# Pelican 4

G:\SBT-MKDocs>pelican --help

usage: pelican [-h] [-t THEME] [-o OUTPUT] [-s SETTINGS] [-d] [-v] [-q] [-D] [--version] [-r]

[--print-settings [SETTING\_NAME [SETTING\_NAME ...]]] [--relative-urls] [--cache-path CACHE\_PATH]

[--ignore-cache] [-w SELECTED\_PATHS] [--fatal errors|warnings]

[--logs-dedup-min-level {DEBUG,INFO,WARNING,ERROR}] [-l] [-p PORT] [-b BIND]

[path]

A tool to generate a static blog, with restructured text input files.

positional arguments:

path Path where to find the content files. (default: None)

optional arguments:

-h, --help show this help message and exit

-t THEME, --theme-path THEME

Path where to find the theme templates. If not specified, it will use the default one included

with pelican. (default: None)

-o OUTPUT, --output OUTPUT

Where to output the generated files. If not specified, a directory will be created, named

"output" in the current path. (default: None)

-s SETTINGS, --settings SETTINGS

The settings of the application, this is automatically set to pelicanconf.py if a file exists

with this name. (default: None)

-d, --delete-output-directory

Delete the output directory. (default: None)

-v, --verbose Show all messages. (default: None)

-q, --quiet Show only critical errors. (default: None)

-D, --debug Show all messages, including debug messages. (default: None)

--version Print the pelican version and exit.

-r, --autoreload Relaunch pelican each time a modification occurs on the content files. (default: False)

--print-settings [SETTING\_NAME [SETTING\_NAME ...]]

Print current configuration settings and exit. Append one or more setting name arguments to

see the values for specific settings only. (default: None)

--relative-urls Use relative urls in output, useful for site development (default: False)

--cache-path CACHE\_PATH

Directory in which to store cache files. If not specified, defaults to "cache". (default:

None)

--ignore-cache Ignore content cache from previous runs by not loading cache files. (default: False)

-w SELECTED\_PATHS, --write-selected SELECTED\_PATHS

Comma separated list of selected paths to write (default: None)

--fatal errors|warnings

Exit the program with non-zero status if any errors/warnings encountered. (default: )

--logs-dedup-min-level {DEBUG,INFO,WARNING,ERROR}

Only enable log de-duplication for levels equal to or above the specified value (default:

WARNING)

-l, --listen Serve content files via HTTP and port 8000. (default: False)

-p PORT, --port PORT Port to serve HTTP files at. (default: 8000) (default: None)

-b BIND, --bind BIND IP to bind to when serving files via HTTP (default: 127.0.0.1) (default: None)

## Pelican简介

Pelican用Python编写的静态站点生成器。亮点：

* 使用reStructuredText或Markdown格式直接用编辑器编写内容
* 简单的CLI工具来(重新)生成站点
* 易于与分布式版本控制系统和Web对接
* 完全静态输出，可在任何地方托管

Pelican 4目前支持：

* 文章（例如博客文章）和页面（例如，“关于”，“项目”，“联系方式”）
* 评论，通过外部服务（Disqus）。如果您希望更好地控制评论，也可自托管评论是另一种选择。
* 主题支持（使用Jinja2模板创建主题）
* 以多种语言发表文章
* Atom / RSS feeds
* 代码高亮
* 从WordPress，Dotclear或RSS导入
* 与外部工具集成：Twitter，Google Analytics等（可选）
* 借助内容缓存和选择性输出写入，可以快速重建

为什么叫“Pelican”这个名字？

“Pelican”是calepin的字谜，在法语中意为“笔记本”。 ;）

您可以访问以下网址获取源代码：https：[//github.com/getpelican/pelican](https://github.com/getpelican/pelican" \t "_blank)。更多相关库参见[https://github.com/china-testing/python-api-tesing](https://github.com/china-testing/python-api-tesing" \t "_blank)。

搭建博客的工具选用了Pelican，Pelican是一个用Python语言编写的静态网站生成器，支持使用restructuredText和Markdown写文章，配置灵活，扩展性强，有许多优秀的主题和插件可供使用。Pelican 的Github地址是：<https://github.com/getpelican/pelican>;

## pelican 的功能

* 支持 reStructuredText、Markdown、AsciiDoc 编写格式
* 完全静态输出任何地方都很容易承载
* 可以通过 Jinja templates 自定义 Themes
* 多语言发布
* 支持 Atom/Rss feeds
* 代码语法高亮
* 支持从 WordPress、Dotclear 、Rss feeds 等服务中导入信息
* 模块化插件系统强大

## Installing Pelican

先根据xxx搭建python环境

vscode+pipenv

pip install pelican markdown

### 安装git,pip

|  |  |
| --- | --- |
| 1  2 | emerge --ask dev-cvs/git  emerge --ask dev-python/pip |

