PyATL Sphinx Introduction Documentation

Release 2.0

Doug Hellmann

November 13, 2015

Contents

Sphinx is a documentation preparation tool that converts reStructuredText input files into several different output formats like HTML and PDF.

This guide was created with Sphinx.

Contents:

CHAPTER 1

Features

1.1 Input is reStructuredText

- Simple text markup
- Uses semantic and stylistic tags
- Extensible with Python
  - Role Quick-reference
  - Directive Quick-reference

1.1.1 Inline Markup

emphasis; **strong emphasis**; *interpreted text*; *interpreted text with role*; `inline literal text`; standalone hyperlink, http://docutils.sourceforge.net; named reference, **reStructuredText_**; **'anonymous reference'__**; footnote reference, **[1]_**; citation reference, [CIT2002]; **|substitution|**; inline internal target.

```
*emphasis*; **strong emphasis**; `interpreted text`; `interpreted text
with role`:emphasis:; ``inline literal text``; standalone hyperlink,
http://docutils.sourceforge.net; named reference, reStructuredText_;
`anonymous reference`__; footnote reference, [1]_; citation reference,
[CIT2002]_; |substitution|; _`inline internal target`.
```

1.1.2 Body Elements

Grid table:

Paragraphs are flush-left, separated by blank lines.
Block quotes are indented.
	`Indented`
	or:
	`> Quoted`

`>>> print 'Doctest block'` `Doctest block`	
Line blocks preserve line breaks & indents. [new in 0.3.6]	

Useful for addresses, verse, and adornment-free lists; long lines can be wrapped with continuation lines.

```
+------------------------------+-----------------------------------+
| Paragraphs are flush-left,   | Literal block, preceded by "::":: |
| separated by blank lines.    |                                   |
|                              |     Indented                      |
|     Block quotes are indented. |                                 |
+------------------------------+ or::                              |
| >>> print 'Doctest block'    |                                   |
| Doctest block                | > Quoted                          |
+------------------------------+-----------------------------------+
| | Line blocks preserve line breaks & indents. [new in 0.3.6]     |
| |     Useful for addresses, verse, and adornment-free lists; long |
|       lines can be wrapped with continuation lines.              |
+------------------------------------------------------------------+
```

Simple tables:

List Type	Examples (syntax in the text source)
Bullet list	• items begin with “-”, “+”, or “*”
Enumerated list	1. items use any variation of “1.”, “A)”, and “(i)” 2. also auto-enumerated
Definition list	**Term is flush-left** [optional classifier] Definition is indented, no blank line between
Field list	**field name** field body
Option list	**-o** at least 2 spaces between option & description

```
===============  ============================================================
List Type        Examples (syntax in the `text source <cheatsheet.txt>`_)
===============  ============================================================
Bullet list      * items begin with "-", "+", or "*"
Enumerated list  1. items use any variation of "1.", "A)", and "(i)"
                 #. also auto-enumerated
Definition list  Term is flush-left : optional classifier
                     Definition is indented, no blank line between
Field list       :field name: field body
Option list      -o  at least 2 spaces between option & description
===============  ============================================================
```

1.2 Document Structure

1.2.1 Headings within Files

Levels set with overline and underline markup.

```
================
 Document Title
================

Heading 1
=========

Heading 2
---------

Heading 1
=========
```

1.2.2 Table of Contents Join Files

`toctree` directive

```
==========
 Features
==========

.. toctree::
    :maxdepth: 1

    reStructuredText
    structure
    footnotes
    auto_index
    code
    output_formats
```

1.3 Footnotes

- Manually numbered [1]. (`[1]_`)
- Automatically numbered [2]. (`[#]_`)
- Labeled [3] (`[#label]_`)
- Symbols [4] and [5] (`[*]`)
- Citations *[PyMOTW]* (`[PyMOTW]_`)
- Hyperlink-References

1.4 Automatically Generated Index

- Insert index markers
- Hyperlinks or cross-references generated in the build

[1] This is a manually numbered footnote.
[2] This footnote was numbered automatically.
[3] This footnote is labeled.
[4] This footnote just has a symbol.
[5] This footnote is given a different symbol automatically.

