

Lists must always be preceded by a blank line.

6.4.1 Numbered Lists

Numbered lists are numbers or letters followed by ".", right bracket ")" or surrounded by brackets "( )"

```
This is a numbered list:

1) Item 1
2) Item 2
```

Displays as:

This is a numbered list:

1. Item 1
2. Item 2

6.4.2 Bullet Points

Bullet point lines start with "-", "+" or "*"

```
This is a bullet point list:

* Item 1
* Item 2
```

Displays as:

This is a bullet point list:

- Item 1
- Item 2

6.5 Code Blocks

Use the code-block directive to display code as it appears, including syntax highlighting if desired.

6.5.1 Command Line

Use this directive for text such as command line input and output (note 2 space indent for the code):

```
.. code-block:: text

  code here...
```

6.5.2 Python

Use this directive for Python (note 2 space indent for the code):

```
.. code-block:: python

  code here...
```

6.6 Hyperlinks

6.6.1 Simple link

(note the backticks, angle brackets and trailing underscore)

```
`<http://www.python.org/>`_
```

6.6.2 Link with name

```
`Python <http://www.python.org/>`_
```

6.6.3 Link to local page

```
`Local Page <local_page.html>`_
```

6.7 Images

```
.. image:: images/build1.png
```

CHAPTER 7

Local Build

You can build your documentation locally if you desire by running this command in the docs folder:

```
make html
```

CHAPTER 8

UI Tweaks

Here are some minor changes that you may want to consider to change the User Interface (UI) look and feel:

8.1 Themes

Themes are used by Sphinx to control how the documentation looks when exported to the final formats.

I prefer the Read the Docs theme over the Alabaster theme, which Sphinx installation has configured, so I update it in docs/source/conf.py.

Original line:

```
html_theme = 'alabaster'
```

Change it to:

```
html_theme = 'default'
```

Other Sphinx built-in themes include:

- classic
- sphinxdoc
- scrolls
- agogo
- traditional
- nature
- haiku

8.2 Sidebar

The local site sidebar is a bit limited, however works fine in Read the Docs. If you want a better sidebar for the local build then try this update. Edit the docs/source/conf.py file. Find this stanza:

```
# Custom sidebar templates, maps document names to template names.
#
# html_sidebars = {}
```

Replace the last line of this stanza so it reads:

```
# Custom sidebar templates, maps document names to template names.
#
html_sidebars = { '**': ['globaltoc.html', 'relations.html', 'sourcelink.html',
↪'searchbox.html'], }
```

CHAPTER 9

Read the Docs Integration

9.1 Integrate with ReadtheDocs

9.1.1 Sign up with Read the Docs

Sign up for a Read the Docs account at:

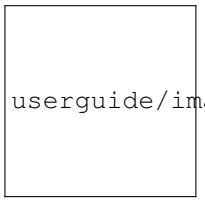
https://readthedocs.org/

9.1.2 Set up Service on GitHub

Go into the admin page for the project on GitHub.

Go to the "Settings" page for your project

- Click "Integrations & services" on the left
- In the "Services" section, click "Add service"



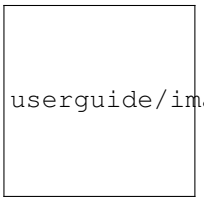
- In the list of available services, click "ReadTheDocs"
- Check "Active"
- Click "Add service"

9.1.3 Import Project in Read the Docs

Log into Read the Docs and click 'Import a Project'.

If the project is not in the list, choose to import it manually:

In GitHub, copy the HTTPS clone URL to clipboard:

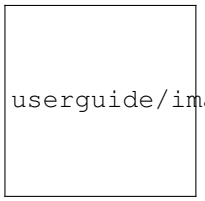


Back in Read the Docs, paste the URL into the 'Repository URL field' and fill in the project name:

9.1.4 Check Read the Docs Versions

Check Read the Docs versions are enabled appropriately for the repository.

Enable where required:



9.2 Autodoc Fix for External Module Dependencies

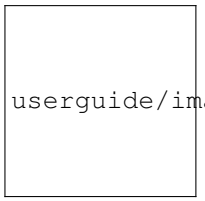
Read the Docs runs Sphinx autodoc against your code in its environment. So, while autodoc may run fine in your own environment, it may fail in ReadtheDocs, due to imported modules not being present.

9.2.1 Example of Import Problem

In Read the Docs, we can see example_module, but not example_module2

We check the build and it passed. What is the problem?

Clicking in Read the Docs admin interface on the 4th line of the build, we see:



and further down this output:

Right. It's failing because colouredlogs module isn't installed in Read the Docs.

There are a couple of ways to fix this if it is a problem. The first one is preferable:

9.2.2 Fixing Missing Imports with virtualenv

In this fix, we tell ReadtheDocs to install module dependencies via pip in a virtual environment, and then run Sphinx autodoc.

