writebysphinx

发布 *1.0*

2019 年 10 月 28 日

目录

CHAPTER 1

sphinx 简介

1.1 sphinx 是什么？

参考 sphinx 中文版使用手册 ，介绍 sphinx。

Sphinx 是一种文档工具，它可以令人轻松的撰写出清晰且优美的文档, 由 Georg Brandl 在 BSD 许可证下开发. 新版的 Python 文档就是由 Sphinx 生成的，并且它已成为 Python 项目首选的文档工具, 同时它对 C/C++ 项目也有很好的支持; 并计划对其它开发语言添加特殊支持. 本站当然也是使用 Sphinx 生成的，它采用 reStructuredText! Sphinx 还在继续开发. 下面列出了其良好特性, 这些特性在 Python 官方文档中均有体现:

- **丰富的输出格式**: 支持 HTML (包括 Windows 帮助文档), LaTeX (可以打印 PDF 版本), manual pages（man 文档）, 纯文本
- **完备的交叉引用**: 语义化的标签, 并可以自动化链接函数, 类, 引文, 术语及相似的片段信息
- **明晰的分层结构**: 可以轻松的定义文档树, 并自动化链接同级/父级/下级文章
- **美观的自动索引**: 可自动生成美观的模块索引
- **精确的语法高亮**: 基于 Pygments 自动生成语法高亮
- **开放的扩展**: 支持代码块的自动测试, 并包含 Python 模块的自述文档 (API docs) 等

Sphinx 使用 reStructuredText 作为标记语言, 可以享有 Docutils 为 reStructuredText 提供的分析，转换等多种工具.

Sphinx 的官网英文版手册地址：http://www.sphinx-doc.org/en/master/contents.html 。

1.2 为什么用 sphinx 写文档？

用 Word、PPT、Excel、Pages、Numbers 也很棒，但是 sphinx 用纯文本编辑

- 方便用 Git/Svn 等版本管理工具实现协作
- 通过 Web 发布，有利于文档的传播，无需安装软件即可查看
- 有更方便的索引和引用，支持检索查询
- 支持用脚本画图，排版，高效率低成本提升文档美观度
- 高度的可扩展性，包括自定义样式、提供自动函数来解决一些常见问题等等

1.3 为什么不推荐 Markdown 作为大型文档协作工具？

Sphinx 缺省使用 reStructuredText 语法，尽管它也支持 Markdown，但为什么不推荐用 Markdown，以下是引用别的 blog 提供的观点。http://ericholscher.com/blog/2016/mar/15/dont-use-markdown-for-technical-docs/

Markdown 是互联网上最普遍使用的轻量级标记语言。对于写博客和评论这类的任务，用 Markdown 很棒。不过最近技术社区的人员开始用它来写文档。

下面我列出一些反对使用 Markdown 的观点，希望能帮助你决定是否适合使用 Markdown。如果你在考虑使用 Markdown，我希望你也可以关注下 Asciidoctor 和 Sphinx，我发现用它们写文档更好。人们选择 Markdown 是因为它可以很简单的处理一些基本任务。开发者选择它是因为 GitHub 支持它，尽管 GitHub 支持 9 种不同的标记语言，包括 Asciidoc 和 reStructuredText。但是当文档从几页逐渐增长到大的文档集，Markdown 很快就崩溃并成为累赘。下面是这种情况发生的原因。

缺乏一个标准

最初，Markdown 是由 John Gruber 写的 initial implementation 来定义，但并未明确规定行为规范。

随着 Markdown 越来越流行，越来越多的站点开始支持 Markdown 的实现，这些站点是用其它语言写成的，因此产生了更多的 Markdown 实现。所有这些实现有轻微差别，但都达不到令人接受的程度。

比如有些实现要求开头有一个空格，但另外的一些不做要求：

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

还有一些小问题使得 Markdown 很难在不同的站点和版本之间移植

在过去的几年里，Commonmark 已经发展成标准化的 Markdown。这很好，应该解决很多问题，但是却没人采用它。

风格

Markdown 缺乏采用的主要原因是这些年来其一直在变化。起初，Markdown 功能很有限，每当有流行的工具在 Markdown 上实现的时候，都会有一个特定的风格。听起来不错，是吧？问题是每一个工具都形成了不同的风格，甚至连实现相似任务的工具都有不同的语法。

举个例子，在 Markdown Extra 中代码块是下面这种样式：

```
~~~ .python
import antigravity
~~~
```

这将 python 类应用于输出的 HTML 块

然而，同样的情况在 GitHub Flavored Markdown 中是这样的：

```
```python
import antigravity
```
```

这会将语法高亮显示应用于实际呈现的 HTML 输出。

相似的概念，不同的语法，但都是 Markdown。

缺乏扩展

其它的标记语言，可以对其进行扩展以提供需要的功能。它们在语法上有增添新功能同时又不会违反初始规范的机制。

比如 reStructuredText，具有内嵌和块级别的标记：

```
.. contents:: All the stuff I'm going to talk about
:depth: 2

Please look at :rfc:`1984` for more information.
This is implemented in our codebase at :class:`Example.Encryption`.
```

