Documentación fácil y atractiva con Sphinx

Haciendo la documentación efectiva y escribible

Alfredo Deza
Software Engineer

16-04-2012

Aprenda cómo crear documentos conservables e impulsados por el estilo que puedan ser automáticamente distribuidos en distintos formatos, como HTML, usando la herramienta Sphinx.

Introducción

Sphinx es una herramienta que permite a los desarrolladores escribir documentos en texto plano para una fácil generación de salidas en formatos que satisfagan diversas necesidades. Esto se vuelve útil usando un Sistema de Control de Versiones para realizar el seguimiento de los cambios. La documentación de texto plano también es útil para colaboradores a través de distintos sistemas. El texto plano es uno de los formatos más portables disponibles actualmente.

Aunque Sphinx está escrito en Python y fue originalmente creado para la documentación del lenguaje de Python, no está centrado necesariamente en el lenguaje y, en algunos casos, ni siquiera es específico del programador. Existen muchos usos para Sphinx, ¡cómo escribir libros enteros!

Resaltado

Sphinx tiene resaltado de código para Python de forma predeterminada, pero también permite la definición de otros lenguajes de programación como C y Ruby.

Piense en Sphinx como una infraestructura de documentación: abstrae las partes tediosas y ofrece funcionalidad automática para solucionar problemas comunes como el indexado de títulos y el resaltado de código especial (si se están mostrando ejemplos de código) con el resaltado de sintaxis apropiado.

Requisitos

Debe sentirse cómo usando un terminal de Linux® o UNIX® (también conocido como emulador de terminal o consola), ya que la interfaz de línea de comandos es la forma principal para interactuar con Sphinx.

Python necesita estar instalado. Ya viene pre-instalado y listo para usarse en todas las distribuciones principales de Linux y en algunos sistemas operativos basados en UNIX (como Mac OSX). Sphinx debe funcionar con Python versiones 2.4, 2.5 y 2.6. Para asegurarse de que tiene Python y una versión válida, ejecute el comando en el Listado 1.

Listado 1. Verificando la versión de Python

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

Sintaxis

Sphinx usa sintaxis de marcación reStructuredText (con algunas adiciones) para proporcionar control de documentos. Si alguna vez ha escrito archivos de texto plano, probablemente ya sepa algo sobre la sintaxis requerida para ser proficiente en Sphinx.

La marcación permite la definición y la estructura de texto para la salida apropiada. Antes de comenzar, vea el Listado 2 para obtener un pequeño ejemplo de la sintaxis de marcación.

Listado 2. Ejemplo de sintaxis de marcación de Sphinx

```
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!
```

Como puede ver, la sintaxis se ve bastante legible en texto plano. Cuando llega el momento de crear un formato específico (como HTML), el título se verá como una cabecera principal y tendrá fuentes más grandes que el subtítulo (como debe ser) y las listas numeradas serán numeradas apropiadamente. Ya cuenta con algo bastante poderoso. Añadir más elementos o cambiar el orden en la lista numerada no afecta la numeración, y los títulos puede cambiar de importancia al sustituir el subrayado utilizado.

Instalación y configuración

La instalación ocurre en la línea de comandos y es sencilla, como se muestra en el Listado 3.

Listado 3. Instalando 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
```

El Listado 3 fue reducido para brevedad, pero proporciona un ejemplo de lo que se puede esperar al instalar Sphinx.

La infraestructura usa una estructura de directorios para tener algo de separación entre el origen (los archivos de texto plano) y la compilación (que se refiere a la salida generada). Por ejemplo, si Sphinx es dirigido para generar un PDF a partir de un origen de documentación, el archivo sería colocado en el directorio de compilación. Este comportamiento puede ser cambiado, pero para mantener la consistencia nos quedaremos con el formato predeterminado.

Hagamos un inicio rápido de un nuevo proyecto de documentación en el Listado 4 que le hará algunas preguntas. Acepte todos los valores predeterminados al presionar **Enter**.

Listado 4. Ejecutando 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).
[...]
```

Elegí "My Project" como el nombre de proyecto que será referenciado en varios lugares. Siéntase libre de elegir un nombre distinto.

Después de ejecutar el comando `sphinx-quickstart` , deben haber archivos en el directorio en funcionamiento que se asemejen a aquellos en el Listado 5.

Listado 5. Listado del directorio en funcionamiento

```
.
### Makefile
### _build
### _static
### conf.py
### index.rst
```

Echemos un vistazo más cercano a cada archivo.

- **Makefile**: Los desarrolladores que han compilado código deben estar familiarizados con este archivo. Si no es así, piense en él como un archivo que contiene instrucciones para compilar salidas de documentación cuando se usa el comando `make` .
- **_build**: Este es el directorio donde los archivos generados serán colocados Después de que una salida específica es desencadenada.