Enable virtualenv in Read the Docs

Log into Read the Docs and go into Settings -> Profile -> <PROJECT_NAME>

Go into Admin -> Advanced Settings and tick the 'Install your project inside a virtualenv using setup.py install' box

Fill in the 'Requirements file:' box with requirements.txt

Click 'Submit'

Create a requirements.txt file

Create requirements.txt file (example) in root of project. Here is an example requirements.txt file to install the coloredlogs library:

```
# Install coloredlogs:
coloredlogs
```

Replace coloredlogs with the name(s) of the programs to install with pip.

9.2.3 Fixing Missing Imports with Mock

If the virtualenv solution isn't fully working from you then consider using mock. Code can be added to docs/source/conf.py to mock troublesome imports so that Read the Docs Sphinx doesn't error trying to load them.

Sub-modules must be listed after their parent module and there must be full listing from the top level module. Example that mocks ryu.base.app_manager:

```
import mock

MOCK_MODULES = [
    'ryu',
    'ryu.base',
    'ryu.base.app_manager']

for mod_name in MOCK_MODULES:
    sys.modules[mod_name] = mock.Mock()
```

CHAPTER 10

Module Documentation

Python module documentation autogenerated by Sphinx autodoc

10.1 example_module module

This is an example Python module for the Python to Read the Docs documentation repository.

It is used to show how Sphinx autodoc can be used to auto-generate Python documentation from doc strings like this...

Written by Matthew John Hayes

class example_module.**ExampleModule**
Bases: object

This is the main class of example_module. It doesn't do anything useful other than show how classes are documented by autodoc

increment(*value*)
Increment the value of self.class_variable by value passed to this method

run()
Run the ExampleModule instance

10.2 example_module2 module

This is a second example Python module for the Python to Read the Docs documentation repository.

It is used to show how Read the Docs can be configured to install dependant modules so that Sphinx autodoc can run

Written by Matthew John Hayes

class example_module2.**ExampleModule2**
Bases: object

This is the main class of example_module2

__dict__ = dict_proxy({'__module__': 'example_module2', '_private_method': <function _private_method>, 'run': <f

__init__()
Initialise the ExampleModule2 class

__module__ = 'example_module2'

__weakref__
list of weak references to the object (if defined)

_private_method()
Example private method that won't be documented by autodoc unless you add :private-members: to the automodule directive

increment(*value*)
Increment the value of self.class_variable by value passed to this method

run()
Run the ExampleModule2 instance

CHAPTER 11

Troubleshooting

11.1 Static Page Problems

11.1.1 Why isn't my page showing up in the contents menu?

Check that you page is correctly listed in the index.rst file (check indent!) example .

Check that you're looking at the right branch in Read the Docs

11.1.2 Why isn't my page loading / display correctly?

Check the source reStructuredText file for issues with rstcheck.

Install rstcheck (if you don't already have it) to check syntax of rst code:

```
sudo pip install rstcheck
```

Run it against a particular file:

```
rstcheck <file>
```

Or run it against all reStructuredText files in a directory:

```
rstcheck *.rst
```

The reStructuredText is good if no results are returned.

11.2 Autodoc Problems

11.2.1 Module files missing or incomplete

Check Read the Docs to see if there has been an import problem as per example-of-import-problem

If your code has submodules (i.e. code is in more than one level of directory) then you may need to alter your path statement.

CHAPTER 12

Acknowledgements

This guide was drawn from many sources, including:

http://www.sphinx-doc.org/en/stable/tutorial.html

http://blog.rtwilson.com/how-to-make-your-sphinx-documentation-compile-with-readthedocs-when-youre-using-numpy-and-scipy/

https://codeandchaos.wordpress.com/2012/07/30/sphinx-autodoc-tutorial-for-dummies/

http://gisellezeno.com/tutorials/sphinx-for-python-documentation.html

Any mistakes are mine, not theirs.

If you have any corrections or updates, please comment on my blog post Automating Python Documentation with Read the Docs

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

example_module, 27
example_module2, 27

Index

Symbols

__dict__ (example_module2.ExampleModule2 attribute), 27
__init__() (example_module2.ExampleModule2 method), 28
__module__ (example_module2.ExampleModule2 attribute), 28
__weakref__ (example_module2.ExampleModule2 attribute), 28
_private_method() (example_module2.ExampleModule2 method), 28

E

example_module (module), 27
example_module2 (module), 27
ExampleModule (class in example_module), 27
ExampleModule2 (class in example_module2), 27

I

increment() (example_module.ExampleModule method), 27
increment() (example_module2.ExampleModule2 method), 28

R

run() (example_module.ExampleModule method), 27
run() (example_module2.ExampleModule2 method), 28