### 使用virtualenv工具创建pelican虚拟环境

|  |  |
| --- | --- |
| 1  2  3  4 | emerge --ask virtualenv ## or pip install virtualenv  virtualenv ~/virtualenvs/pelican  cd ~/virtualenvs/pelican  source bin/activate |

### 安装pelican

|  |  |
| --- | --- |
| 1 | pip install pelican |

### 安装markdown,typogrify

|  |  |
| --- | --- |
| 1  2 | pip install Markdown  pip install typogrify |

## 更新pelican

pipenv install --upgrade pelican

## Create a project

**环境安装和初始化**

pipenv install pelican markdown

运行初始化pipenv run pelican-quickstart，它会问些关于站点的基本问题。

Welcome to pelican-quickstart v3.7.1.   
This script will help you create a new Pelican-based website.   
Please answer the following questions so this script can generate the files   
needed by Pelican.   
> Where do you want to create your new web site? [.]   
> What will be the title of this web site?   
> Who will be the author of this web site?   
> What will be the default language of this web site?   
> Do you want to specify a URL prefix? e.g., <http://example.com> (Y/n)   
> Do you want to enable article pagination? (Y/n)   
and balabalabala……

blog/

├── content # 存放输入的markdown或RST源文件

│ └── (pages) # 存放手工创建的静态页面，可选

├── output # 存放最终生成的静态博客

├── develop\_server.sh # 测试服务器

├── Makefile # 管理博客的Makefile

├── pelicanconf.py # 配置文件

└── publishconf.py # 发布文件，可删除

> Where do you want to create your new web site? [.] # 默认

> What will be the title of this web site? myblog # 输入博客名称，例如：myblog

> Who will be the author of this web site? obama # 输入博客作者

> What will be the default language of this web site? [Chinese (Simplified)] # 默认中文

> Do you want to specify a URL prefix? e.g., https://example.com (Y/n) # Y ，如果打算放在github上，可以选择 n

> What is your URL prefix? (see above example; no trailing slash) # 输入githup地址，例如：https://myblog.github.io

> Do you want to enable article pagination? (Y/n) # 默认

> How many articles per page do you want? [10] # 默认

> What is your time zone? [Europe/Paris] # Asia/Shanghai

> Do you want to generate a tasks.py/Makefile to automate generation and publishing? (Y/n) # 默认

> Do you want to upload your website using FTP? (y/N) # N

> Do you want to upload your website using SSH? (y/N) # N

> Do you want to upload your website using Dropbox? (y/N) # N

> Do you want to upload your website using S3? (y/N) # N

> Do you want to upload your website using Rackspace Cloud Files? (y/N) # N

> Do you want to upload your website using GitHub Pages? (y/N) # 如果是用 github，默认

Done. Your new project is available at F:\Blog\pelican\_blog

新建一个文件夹，

mkdir -p ~/projects/yoursite

cd ~/projects/yoursite

|  |  |
| --- | --- |
| 1  2  3 | mkdir blog //创建文件夹，名称可根据自己喜欢定  cd blog  pelican-quickstart  mkdir -p ~/projects/yoursite  cd ~/projects/yoursite  pelican-quickstart |

pelican-quickstart执行命令后，会提示输入博客的配置项，除了少数几个必填以外，其它都可以选择默认，而且都可以在pelicanconf.py文件中进行更改，所以你可以随意选择。 命令成功执行后，会出现pelican的框架，如下所示

## pelican站点目录组成

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | blog/  ├── content # 存放输入的markdown或RST源文件  │ └── (pages) # 存放手工创建的静态页面，可选  │ └── (posts) # 存放手工创建的文章，可选  ├── output # 存放最终生成的静态博客  ├── develop\_server.sh # 测试服务器  ├── Makefile # 管理博客的Makefile  ├── pelicanconf.py # 配置文件  └── publishconf.py # 发布文件，可删除 |

## Writing content

### Articles and pages 文章和页面

Pelican 认为 “文章(articles)” 是按时间顺序排列的内容, 如博客上的帖子, 因此与日期相关。

“页面(pages)” 背后的想法是, 它们通常不是时态性质的, 用于不经常更改的内容 (例如, “关于” 或 “联系” 页面)。

你可以在官方库中找到一些例子在https://github.com/getpelican/pelican/tree/master/samples/content

创建文章

~/projects/yoursite/content/keyboard-review.md

Title: My First Review

Date: 2010-12-03 10:20

Category: Review

Following is a review of my favorite mechanical keyboard.

