Easy and beautiful documentation with Sphinx

Making documentation effective and writable

Alfredo Deza

November 29, 2011

Learn how to create maintainable, style-driven documents that can be automatically distributed in different formats, like HTML, using the Sphinx tool.

Introduction

Sphinx is a tool allowing developers to write documentation in plain text for easy output generation in formats meeting varying needs. This becomes helpful when using a Version Control System to track changes. Plain text documentation is also useful for collaborators across different systems. Plain text is one of the most portable formats currently available.

Although Sphinx is written in Python and was originally created for the Python language documentation, it is not necessarily language-centric and in some cases, not even programmer-specific. There are many uses for Sphinx, such as writing entire books!

Highlighting

Sphinx has code highlighting for Python by default, but it also allows definition of other programming languages like C and Ruby.

Think of Sphinx as a *documentation framework*: it abstracts the tedious parts and offers automatic functionality to solve common problems like title indexing and special code highlighting (if showing code examples) with proper syntax highlighting.

Requirements

You should feel comfortable using a Linux® or UNIX® terminal (also referred to as console or terminal emulator) since the command-line interface is the primary way to interact with Sphinx.

Python needs to be installed. It comes pre-installed and ready to use in all major Linux distributions and some UNIX-based operating systems (like Mac OSX). Sphinx should work with Python versions 2.4, 2.5 and 2.6. To make sure you have Python and a valid version, run the command in Listing 1.

Listing 1. Checking the Python version

```
$ python --version
Python 2.6.1
```

Syntax

Sphinx uses reStructuredText markup syntax (with some additions) to provide document control. If you have ever written plain text files, you probably already know quite a bit about the syntax required to be proficient in Sphinx.

The markup allows definition and structure of text for proper output. Before beginning, see Listing 2 for a small example of the markup syntax.

Listing 2. Example of Sphinx markup syntax

```
This is a Title
===============
That has a paragraph about a main subject and is set when the '='
is at least the same length of the title itself.

Subject Subtitle
----------------
Subtitles are set with '-' and are required to have the same length
of the subtitle itself, just like titles.

Lists can be unnumbered like:

 * Item Foo
 * Item Bar

Or automatically numbered:

 #. Item 1
 #. Item 2

Inline Markup
-------------
Words can have *emphasis in italics* or be **bold** and you can define
code samples with back quotes, like when you talk about a command: ``sudo``
gives you super user powers!
```

As you can see, that syntax looks very readable in plain text. When the time comes to create a specific format (like HTML), the title will look like a major heading and it will have bigger fonts than the subtitle (as it should), and the numbered lists will be properly numbered. Already you have something quite powerful. Adding more items or changing the order in the numbered list doesn't affect the numbering, and titles can change importance by replacing the underline used.

Installation and configuration

Installation occurs in the command line and is straightforward, as shown in Listing 3.

Listing 3. Installing Sphinx

```
$ easy_install sphinx
Searching for sphinx
Reading http://pypi.python.org/simple/sphinx/
Reading http://sphinx.pocoo.org/
Best match: Sphinx 1.0.5
Downloading http://pypi.python.org/packages/[...]
Processing Sphinx-1.0.5-py2.5.egg
[...]
Finished processing dependencies for sphinx
```

Listing 3 was shortened for brevity, but it provides an example of what to expect when installing Sphinx.

The framework uses a directory structure to have some separation between the source (the plain text files) and the build (which refers to the output generated). For example, if Sphinx is directed to generate a PDF from a documentation source, the file would be placed in the build directory. This behavior can be changed, but for consistency we will stick to the default format.

Let's quick start a new documentation project in Listing 4 that will prompt you with a few questions. Accept all the default values by pressing **Enter**.

Listing 4. Executing sphinx-quickstart

```
$ sphinx-quickstart
Welcome to the Sphinx 1.0.5 quickstart utility.

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

I chose "My Project" as the project name that will be referenced in several places. Feel free to choose a different name.

After running the `sphinx-quickstart` command, there should be files in the working directory resembling those in Listing 5.

Listing 5. Listing of the working directory