- See upper right corner for link

```
.. index::
   single: reStructuredText

.. index::
   pair: source; code
```

1.5 Special Handling for Source Code

1.5.1 Syntax Highlighting

```
#!/usr/bin/env python

"""Example module.
"""

import os

class MyClass(object):
    """This is a simple class.

    This module illustrates three features of Sphinx:

    1. Pygments integration.
    2. Auto-doc features.
    3. Use of rst in docstrings.
    """

    def __init__(self, arg1):
        """Initialize MyClass instance.

        arg1
          Provide a value for the argument.
        """
        self.arg1 = arg1

    def another_method(self):
        """Returns something.
        """
        return self.arg1 * 2

    def method_with_arguments(self, a, b):
        """This method takes arguments.

        :param a: The first argument.
        :type a: int
        :param b: The second argument.
        :type b: str
        """
        return (self.arg1 * a) + b

def main():
    o = MyClass('foo ')
    print o.another_method()
```

1.5.2 Incorporate docstrings

MyClass

class example.**MyClass**(*arg1*)

This is a simple class.

This module illustrates three features of Sphinx:

1. Pygments integration.
2. Auto-doc features.
3. Use of rst in docstrings.

Methods

```
    def another_method(self):
        """Returns something.
        """
        return self.arg1 * 2
```

MyClass.**another_method**()

Returns something.

```
    def method_with_arguments(self, a, b):
        """This method takes arguments.

        :param a: The first argument.
        :type a: int
        :param b: The second argument.
        :type b: str
        """
        return (self.arg1 * a) + b
```

MyClass.**method_with_arguments**(*a*, *b*)

This method takes arguments.

Parameters

- **a** (*int*) – The first argument.
- **b** (*str*) – The second argument.

main()

example.**main**()

1.5.3 Module Index

- Modules automatically indexed

1.6 Output Formats

- HTML
  - Themes
  - Templates (Jinja2)
  - Built-in search
  - Navigation links
- PDF, via LaTeX

CHAPTER 2

Using Sphinx

2.1 Installation

Use pip or easy_install to install sphinx inside of a virtualenv:

```
$ mkvirtualenv pyatlsphinx
$ pip install Sphinx
```

The dependencies installed for you include:

- docutils
- Jinja2
- Pygments

2.2 Getting Started

Starting a brand new project is as simple as running the interactive quickstart script:

```
$ sphinx-quickstart
Welcome to the Sphinx 1.3.1 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

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

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

Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:

The project name will occur in several places in the built documentation.
> Project name: PyATL Sphinx Introduction
> Author name(s): Doug Hellmann
```

```
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: 2.0
> Project release [2.0]:

If the documents are to be written in a language other than English,
you can select a language here by its language code. Sphinx will then
translate text that it generates into that language.

For a list of supported codes, see
http://sphinx-doc.org/config.html#confval-language.
> Project language [en]:

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]:

One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:

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

Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/n) [n]: y
> 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]:
> pngmath: include math, rendered as PNG 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]:

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

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.
```

2.3 Building Docs

- `sphinx-build`
- only modified files are updated [1]

2.3.1 HTML Output

- All dependencies installed with Sphinx
- `make html`

2.3.2 PDF Output

- Requires LaTeX distro such as http://texlive.org/
- `make latex && (cd build/latex; make all-pdf)`
- Straight-to-PDF writer in development

[1] The definition of "modified" depends on the contents of the file. Include directives seem to always cause a rebuild of the node.

CHAPTER 3

Additional Resources

Sphinx The Sphinx home page, including the user manual and links to projects that use Sphinx for their documentation.

sphinx-dev Google group for developers and users of Sphinx.

docutils The docutils site includes links to rst reference guides and other tools for working with rst.

Writing Technical Documentation with Sphinx, Paver, and Cog A post from my blog covering the tool chain I've built up to produce PyMOTW.

Using Sphinx and Doctests to provide Robust Documentation Chris Perkins' presentation from PyCon 2009

The source for this presentation This presentation is available on my website in HTML form. The source is hosted on Github (https://github.com/dhellmann/pyatl-regex-performance).

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

[PyMOTW] This is a citation.

e

A

C

E

M

R

S