- **_static**: Los archivos que no son parte del código de origen (como las imágenes) son colocados aquí y después se enlazan en el directorio de compilación.
- **conf.py**: Este es un archivo de Python que contiene valores de configuración para Sphynx, incluyendo aquellos seleccionados cuando `sphinx-quickstart` fue ejecutado en el terminal.
- **index.rst**: La raíz del proyecto de documentación. Esto conectará otros archivos si la documentación es dividida en otros archivos.

Iniciándose

En este punto, tenemos a Sphinx instalado apropiadamente, viendo la apariencia de la estructura personalizada, y reconocemos alguna sintaxis básica. Tenga cuidado al entrar justo a la escritura de la documentación. Una carencia de conocimiento sobre el diseño y las salidas puede causar confusión y hacer todo su proceso significativamente más lento.

Eche un vistazo a `index.rst`. Hay una cantidad significativa de información y algo de sintaxis compleja adicional. Para facilitar las cosas y evitar distracciones, incorporaremos un nuevo archivo al listarlo en la sección principal.

Justo después del título principal en el archivo `index.rst` hay un listado de contenido con una declaración `toctree` . La `toctree` es el elemento central para recolectar todos los documentos en la documentación. Si hay otros archivos presentes, pero no listados bajo esta directiva, esos archivos no serían generados con la documentación al momento de la compilación.

Deseamos añadir un nuevo archivo a la documentación y va a ser llamado `example.rst`. Necesita ser listado en `toctree`, pero tenga cuidado. Existe un espaciado que debe ser seguido para que el listado funcione y no necesita la extensión de archivo (`.rst` en este caso). El Listado 6 muestra cómo se debe ver esa lista. Se necesitan tres espacios en blanco para separar el nombre de archivo del margen izquierdo, con una línea blanca después de la opción `maxdepth` .

Listado 6. Ejemplo de toctree en index.rst

```
Contents:

.. toctree::
   :maxdepth: 2

   example
```

No se preocupe sobre las otras opciones en este punto. Por ahora, tenga en cuenta que hay un archivo de índice donde otros archivos separados son listados y pueden alojar documentación válida, y que este listado sigue un cierto orden y espaciado para funcionar.

Recuerde la sintaxis de ejemplo en el Listado 2? Copie y pegue ese ejemplo en el archivo `example.rst` y sálvelo. Ahora estamos listos para generar la salida.

Ejecute el comando `make` y especifique HTML como la salida. Esta salida puede ser usada directamente como un website, ya que tiene todo generado, incluyendo JavaScript y archivos de CSS. Vea el Listado 7.

Listado 7. Salida del comando make html

```
$ 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.
```

Si está interesado en otras opciones que ofrece el comando `make` , vea el Listado 8 para pasarle el distintivo de ayuda y ver una descripción completa.

Listado 8. Opciones de make del listado

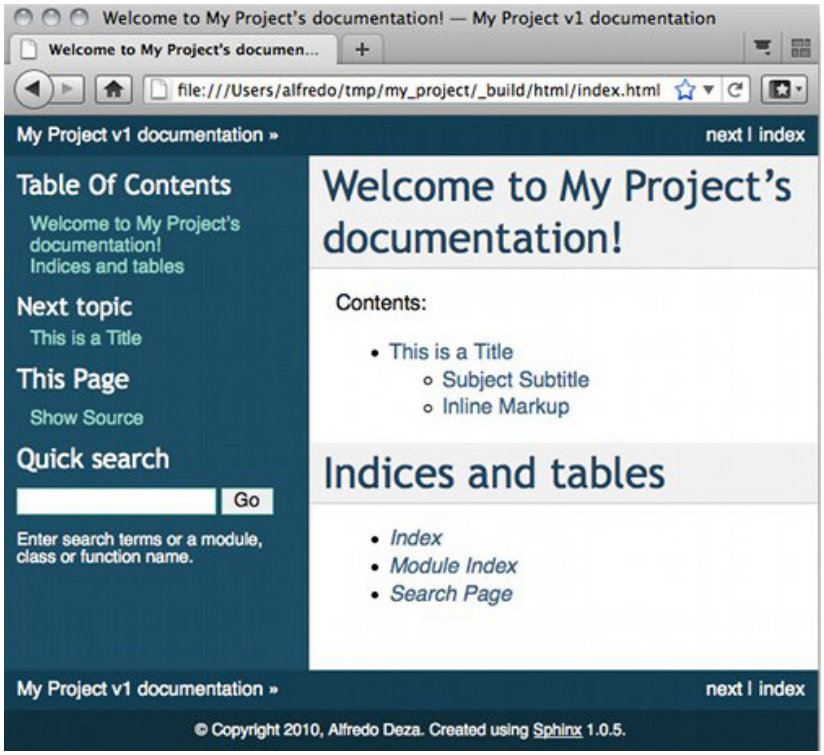
```
$ make -h
Usage: make [options] [target] ...
Options:
[...]
```

Volviéndose estático

Con nuestro primer pase para recolectar HTML de los dos archivos, tenemos un website completamente funcional (estático).