```
.
├── Makefile
├── _build
├── _static
├── conf.py
└── index.rst
```

Let's take a closer look at each file.

- **Makefile**: Developers who have compiled code should be familiar with this file. If not, think of it as a file containing instructions to build documentation output when using the `make` command.
- **_build**: This is the directory where generated files will be placed after a specific output is triggered.

- **_static**: Any files that are not part of the source code (like images) are placed here, and later linked together in the build directory.
- **conf.py**: This is a Python file holding configuration values for Sphinx, including the ones selected when `sphinx-quickstart` was executed in the terminal.
- **index.rst**: The root of the documentation project. This will connect other files if the documentation is split into other files.

Getting started

At this point, we have Sphinx properly installed, seen what the default structure looks like, and recognize some basic syntax. Be cautious of jumping right into writing documentation. A lack of knowledge about layout and outputs can be confusing and could significantly slow your entire process.

Take a look inside `index.rst`. There is a significant amount of information and some additional complex syntax. To make things easier and avoid distractions, we will incorporate a new file by listing it in the main section.

Right after the main title in the `index.rst` file, there is a content listing with a `toctree` declaration. The `toctree` is the central element to gather all documents into the documentation. If other files are present, but not listed under this directive, those files would not get generated with the documentation at build time.

We want to add a new file to the documentation and it is going to be named `example.rst`. It needs to be listed in the `toctree`, but be careful. There is a spacing that needs to be followed for the listing to work and it does not need the file extension (`.rst` in this case). Listing 6 shows how that list should look. Three blank spaces need to separate the filename from the left margin, with one blank line after the `maxdepth` option.

Listing 6. toctree example in index.rst

```
Contents:

.. toctree::
   :maxdepth: 2

   example
```

Do not worry about the other options at this point. For now, be aware that there is an index file where other separate files are listed that can hold valid documentation, and that this listing follows a certain order and spacing to work.

Remember the example syntax in Listing 2? Copy and paste that example into the `example.rst` file and save it. We are now ready to generate output.

Run the `make` command and specify HTML as output. This output can be used directly as a website as it has everything generated, including JavaScript and CSS files. See Listing 7.

Listing 7. Output of `make html` command

```
$ make html
sphinx-build -b html -d _build/doctrees   . _build/html
Making output directory...
Running Sphinx v1.0.5
loading pickled environment... not yet created
building [html]: targets for 2 source files that are out of date
updating environment: 2 added, 0 changed, 0 removed
reading sources... [100%] index
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index
writing additional files... genindex search
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.