可以了解更多关于 rfc，class 和 contents 的概念。

作为一个使用 rST 或 Asciidoctor 的开发者，我可以以一种简单，可插入的方式增添新的标记。我不必改变语言的解析方式，而且我还可以通过标准扩展机制向其它用户分享那些新功能。

在不同的版本间进行移植这些功能，用 Markdown 可办不到。

注解: 注意：CommonMark 正在开发可扩展性的语法，但还没有实现。

缺乏语义化

尽管很多人添加了很多扩展，但都不够语义化。这意味着不能写 Class 或 Warning，只能写文本。因此很多人直接将 HTML 嵌入到 Markdown 中：

而在 reStructuredText 中，你可以写成：

```
<div class="warning">
This is a Warning!
</div>
```

这在 HTML,PDF，甚至任何创建的输出格式中都可以合适的显示为一个 Warning.

```
.. warning:: This is a Warning!
```

语义化标记可以将所写的文字与它们的显示方式完全分开。

缺乏语义化将导致问题，原因如下：

- 现在 Markdown 依赖于显示中的特定 CSS 类，这意味着编写者必须考虑如何设计页面。
- 内容不可再移植到其他输出格式（PDF 等）。
- 转换到其他标记工具和页面设计变得更加困难。

注解： 注意：在我的博文 Semantic Meaning in Authoring Documentation 中有关于语义化更详细的介绍。

锁定和缺乏可移植性

风格的多种多样及缺乏语义化导致锁定。一旦创建了大的 Markdown 文档集，就很难将它们迁移到另一个工具上，即使该工具宣称支持 Markdown！自定义的 HTML 类和奇怪风格的扩展组成的文档集，除了在当前的工具和设计外，在其它地方都行不通。

同时也无法轻易将 Markdown 迁移到其他标记语言（Asciidoc 或 RST），因为 Pandoc 和其他转换工具不支持您风格的扩展。很多人选择 Markdown 是因为他们认为他们可以在稍后迁移到其他工具或其它标记语言。

Markdown 绝对是最低的共同标准，除非文档集足够小，否则你所需要的东西都不在基本语法中。任何有意义的文档都需要扩展，而一旦使用 Markdown 各种各样风格的扩展，你将失去所有可移植性的优势。

结论

我认为 CommonMark 是向前迈出的一大步，如果它被更广泛地使用，并且增加对扩展的支持，我会全心全意地推荐它作为解决这个问题的方法。我不能拥护 Markdown 目前的生态系统，并且我认为它很大程度上阻止了人们使文档变得更好。

我希望我们可以开始推进更加标准化的语言集合，包括 CommonMark，reStructuredText 和 Asciidoc，在我们使用的工具套件中全面支持他们。目前，Sphinx 和 Asciidoctor 是很好的替代品。它们语音内部内置了更多的扩展，并且含有用于构建当今文档更完整的工具。

Markdown 更像是一个概念，而不是实现。它通常意味着 “在看起来与 Markdown 类似的语言上的一组互不兼容的扩展”。当创建大的文档集时，它显然不是正确的工具。

1.4 为什么用脚本画图？

效率

用脚本画图，一开始需要有熟悉的过程，可能比你用现有工具要慢，但一旦你掌握了这些工具，特别是 uml 等标准图，你的画图的速度会大大加快，因为你只需要关注图的内容和逻辑，排版和美化等问题由工具自动解决了。当然为了布局合理，画图的逻辑结构也是会有影响的，这可以迫使你对于图的内容逻辑更多的思考。

协作

如果用画图工具，并采用图片标记插入，可能比较符合你原来的创作习惯，但这种方式将导致协作和版本管理的困难，因为别人无法通过修改文本文件来生成新的图片。

CHAPTER 2

安装设置

你不必安装本文所列的所有程序，实际上你只需要一个文本编辑器就能参与编写 sphinx 文档，通过 git 或 svn 等版本管理工具，整个团队可以进行有效协作，而将文本编译为 html 或 pdf，可以交给服务器去做。

如果你要重度参与编写 sphinx 文档，那么安装本文所列工具后，你将获得非常强大的文档编辑的能力，包括布局、画图、建立索引、自动化格式（尤其是各种代码格式）、内容参数化以及一些自动化功能。

2.1 安装 Python 虚拟机

为什么要用虚拟机？这里参考一个博客的说法如下

在实际项目开发中，我们通常会根据自己的需求去下载各种相应的框架库，如 Scrapy、Beautiful Soup 等，但是可能每个项目使用的框架库并不一样，或使用框架的版本不一样，这样需要我们根据需求不断的更新或卸载相应的库。直接怼我们的 Python 环境操作会让我们的开发环境和项目造成很多不必要的麻烦，管理也相当混乱。

场景 1：项目 A 需要某个框架 1.0 版本，项目 B 需要这个库的 2.0 版本。如果没有安装虚拟环境，那么当你使用这两个项目时，你就需要来回的卸载安装了，这样很容易就给你的项目带来莫名的错误。

场景 2：公司之前的项目需要 python2.7 环境下运行，而你接手的项目需要在 python3 环境中运行，想想就应该知道，如果不使用虚拟环境，这这两个项目可能无法同时使用，使用 python3 则公司之前的项目可能无法运行，反之则新项目运行有麻烦。而如果虚拟环境可以分别为这两个项目配置不同的运行环境，这样两个项目就可以同时运行。