Dentro del directorio `_build` , ahora debe tener dos nuevos directorios: `doctrees` y `HTML`. Estamos interesados en el directorio HTML que contiene todos los archivos necesarios para el sitio de documentación. Abrir el archivo index.html con un navegador debe revelar algo como la Figura 1.

Figura 1. Página principal en HTML estático



Con tan poca información, Sphinx pudo crear bastante. Tenemos un diseño básico con algo de información sobre la documentación del proyecto, una sección de búsqueda, tabla de contenidos, avisos de copyright con nombre y fecha y paginación.

La parte de búsqueda es interesante porque Sphinx ha indexado todos los archivos y con algo de magia de JavaScript ha creado un sitio estático que puede ser buscado.

Recuerde que añadimos `example` como un archivo separado a la documentación en `toctree` en el Listado 6? Puede ver que el título principal muestra una viñeta principal en el índice de contenido y el subtítulo como viñetas de segundo nivel. Sphinx se ha encargado de toda la estructura apropiada.

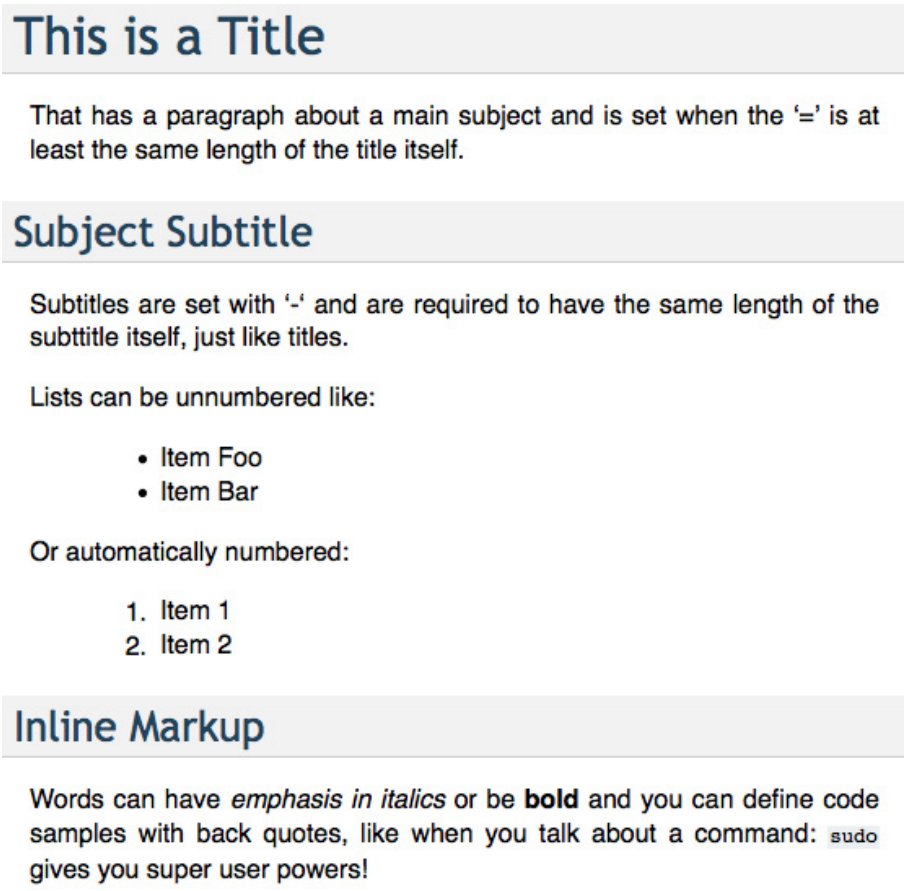
Si la primera vez no tiene éxito...

Después de la modificación adicional, simplemente ejecute el comando make nuevamente para regenerar los archivos.

Todos los enlaces apuntarán a los lugares correctos en la documentación, y los títulos y subtítulos tienen anclas que permiten enlaces directos. Por ejemplo, la sección `Subject Subtitle` contiene un ancla que se ve como `../example.html#subject-subtitle` en el navegador. Como se mencionó anteriormente, la herramienta elimina la preocupación sobre estos diminutos y repetitivos requisitos.