Build finished. The HTML pages are in _build/html.
```

If you are interested in other options that the `make` command offers, see Listing 8 to pass the help flag to it and see a full description.

Listing 8. Listing make options

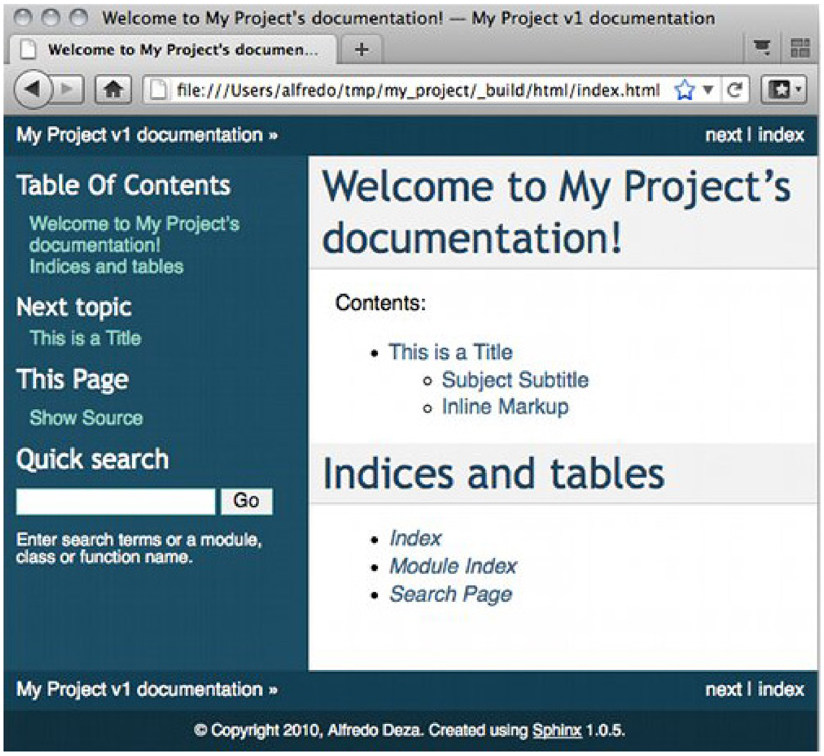
```
$ make -h
Usage: make [options] [target] ...
Options:
[...]
```

Going static

With our first pass at generating HTML from the two files, we have a fully functional (static) website.

Inside the `_build` directory, you should now have two new directories: `doctrees` and `HTML`. We are interested in the HTML directory holding all the files needed for the documentation site. Opening the index.html file with a browser should reveal something like Figure 1.

Figure 1. Main page in static HTML


Welcome to My Project's documentation! — My Project v1 documentation

file:///Users/alfredo/tmp/my_project/_build/html/index.html

My Project v1 documentation » next | index

Table Of Contents
Welcome to My Project's documentation!
Indices and tables

Next topic
This is a Title

This Page
Show Source

Quick search
Go
Enter search terms or a module, class or function name.

Welcome to My Project's documentation!

Contents:

- This is a Title
  - Subject Subtitle
  - Inline Markup

Indices and tables

- *Index*
- *Module Index*
- *Search Page*

My Project v1 documentation » next | index

© Copyright 2010, Alfredo Deza. Created using Sphinx 1.0.5.

With so little information, Sphinx was able to create a lot. We have a basic layout with some information about the project's documentation, a search section, table of contents, copyright notices with name and date, and pagination.

The search part is interesting because Sphinx has indexed all the files and with some JavaScript magic has created a static site that is searchable.

Remember we added `example` as a separate file to the documentation in the `toctree` in Listing 6? You can see that the main title shows as a main bullet in the content index and the subtitle as second level bullets. Sphinx has taken care of all proper structuring.

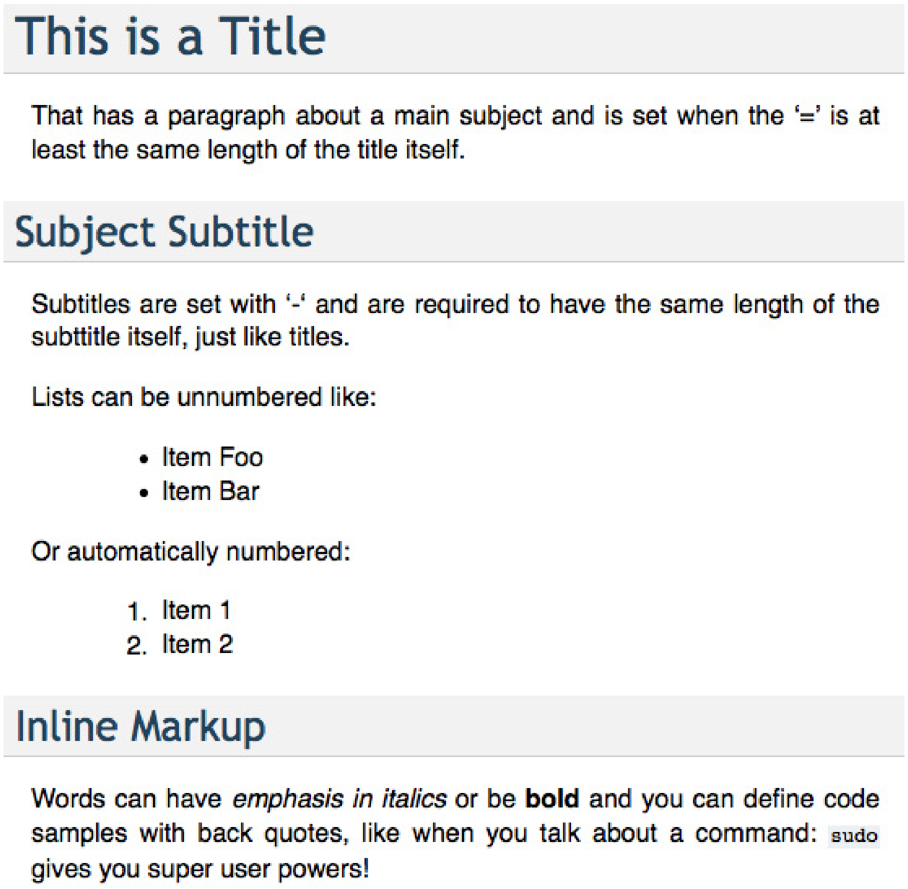
If at first you don't succeed...

After additional modification, simply run the `make` command again to regenerate the files.

All links will point to the right places in the documentation, and titles and subtitles have anchors that allow direct links. For example, the `Subject Subtitle` section holds an anchor that looks like `../example.html#subject-subtitle` in the browser. As mentioned before, the tool removes worrying about these tiny, repetitive requirements.

Figure 2 shows how `example.rst` looks as an HTML file in the static site.

Figure 2. Example page as HTML



This is a Title

That has a paragraph about a main subject and is set when the '=' is at least the same length of the title itself.

Subject Subtitle

Subtitles are set with '-' and are required to have the same length of the subttitle itself, just like titles.

Lists can be unnumbered like:

- Item Foo
- Item Bar

Or automatically numbered:

1. Item 1
2. Item 2

Inline Markup

Words can have *emphasis in italics* or be **bold** and you can define code samples with back quotes, like when you talk about a command: `sudo` gives you super user powers!

Going graphic

Short, concise paragraphs, images, and graphs all add interest and readability to a project's documentation. Sphinx helps keep a reader's attention with these important elements with the possibility of adding static files.

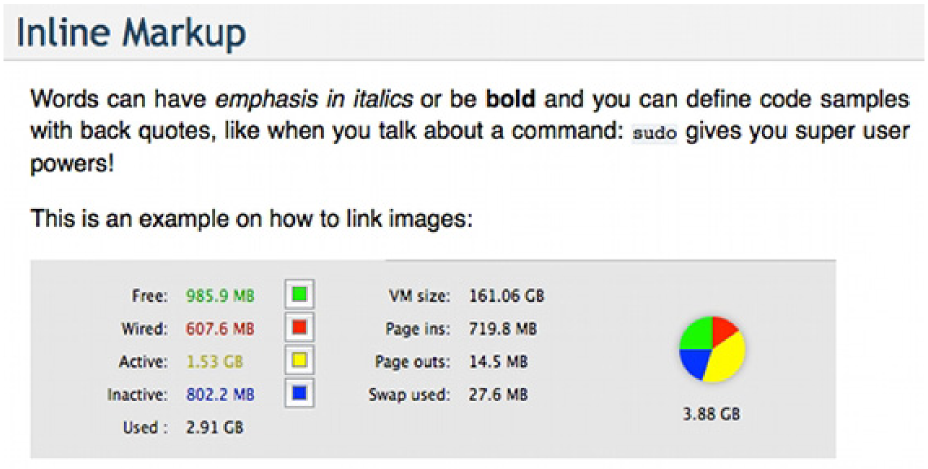
The proper syntax to add static files is easy to remember. As long as you are placing static files in the `_static` directory that Sphinx created when the documentation layout was created, you should be able to reference it without trouble. In Listing 9, see how that reference should look in the reStructuredText file. In this case, I am adding it at the bottom of `example.rst`.

Listing 9. Static listing in example.rst