下面是安装 Python 虚拟机的简单方法：

```
# 安装 python 虚拟环境应用
$ pip install virtualenv
# 或者用 pip3 install virtualenv

# 新建 python 虚拟机目录，并创建虚拟机，我这里用 sphinx-doc 作为虚拟机名字
MacBook:~ vik.qian$ mkdir pyenv
MacBook:pyenv vik.qian$ cd pyenv
MacBook:pyenv vik.qian$ virtualenv sphinx-doc

# 激活虚拟机
MacBook:pyenv vik.qian$ source sphinx-doc/bin/activate
(sphinx-doc) MacBook:pyenv vik.qian$
# 前面多了个括号，说明进入了虚拟机

# 退出虚拟机
(sphinx-doc) MacBook:pyenv vik.qian$ deactivate
```

2.2 安装 sphinx

2.2.1 安装到 Linux

当然是通过 官网链接 说明安装，以下 English 内容引用官网。

注解: 使用命令请在相应的 python 虚拟机运行安装

Debian/Ubuntu

Install either `python3-sphinx` (Python 3) or `python-sphinx` (Python 2) using **`apt-get`**:

```
$ apt-get install python3-sphinx
```

If it not already present, this will install Python for you.

RHEL, CentOS

Install `python-sphinx` using **`yum`**:

```
$ yum install python-sphinx
```

If it not already present, this will install Python for you.

Other distributions

Most Linux distributions have Sphinx in their package repositories. Usually the package is called `python3-sphinx`, `python-sphinx` or `sphinx`. Be aware that there are at least two other packages with `sphinx` in their name: a speech recognition toolkit (*CMU Sphinx*) and a full-text search database (*Sphinx search*).

2.2.2 安装到 macOS

Sphinx can be installed using Homebrew, MacPorts, or as part of a Python distribution such as Anaconda.

Homebrew

```
$ brew install sphinx-doc
```

For more information, refer to the package overview.

MacPorts

Install either `python36-sphinx` (Python 3) or `python27-sphinx` (Python 2) using `port`:

```
$ sudo port install py36-sphinx
```

To set up the executable paths, use the `port select` command:

```
$ sudo port select --set python python36
$ sudo port select --set sphinx py36-sphinx
```

For more information, refer to the package overview.

Anaconda

```
$ conda install sphinx
```

2.2.3 安装到 Windows

Most Windows users do not have Python installed by default, so we begin with the installation of Python itself. To check if you already have Python installed, open the *Command Prompt* (`âŠžWin-r` and type `cmd`). Once the command prompt is open, type `python --version` and press Enter. If Python is installed, you will

see the version of Python printed to the screen. If you do not have Python installed, refer to the Hitchhikers Guide to Python’ s Python on Windows installation guides. You must install Python 3.

Once Python is installed, you can install Sphinx using **pip**. Refer to the *pip installation instructions* below for more information.

Installation from PyPI

Sphinx packages are published on the Python Package Index. The preferred tool for installing packages from *PyPI* is **pip**. This tool is provided with all modern versions of Python.

On Linux or MacOS, you should open your terminal and run the following command.

```
$ pip install -U sphinx
```

On Windows, you should open *Command Prompt* (**âŠžWin-r** and type **cmd**) and run the same command.

```
C:\> pip install -U sphinx
```

After installation, type **sphinx-build --version** on the command prompt. If everything worked fine, you will see the version number for the Sphinx package you just installed.

Installation from *PyPI* also allows you to install the latest development release. You will not generally need (or want) to do this, but it can be useful if you see a possible bug in the latest stable release. To do this, use the **--pre** flag.

```
$ pip install -U --pre sphinx
```

2.3 安装 VSCode

Visual Studio Code 可能是我使用过的最酷的文档编辑器，超级好用，且有很多插件，强烈推荐使用 VSCode 写 sphinx 文档。正如，我现在在使用的，VSCode 也支持 reStructuredText 语言，且支持实时预览 – 如果你习惯于写 Markdown 时实时预览的话，你会喜欢的。VSCode 安装地址 https://code.visualstudio.com/ 。

注解: 如果系统有多个版本 python，配置不对可能导致 VSCode 渲染 html 失败，可以在 VSCode setting 中设置 python 的 path，从而使用正确的版本：`"python.pythonPath": "/Users/vik.qian/pythonvenvs/sphinx-doc/bin/python3`，如下图：

提示: 注意：VSCode 可以安装许多插件，一些插件会影响编辑习惯，比如如果安装了默认 VIM 风格的输入方式，那么你必须按照 VIM 方式进行输入，如果不习惯则可以考虑禁用这些插件。

