参考：<https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/>

**使用 sphinx 制作简洁而又美观的文档**

**使文档变得更有效并且可编写**

学习如何使用 Sphinx 工具创建能够以各种格式（如 HTML）自动进行分布的可维护的、样式支配的文档。



在 IBM Bluemix 云平台上开发并部署您的下一个应用。

[**开始您的试用**](https://developer.ibm.com/sso/bmregistration?lang=zh_CN&ca=dwchina-_-bluemix-_-os-sphinx-documentation-_-sidebar)

## 简介

Sphinx 是一种工具，它允许开发人员以纯文本格式编写文档，以便采用满足不同需求的格式轻松生成输出。这在使用 Version Control System 追踪变更时非常有用。纯文本文档对不同系统之间的协作者也非常有用。纯文本是当前可以采用的最便捷的格式之一。

虽然 Sphinx 是用 Python 编写的，并且最初是为 Python 语言文档而创建，但它并不一定是以语言为中心，在某些情况下，甚至不是以程序员为中心。Sphinx 有许多用处，比如可以用它来编写整本书！

## 突出显示

默认情况下，Sphinx 会为 Python 突出显示代码，但它还允许定义其他编程语言，比如 C 和 Ruby。

可以将 Sphinx 想像成为一种文档框架：它会抽象化比较单调的部分，并提供自动函数来解决一些常见问题，比如突出显示标题索引和特殊代码（在显示代码示例时），以及突出显示适当的语法。

### 要求

您应该能轻车熟路地使用 Linux® 或 UNIX® 终端（也称为控制台或终端仿真器），因为命令行界面是与 Sphinx 进行互动的主要方式。

您需要安装 Python。在所有主要的 Linux 发行版和一些基于 UNIX 的操作系统（如 Mac OSX）上预先安装 Python 并做好使用它的准备。Sphinx 支持 Python V 2.4、2.5 和 2.6。要确定您已经安装了 Python 并且安装的是有效版本，请运行 [清单 1](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing1) 中的代码。

##### 清单 1. 检查 Python 的版本

$ python --version

Python 2.6.1

### 语法

Sphinx 使用 reStructuredText 标记语法（和其他一些语法）来提供文档控制。如果您之前编写过纯文本文件，那么您可能非常了解精通 Sphinx 所需的语法。

标记允许为适当的输出实现文本的定义和结构。开始之前，请参见 [清单 2](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing2) 中的一个小的标记语法示例。

##### 清单 2. Sphinx 标记语法示例

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!

正如您所看到的，纯文本格式的语法非常容易读懂。在创建特定格式（如 HTML）时，标题会成为主要目标，其字体会比副标题大一些（理应如此），并且会对编号列表进行适当的编号。您已经拥有一些非常强大的功能。添加更多的项或更改编号列表中的顺序不会影响到编号，而通过替换使用的下划线可以改变标题的重要性。

## 安装和配置

安装是通过命令行进行的，非常简单明了，如 [清单 3](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing3) 所示。

##### 清单 3. 安装 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

为了简便起见，[清单 3](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing3) 中的内容有所缩减，但它提供了在一个在安装 Sphinx 时应执行的操作的示例。

框架使用了一个目录结构来分离源文件（纯文本文件）和构建（指生成的输出）。例如，如果使用 Sphinx 从文档源中生成一个 PDF，那么该文件会放置在构建目录中。您可以更改此行为，但为了获得一致性，我们还是保留了默认格式。

让我们快速启动 [清单 4](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing4) 的一个新的文档项目，系统会通过一些问题提示您如何操作。请按下 **Enter** 键接受所有的默认值。

##### 清单 4. 执行 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).

[...]

我选择 "My Project" 作为项目名称，该名称会在多处被引用。您可以随意选择不同的名称。

运行 sphinx-quickstart 命令后，在工作目录中会出现类似 [清单 5](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing5) 的文件。

##### 清单 5. 工作目录的列表

.