La Figura 2 muestra cómo se ve `example.rst` como un archivo HTML en el sitio estático.

Figura 2. Página de ejemplo como HTML



Volviéndose gráfico

Los párrafos, imágenes y gráficos cortos y concisos añaden interés y legibilidad a la documentación de un proyecto. Sphinx ayuda a mantener la atención de un lector con estos elementos importantes con la posibilidad de añadir archivos estáticos.

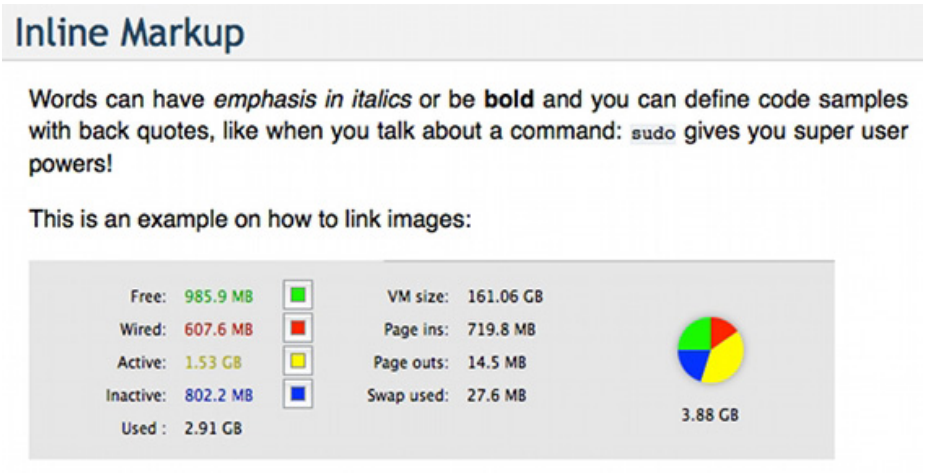
La sintaxis apropiada para añadir archivos estáticos es fácil de recordar. Mientras esté colocando archivos estáticos en el directorio `_static` que Sphinx creó cuando el diseño de documentación fue creado, debe poder referenciarlo sin problemas. En el Listado 9, vea cómo esa referencia se debe ver en el archivo reStructuredText. En este caso, estoy añadiéndolo al fondo de `example.rst`.

Listado 9. Listado estático en example.rst

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

Después de que la documentación es regenerada, la imagen debe ser colocada correctamente en el mismo lugar donde indicamos la ruta hacia el gráfico pequeño JPG sobre la actividad del sistema. Se debe ver similar a la Figura 3.

Figura 3. Imagen de la actividad del sistema



Conclusión

Este artículo ha cubierto las bases para iniciarse con Sphinx, pero hay mucho más por explorar. Sphinx tiene la habilidad de exportar documentación en distintos formatos, pero requieren la instalación de bibliotecas y software extra. Algunos de los formatos que pueden ser generados son: PDF, epub, man (UNIX Manual Pages), y LaTeX.

Para gráficos complejos, existe un plug-in para añadir gráficos de Graphviz a su proyecto de documentación. Una vez tuve que crear uno para una correlación pequeña de redes de oficina y fue muy agradable que pudiera reunir todo en la misma documentación sin tener que usar una herramienta distinta. Como el plug-in de Graphviz, existe una amplia gama de plug-ins (también llamados extensiones) disponibles para Sphinx. Algunos están incluidos, como interSphinx, que le permite enlazar distintos proyectos de Sphinx.

Si la sensación de la salida generada no fue de su agrado, Sphinx incluye muchos temas que pueden ser aplicados para cambiar completamente la forma en que los archivos HTML representan la documentación. Algunos proyectos importantes de código abierto, como Celery y Lettuce, modifican en gran medida la forma en que se ve el HTML al cambiar el CSS y extender las plantillas. Vea la sección Recursos para obtener enlaces a esos proyectos y a la documentación que explica cómo extender y modificar el CSS y diseño predeterminados.

Sphinx cambió la forma en que pensaba sobre la escritura de documentación. Me emocioné cuando pude documentar fácilmente casi todos mis proyectos personales de código abierto y algunos internos. Use Sphinx para recuperar fácilmente información olvidada en su propia documentación.

Descargas

Descripción	Nombre	tamaño
Sample Sphinx project for this article	example_sphinx_project.zip	142KB

Sobre el autor

Alfredo Deza

Alfredo Deza es un ingeniero de software y ex atleta profesional y olímpico con una sólida formación en la administración de sistemas. Le apasiona el software de código abierto y es un ponente regular para grupos de tecnología nacionales y conferencias internacionales como PyCon. En su tiempo libre, intenta mejorar sus habilidades fotográficas y disfruta contribuyendo con proyectos de código abierto.