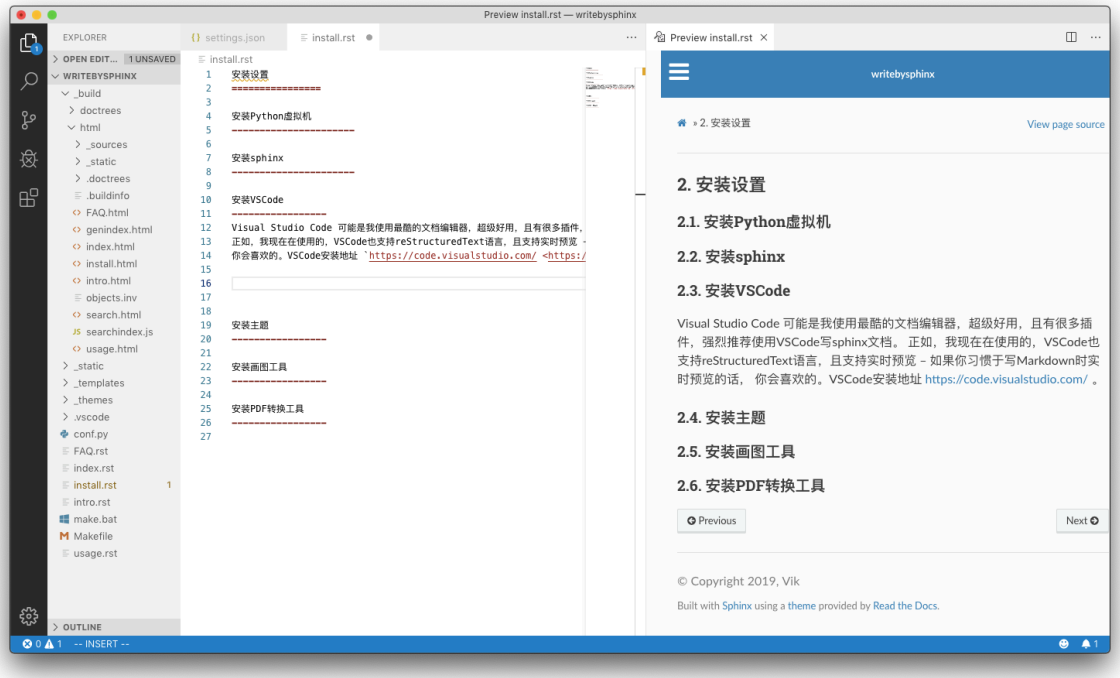


图 1: 用 Visual Studio Code 编辑 reStructuredText，并渲染为 html

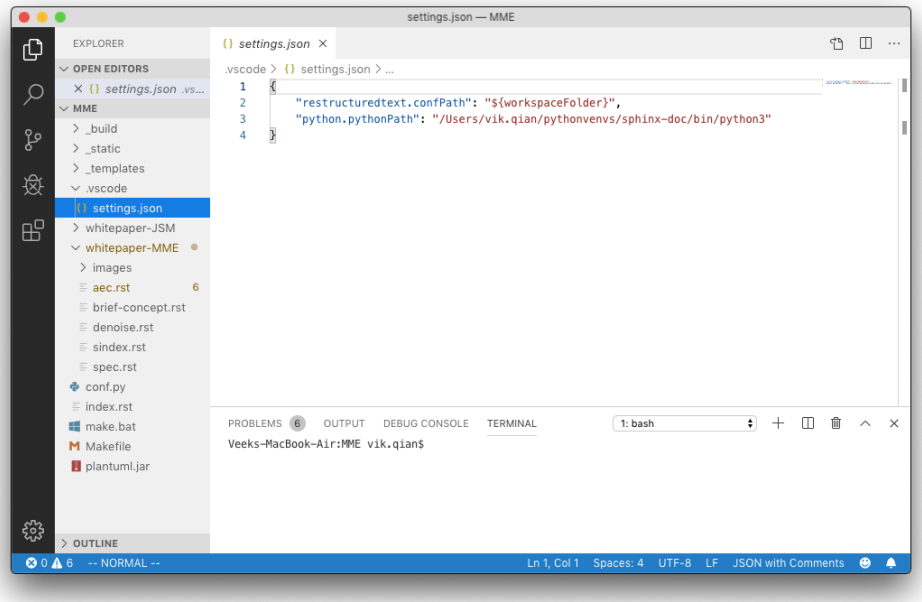


图 2: setting.json 中设置 python 版本

2.4 安装主题

sphinx 默认主题 alabaster 太素，可以选一个最经典的主题 sphinx_rtd_theme ，直接用这个命令装：

```
pip install sphinx_rtd_theme
```

注解: 命令请在相应的 python 虚拟机运行安装

2.4.1 在 conf.py 配置

```
import sphinx_rtd_theme
html_theme = 'sphinx_rtd_theme
```

2.4.2 解决表格过宽问题

主题 *sphinx_rtd_theme* 的表格有个内容无法换行的 bug，导致许多表格过宽，参考这个 网站 的方法，重载 css 并在 conf.py 配置后，就可以解决。

The **Read the Docs** Sphinx theme contains a bug that causes text in table cells not to wrap. This results in very wide tables with horizontal scroll bars.

You can workaround this issue by defining a custom CCS override file.