├── Makefile

├── \_build

├── \_static

├── conf.py

└── index.rst

让我们详细研究一下每个文件。

* **Makefile**：编译过代码的开发人员应该非常熟悉这个文件，如果不熟悉，那么可以将它看作是一个包含指令的文件，在使用 make 命令时，可以使用这些指令来构建文档输出。
* **\_build**：这是触发特定输出后用来存放所生成的文件的目录。
* **\_static**：所有不属于源代码（如图像）一部分的文件均存放于此处，稍后会在构建目录中将它们链接在一起。
* **conf.py**：这是一个 Python 文件，用于存放 Sphinx 的配置值，包括在终端执行 sphinx-quickstart 时选中的那些值。
* **index.rst**：文档项目的 root 目录。如果将文档划分为其他文件，该目录会连接这些文件。

## 入门指南

此时，我们已经正确安装了 Sphinx，查看了默认结构，并了解了一些基本语法。不要直接开始编写文档。缺乏布局和输出方面的知识会让您产生混淆，可能耽误您的整个进程。

现在来深入了解一下 index.rst 文件。它包含大量的信息和其他一些复杂的语法。为了更顺利地完成任务并避免干扰，我们将合并一个新文件，将它列在主要章节中。

在 index.rst 文件中的主标题之后，有一个内容清单，其中包括 toctree 声明。toctree 是将所有文档汇集到文档中的中心元素。如果有其他文件存在，但没有将它们列在此指令下，那么在构建的时候，这些文件不会随文档一起生成。

我们想将一个新文件添加到文档中，并打算将其命名为 example.rst。还需要将它列在 toctree 中，但要谨慎操作。文件名后面需要有一个间隔，这样文件名清单才会有效，该文件不需要文件扩展名（在本例中为 .rst）。[清单 6](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/" \l "listing6) 显示该列表的外观。在文件名距离左边距有三个空格的距离，maxdepth 选项后面有一个空白行。

##### 清单 6. index.rst 中的 toctree 示例

Contents:

.. toctree::

:maxdepth: 2

example

此时，不用担心其他选项。目前，注意到了有一个列出其他单独的文件的索引文件，该文件可存储有效文档，因此，该列表有一定的顺序和空格，才能使该列表变得有效。

还记得 [清单 2](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing2) 中的示例语法吗? 请复制该示例，将它粘贴到 example.rst 文件中并保存它。现在我们准备生成输出。

运行 make 命运，并将 HTML 指定为输出格式。可直接将该输出用作网站，因为它包含了生成的所有内容，包括 JavaScript 和 CSS 文件。请参见 [清单 7](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing7)。

##### 清单 7. make html 命令的输出

$ 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.

如果您对 make 命令提供的其他选项感兴趣，请参见 [清单 8](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing8)，将帮助标志传至此处，并查看完整的描述。

##### 清单 8. 列示 make 选项

$ make -h

Usage: make [options] [target] ...

Options:

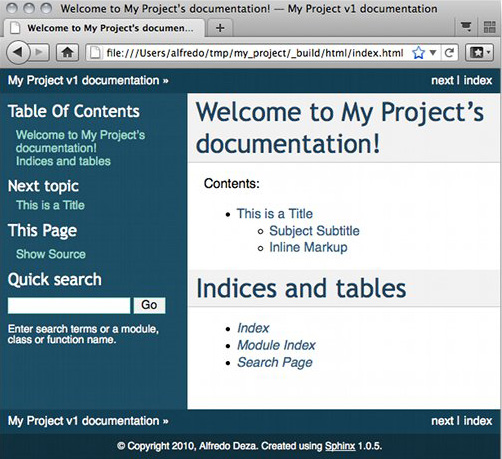
[...]

### 生成静态网站

随着我们完成第一步操作，从两个文件中生成 HTML 之后，我们就拥有一个完整的函数式（静态）网站。

在 \_build 目录内，现在应该有两个目录：doctrees 和 HTML。我们对于这个存储了文档网站所需的全部文件的 HTML 目录很感兴趣。使用浏览器打开 index.html 文件，就会发现如 [图 1](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#fig1) 所示的内容。

##### 图 1. 静态 HTML 形式的主页



虽然信息很少，但 Sphinx 能够创建很多内容。我们拥有一个基本布局，该布局包含有关项目文档、搜索部分、内容表、附带名称和日期的版权声明、页码的一些信息。

搜索部分非常有趣，因为 Sphinx 已经为所有文件建立索引，并使用 JavaScript 的一些强大功能创建了一个可搜索的静态网站。

还记得我们已将 example 作为一个单独的文件添加至 [清单 6](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing6) 的 toctree 中的文档吗？您可以看到，主标题显示为内容索引中的主要项目符号，副标题显示为二级项目符号。Sphinx 小心维护着让整个结构保持正确。