```
.. image:: _static/system_activity.jpg
```

After the documentation is regenerated, the image should be placed correctly in the same spot where we indicated the path to the small JPEG graph about system activity. It should look similar Figure 3.

Figure 3. System activity image



Inline Markup

Words can have *emphasis in italics* or be **bold** and you can define code samples with back quotes, like when you talk about a command: `sudo` gives you super user powers!

This is an example on how to link images:

Free:	985.9 MB	🟩	VM size:	161.06 GB
Wired:	607.6 MB	🟥	Page ins:	719.8 MB
Active:	1.53 GB	🟨	Page outs:	14.5 MB
Inactive:	802.2 MB	🟦	Swap used:	27.6 MB
Used :	2.91 GB			

Conclusion

This article has covered the basics of getting started with Sphinx, but there is a lot more to explore. Sphinx has the ability to export documentation in different formats, but they require extra libraries and software to be installed. Some of the formats that can be generated are: PDF, epub, man (UNIX Manual Pages), and LaTeX.

For complex graphs, there is a plugin to add Graphviz graphs to your documentation project. I once had to create one for a small office network map, and it was rather nice that I could manage to get everything in the same documentation without having to use a different tool. Like the Graphviz plugin, there is a wealth of plugins (also called extensions) available for Sphinx. Some are included, like interSphinx which allows you to link different Sphinx projects.

If the look and feel of the generated output was not to your liking, Sphinx includes many themes that can be applied to completely change how the HTML files render the documentation. Some important open source projects, such as Celery and Lettuce, heavily modify how the HTML looks by changing the CSS and extending the templates. See the Related topics section for links to those projects and to the documentation explaining how to extend and modify the default CSS and layout.

Sphinx changed the way I thought about writing documentation. I went from uninspired to excited when I was able to easily document almost all of my personal open source projects and a few internal ones. Use Sphinx to easily retrieve forgotten information in your own documentation.

Downloadable resources

Description	Name	Size
Sample Sphinx project for this article	example_sphinx_project.zip	142KB

Related topics

- Sphinx project documentation
- Sphinx extensions: Find a complete list and some external references for third party extensions.
- Theming and Templating: Explore this section that explains extending current themes or applying new ones.
- Python Programming Language documentation: Sphinx was built to support the complete documentation on Python.
- Lettuce another open source project that modifies (greatly) how Sphinx generates HTML.
- developerWorks on Twitter: Follow us for the latest news.