1. Change into your documentation directory. This is usually where the `index.rst` and `conf.py` files are located:

   ```
   $ cd doc
   ```

2. If it does not already exist, create a `_static` directory:

   ```
   $ mkdir _static
   ```

3. Create a `theme_overrides.css` file in the `_static` directory:

   ```
   $ touch _static/theme_overrides.css
   ```

4. Open the `theme_overrides.css` file and add the following CSS:

   ```
   /* override table width restrictions */
   @media screen and (min-width: 767px) {
   ```

(下页继续)

(续上页)

```
   .wy-table-responsive table td {
      /* !important prevents the common CSS stylesheets from overriding
         this as on RTD they are loaded after this stylesheet */
      white-space: normal !important;
   }

   .wy-table-responsive {
      overflow: visible !important;
   }
}
```

5. Open the `conf.py` file and add the following configuration options:

```
html_static_path = ['_static']

html_context = {
    'css_files': [
        '_static/theme_overrides.css',  # override wide tables in RTD theme
        ],
     }
```

6. Build your documentation using Sphinx and check the tables; cells should now wrap correctly.

2.5 安装画图工具

文档中的图最好都能够用脚本描述，工具自动画，这样有利于版本维护，推荐使用 plantuml 和 Graphviz，前者用来画 UML 图，后者可以画状态转移图。

2.5.1 安装 sphinxcontrib-plantuml

plantuml.jar 是个 java 程序，运行起来有点慢，但功能强大，也还能接受。

注解: 使用命令请在相应的 python 虚拟机运行安装

---

pip 安装

```
pip install sphinxcontrib-plantuml
```

下载安装

点击官网页面“Download”链接下载。下载完，解压，运行命令：

```
python setup.py install
```

2.5.2 修改 Sphinx 文档配置

找到文档目录中的 conf.py 文件，添加 extension 和 plantuml 调用指令。

```
import os

# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = ['sphinxcontrib.plantuml',
              'sphinx.ext.graphviz']

# 设置 plantuml.jar 路径
currentpath = os.getcwd() + '/'
plantuml = 'java -jar ' + currentpath + 'plantuml.jar'
```

2.5.3 下载 plantuml.jar 文件

下载 plantuml.jar 文件，plantuml 放在当前工程目录（因为 conf.py 中，假设是放在当前工作目录 `currentpath = os.getcwd() + '/'` ）。

2.5.4 安装 graphviz

Graphviz 是一个很轻量的软件，性能非常好，用起来飞一样。

注解： 使用命令请在相应的 python 虚拟机运行安装

Graphviz 的安装说明网站 ，最简单的安装命令如下

```
$ pip install graphviz
```

需要在 conf.py 中进行如下配置：

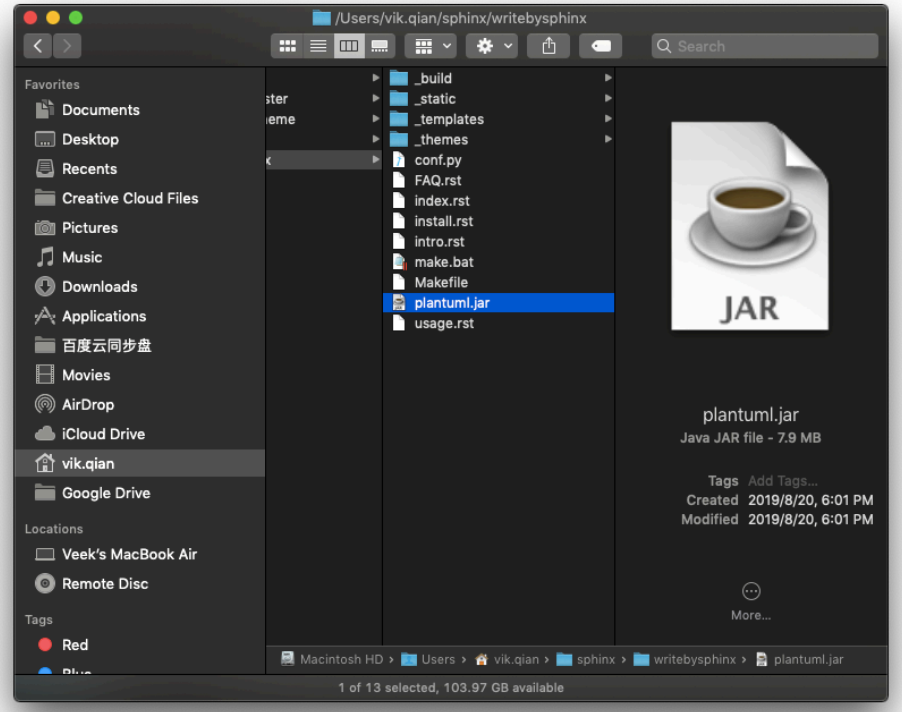


图 3: 注意 plantuml 放对目录以便 java 命令能找到它

```
extensions = ['sphinxcontrib.plantuml',
              'sphinx.ext.graphviz']

# 设置 graphviz_dot 布局风格
# dot 默认布局方式，主要用于有向图
# neato 基于 spring-model(又称 force-based) 算法    基于斥力 + 张力的布局
# twopi 径向布局
# circo 圆环布局
# osage
# fdp 用于无向图
# sfdp 用于无向图
graphviz_dot = 'dot'

# 设置 graphviz_dot_args 的参数，这里默认了默认字体
graphviz_dot_args = ['-Gfontname=Georgia',
                     '-Nfontname=Georgia',
                     '-Efontname=Georgia']
# 输出格式，默认 png，这里我用 svg 矢量图
graphviz_output_format = 'svg
```