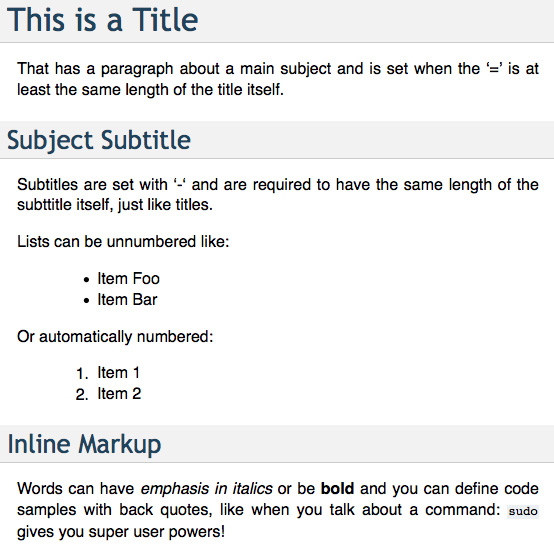
## 如果一开始就没有成功...

进行一些修改后，只需再次运行 make 命令，即可生成这些文件。

所有的链接都指向文档中的正确位置，并且标题和副标题均有定位点，允许直接进行链接。比如，Subject Subtitle 部分在浏览器中有一个类似 ../example.html#subject-subtitle 的定位点。如前所述，该工具消除了我们对这些琐碎的、重复的需求的顾虑。

[图 2](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#fig2) 显示了 example.rst 如何显示为静态网站中的 HTML 文件。

##### 图2. HTML 页面示例



### 添加图形

简明的段落、图像和图形都为项目文档增加趣味性和可读性。Sphinx 有助于利用这些有可能添加了静态文件的主要元素来吸引读者的注意。

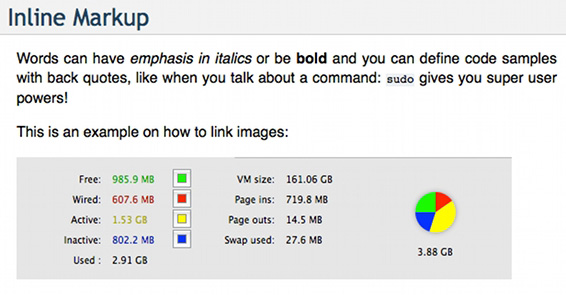
添加静态文件的正确语法很容易记忆。只要将静态文件放置 \_static 目录（Sphinx 在创建文档布局时创建了该目录）中，就可以轻松地对其进行引用。在 [清单 9](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#listing9)，查看 reStructuredTex 文件中的引用应该是什么样子的。在本例中，我将其添加在 example.rst 的底部。

##### 清单 9. example.rst 的静态清单

.. image:: \_static/system\_activity.jpg

生成文档之后，应将图像正确放置在我们为有关系统活动的 JPEG 小图像指定的地方。它看上起应该类似于 [图 3](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#fig3)。

##### 图 3. 系统活动图像



## 结束语

本文介绍了开始使用 Sphinx 的一些基础知识，但仍有许多内容有待我们探索。Sphinx 能够用不同的格式导出文档，但要求安装额外的库和软件。可生成的格式包括：PDF、epub、man (UNIX Manual Pages) 和 LaTeX。

对于复杂的图形，有一个插件可将 Graphviz 图形添加至您的文档项目。我曾经不得不为一个小型办公网络地图创建一个插件，但它表现相当出色，无需使用其他工具，便可在同一文档中获取所有的东西。与 Graphviz 插件类似，有大量可用于 Sphinx 的插件（亦称为扩展）。Sphinx 提供了一些插件，比如 interSphinx，该插件允许您链接不同的 Sphinx 项目。

如果生成的输出的外观不符合您的喜好，Sphinx 还提供了许多主题，可应用它们来完全改变 HTML 文件呈现文档的方式。一些重要的开源项目，如 Celery 和 Lettuce，通过更改 CSS 并扩展模板完全更改了 HTML 的外观。请参阅 [参考资料](https://www.ibm.com/developerworks/cn/opensource/os-sphinx-documentation/#resources) 部分，以获取这些项目的链接、解释如何扩展的文档的链接以及修改默认 CSS 和布局的链接。

Sphinx 改变了我对编写文档的看法。从一开始的毫无灵感，到现在能够轻易编制我的几乎所有的个人开源项目以及少数内部项目，我感到非常兴奋。使用 Sphinx 可轻松检索遗忘在您自己文档中的信息。