2.6 安装 PDF 转换工具

有时候文档需要发布到 pdf，这篇博客写了如何转 pdf ，我还没试过，主要是 MacTeX/TeXLive 太大，适合装在服务器端。

另外，如果你使用 readthedocs 来发布文档，那么 readthedocs 也会同时自动生成 pdf，还是比较方便的。

2.7 安装 Breathe

如果你计划使用 doxygen 从代码生成文档，并导入 sphinx，那么使用工具 Breathe，作为一个“桥梁”可以实现这个目标。如果你没有这个需求，那么安装 Breathe 不是必须的，Breathe 安装命令：

```
$ pip install breathe
```

安装完成后，在 conf.py 中进行如下配置，其中 yourcodefile 是由 doxygen 生成的一系列 xml/xsd/xslt 文件：

```
extensions = ['sphinxcontrib.plantuml',
              'sphinx.ext.graphviz',
              'breathe']
```

(下页继续)

(续上页)

```
# 设置 doxyxml 路径
breathe_projects = {
            "yourcodefile": '../doxyxml/yourcodefile'
}
```

CHAPTER 3

基本用法

这里只是介绍我在用这个工具一周内碰到的常用问题，比较浅显，但应该是碰到概率较高的情况，做个记录供参考。更详细的参考应查阅 sphinx 手册 ，这个是 英文版 。

3.1 基本布局

使用 align、width 设置，使得布局舒适

丑的做法 (图片太大)

```
.. image:: _static/justalk-ct.png
```


17:40
Messages
满
wxm王兴满
快来加入会议!
7/27/19
Lq刘谦
6 mins 6 secs
7/27/19
方宇宗
1 min 46 secs
7/24/19
Zb周波
canceled
7/20/19
黄连港
canceled
7/20/19
王倩
4 mins 25 secs
7/18/19
吉
周吉
+8613884414644 has added y...
7/14/19

好的做法（图片适中，并居中）

```
.. image:: _static/justalk-ct.png
    :align: center
    :width: 70%
```


17:40
Messages
满
wxm王兴满
快来加入会议!
7/27/19
Lq刘谦
6 mins 6 secs
7/27/19
方宇宗
1 min 46 secs
7/24/19
Zb周波
canceled
7/20/19
黄连港
canceled
7/20/19
王倩
4 mins 25 secs
7/18/19
吉
周吉
+8613884414644 has added y...
7/14/19
俞谢益
canceled
7/13/19
cyl陈一乐
Join the meeting now!
7/8/19
1

3.2 制表格技巧

Grid table 比较难用，除非必要，尽量使用 list-table 或 simple talbe 或 csv-table。

list-table 和 simple table 非常简单，而 csv-table 最紧凑，也可以考虑在 excel 表格中编辑好后，存为 csv 格式再 copy 过来，比如：

```
.. csv-table:: bitrate per resolution -- csv talbe sample
    :header: " 分辨率","Lowest br", "Enough br", "High br"

    1080p,"1,546","2,782","4,637"
    720p,876,"1,577","2,628"
    Quad-VGA,"1,072","1,929","3,215"
    XGA,784,"1,411","2,352"
    SVGA,555,999,"1,665"
    VGA,406,731,"1,218"
    Quarter-VGA,154,277,462
    CIF,187,336,561
    QCIF,71,127,212
    subQCIF,43,77,128
```

呈现如下：

表 1: bitrate per resolution – csv talbe sample

分辨率	Lowest br	Enough br	High br
1080p	1,546	2,782	4,637
720p	876	1,577	2,628
Quad-VGA	1,072	1,929	3,215
XGA	784	1,411	2,352
SVGA	555	999	1,665
VGA	406	731	1,218
Quarter-VGA	154	277	462
CIF	187	336	561
QCIF	71	127	212
subQCIF	43	77	128

Sphinx 默认的主题有个表格过宽的问题，如下图。

通过重载 css 可以解决这个问题，详见解决表格过宽问题。解决后，表格中内容会自动换行：

序号	支持的操作系统	说明
1	Windows	支持Windows XP, 7, 10
2	Mac OSX	支持
3	Android	支持Android 2.3 up to 9.x
4	iOS	支持iOS 5.0 up to 12.x
5	Linux	可支持X86、ARM系统，需要商用推动，定制工作量根据具体环境

图 1: 表格过宽，导致有横向滑动条

序号	支持的操作系统	说明
1	Windows	支持 Windows XP, 7, 10
2	Mac OSX	支持
3	Android	支持 Android 2.3 up to 9.x
4	iOS	支持 iOS 5.0 up to 12.x
5	Linux	可支持 X86、ARM 系统，需要商用推动，定制工作量根据具体环境评估

3.3 注意空行和空格

熟悉 python 开发的都知道，源码中空行和空格都是关键的“代码”，有时候少一行或少一个空格都会导致语法错误。

例如，要保留原输出格式，以下写法会不按照你的想法来：

```
::
    没有空行，语法错误!
```

实际输出：

:: 没有空行，语法错误!

正确的应该是有一个空行：

```
::

    有空行，才是正确的语法!
```

实际输出：

```
有空行，才是正确的语法!
```

3.4 注意标点

```
`和 ' 和 ‘ 是有区别的，要注意区分
`是反引号，大量用于关键标记，比如 ``为内联代码样式 ``
' 只是一个单引号
 ‘只是一个中文输入的单引号
```

3.5 使用 python 设置路径

由于 sphinx 最初是用来编写 python 的文档的，conf.py 本身就是 python 脚本，因此最简单的，使用 python 在这里可以干一点事。比如，使用 plantuml 时，需要在 conf.py 指定 plantuml.jar 的地址，每个用户存放的地址可能不同，那么可以通过 python 命令获取当前地址来实现文档统一。

```
import os
# 设置 plantuml.jar 路径
currentpath = os.getcwd() + '/'
plantuml = 'java -jar ' + currentpath + 'plantuml.jar'
```

提示： 以上简单脚本避免了设置绝对地址的硬编码，实现版本统一。

CHAPTER 4

发布文档

4.1 通过 Github 和 readthedoc 发布

基本流程是：用 Sphinx 生成文档，GitHub 托管文档，再导入到 ReadtheDocs。不重复造轮子，以下攻略引用于 一个博客 。

将文档托管到版本控制系统比如 github 上面，push 源码后自动构建发布到 readthedoc 上面，这样既有版本控制好处，又能自动发布到 readthedoc，实在是太方便了。

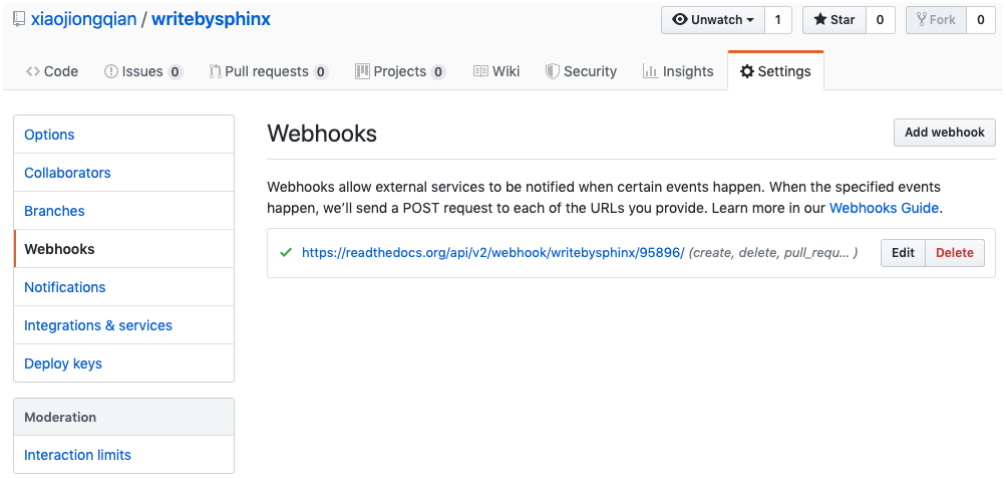
先在 GitHub 创建一个仓库名字叫 writebysphinx，然后在本地.gitignore 文件中添加 build/目录，初始化 git，commit 后，添加远程仓库。

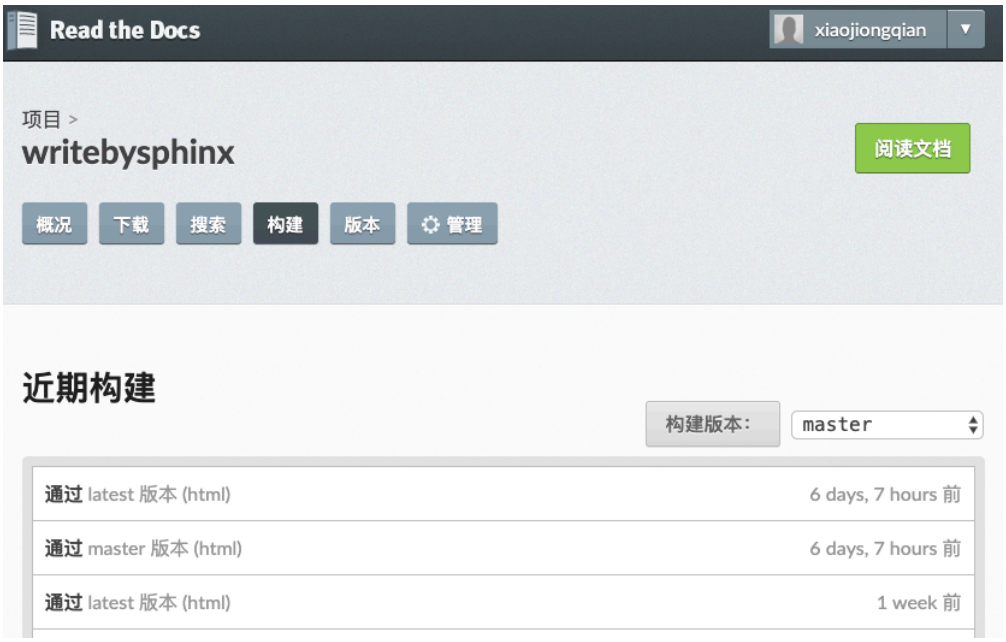
具体几个步骤非常简单，参考官方文档：https://github.com/rtfd/readthedocs.org:

1. 在 Read the Docs 上面注册一个账号
2. 登陆后点击“Import”
3. 给该文档项目填写一个名字比如“writebysphinx”，并添加你在 GitHub 上面的工程 HTTPS 链接，选择仓库类型为 Git

4. 其他项目根据自己的需要填写后点击“Create”，创建完后会自动去激活 Webhooks，不用再去 GitHub 设置，激活后可以在 Github 的 Webhooks 中看到配置生效

5. 一切搞定，从此只要你往这个仓库 push 代码，readthedoc 上面的文档就会自动更新


xiaojiongqian / writebysphinx
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Settings
Options
Collaborators
Branches
Webhooks
Notifications
Integrations & services
Deploy keys
Moderation
Interaction limits
Webhooks
Add webhook
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
https://readthedocs.org/api/v2/webhook/writebysphinx/95896/ (create, delete, pull_requ...)
Edit
Delete


Read the Docs
xiaojiongqian
项目 >
writebysphinx
阅读文档
概况
下载
搜索
构建
版本
管理
近期构建
构建版本:
master
通过 latest 版本 (html)
6 days, 7 hours 前
通过 master 版本 (html)
6 days, 7 hours 前
通过 latest 版本 (html)
1 week 前

4.2 在公司内部发布

基本流程是：用 Sphinx 生成文档，公司 Git 仓库管理文档版本，提交版本时通过 Jenkins 的机制在服务器端触发 Sphinx 编译，并将 build 文件同步到 Web 服务器。

- 需要在 Jenkins 上配置一个 pipeline 工程，在 git 提交新版本时触发脚本
- 编写那个脚本，让服务器自动运行 make html，并将 build 目录下的文件同步到 Web 服务器

CHAPTER 5

FAQ

5.1 目录设置问题

为了方便基于目录的版本管理，我们可以将文档源文件和分离，可以在使用 sphinx-quickstart 时设置 source 和 build 为不同目录。也可以后期在 Makefile(Mac, Linux) 或 make.bat(Windows) 文件进行设置。

例如，将：

```
SOURCEDIR     = .
BUILDDIR      = _build
```

改为

```
SOURCEDIR     = source/
BUILDDIR      = build/
```

提示： 于此同时，你应该将除 Makefile 文件和 _build 目录文件之外的其它所有源文件移到 source 目录中。另外，为了防止将 build 目录下的文件也同步到 git 上，你可以设置.gitignore 或/info/exclude 文件，来忽略上传这个目录的文件。

5.2 为什么有些目录没更新？

由于 sphinx 是增量编译的，对于没有变化的文件是不会去重新编译处理，如果只是更新了根目录下的 index.rst 文件内容，并且增加新的对应文件，可能会导致首页目录正常，有些老旧页面目录没有更新的情况。

处理办法：**只要删除已经 build 的文件目录，重新生成一遍即可**。

5.3 Read the Docs 部分设置注意点

总体而言，readthedocs 还是比较容易设置的，如果出错，也有明确的错误输出，根据其错误输出解决问题即可。比如本 sphinx 工程用到了 plantuml 画图工具，网站提示 plantuml 没有安装，那么我们只要新增加一个文本文件 requirements.txt，放在根目录即可。其中 requirements 文件内容为:

```
sphinxcontrib-plantuml
graphvi
```

然后在网站上高级设置中填上这个文件名，它就会自动去安装了，如下图。

另一个碰到的问题是编译警告出错，提示找不到 content.rst，原因是 readthedocs 默认认为初始文件是 content.rst，只要在 conf.py 文件中增加一行配置，告知系统初始文件名（不用带后缀）即可。

```
master_doc = 'index'
```

5.4 在 Sphinx 上使用 Markdown

虽然不建议用 Markdown 写大型文档，但目前有大量现存 Markdown 写的文档，可以支持将其移植进来。

Sphinx 支持用 Markdown 进行写作。

启用 Markdown 需要如下步骤:

安装 recommonmark

```
pip install --upgrade recommonmark
```

注解: 版本要求 recommonmark version 0.5.0 or later.

在 conf.py 文件增加配置:

```
extensions = ['recommonmark']
```

注解: 在 version 1.8 需要用到 source_parsers 变量:

```
source_parsers = {
'.md': 'recommonmark.parser.CommonMarkParser',
}
```

在 version 3.0 只要在扩展声明即可

如果你想在 *.md* 文件之外使用 *Markdown* 解析，比如.txt，用如下配置即可:

```
source_suffix = {
    '.rst': 'restructuredtext',
    '.txt': 'markdown',
    '.md': 'markdown',
}
```

更多的 Markdown 定制化配置，见 recommonmark documentation 。

CHAPTER 6

Indices and tables

- genindex
- search

索引