**创建第一篇文章**

在创建文章之前，Pelican是不能运行的。在yoursite/content/目录下添加第一篇markdown。

Title: 我的第一篇文章

Date: 2018-01-25 11:37

Category: 日记

胡乱写点什么试试

源文件顶部Title,Date,Category等内容则是必须的，具体可参见[文档](http://docs.getpelican.com/en/3.3.0/getting_started.html#writing-content-using-pelican)，它们各自意义如下

Title: 文章标题

Date: 创建日期

Modified: 修改日期

Category: 文章分类，标志本文处于该分类下

Tags: 文章标签，标志本文处于该标签下

Slug: URL中该文章的链接地址

Author: 作者

### File metadata 文件中的信息

Pelican 尽可能的自己从文件系统得到信息（如，关于文章的类别，当不指定时pelican会使用文件夹作为分类依据），但是你仍需要在文件中提供一些信息(metadata)。

完成上述博客主体搭建后，使用markdownpad创建一个.md文件，保存于content文件夹中（或自己加入posts文件夹）。 Metadata syntax for Markdown posts should follow this pattern:

|  |  |
| --- | --- |
| 12345678910 | Title: My super titleDate: 2010-12-03 10:20Modified: 2010-12-05 19:30Category: PythonTags: pelican, publishingSlug: my-super-postAuthors: Alexis Metaireau, Conan DoyleSummary: Short version for index and feedsThis is the content of my super blog post. |

如果您以 reStructuredText 格式编写内容, 则可以通过以下语法在文本文件中提供此信息 (以 . rst后缀名的文件):

My super title

:date: 2010-10-03 10:20   
:modified: 2010-10-04 18:40   
:tags: thats, awesome   
:category: yeah   
:slug: my-super-post   
:authors: Alexis Metaireau, Conan Doyle   
:summary: Short version for index and feeds

authors 和 tags 可以用分号（;）或逗号 (,) 分隔。

:tags: pelican, publishing tool; pelican, bird   
:authors: Metaireau, Alexis; Doyle, Conan

Pelican 通过abbr标记实现了对 reStructuredText 的扩展, 以启用对 HTML 的支持。你可以在你的推送(post)中像这样写:

This will be turned into :abbr:HTML (HyperText Markup Language).

你同样可以使用 Markdown 语法 （后缀名为 .md，.markdown，.mkd或.mdown）。Markdown生成器要求你先安装Markdown包，请尝试运行pip install Markdown命令。

Pelican 同时支持扩展的Markdown语法 (Markdown Extensions)，如果扩展语法没有包含在Markdown包中，可能需要另行安装。

Markdown 的文件需要像下面一样添加信息(metadata)：

Title: My super title

Date: 2010-12-03 10:20

Modified: 2010-12-05 19:30

Category: Python

Tags: pelican, publishing

Slug: my-super-post

Authors: Alexis Metaireau, Conan Doyle

Summary: Short version for index and feeds

This is the content of my super blog post.

| **Metadata** | **描述** | **备注** |
| --- | --- | --- |
| title | 文章或页面的标题 |  |
| date | 出版日期(例如，YYYY-MM-DD HH:SS) |  |
| modified | 修改日期(例如，YYYY-MM-DD HH:SS) |  |
| tags | 内容标签，用逗号分隔 |  |
| keywords | 内容关键字，用逗号分隔(仅HTML内容) |  |
| category | 内容类别(只有一个-不是多个) |  |
| slug | URL和翻译中使用的标识符 | 文件名（不含扩展名） |
| author | 内容作者，当只有一个 |  |
| authors | 内容作者，当有多个 |  |
| summary | 索引页内容简介 |  |
| lang | 内容语言ID(en, fr等等) |  |
| translation | 是另一个(true或false) |  |
| status | 内容状况：draft, hidden，或published |  |
| template | 用于生成内容的模板名称(没有扩展) |  |
| save\_as | 将内容保存到此相对文件路径 |  |
| url | 用于本文/页的URL |  |

其他格式（如 AsciiDoc）需要通过插件支持，请参与pelican-plugins的相关说明。

Pelican同样可以处理.html和.htm结尾的HTML文件。它以非常直接的方式读取信息，title从title，body从body：

<html>

<head>

<title>My super title</title>

<meta name="tags" content="thats, awesome" />

<meta name="date" content="2012-07-09 22:28" />

<meta name="modified" content="2012-07-10 20:14" />

<meta name="category" content="yeah" />

<meta name="authors" content="Alexis Métaireau, Conan Doyle" />

<meta name="summary" content="Short version for index and feeds" />

</head>

<body>

This is the content of my super blog post.

</body>

</html>

使用 html, 标准信息有一个简单的做法: tags可以按Pelican标准要求的通过tags标记指定, 也可以按HTML要求的通过keywords提供信息。两者可以互换使用。

请注意, 除了title之外, 本文的其他信息(metadata)都不是必需提供的: 如果未指定日期, 并且 DEFAULT\_DATE 设置为 'fs', 则将依赖于文件的 “mtime” 时间戳, 并且该类别可以由文件所在的目录确定。例如, 位于 python/foobar/myfoobar. rst 的文件将有一个foobar类别。如果您想以其他方式组织文件, 而该文件夹的名称不是一个好的类别名称, 则可以将设置 USE\_FOLDER\_AS\_CATEGORY 设置为 False。在分析页面信息给定的日期时, Pelican支持 W3C’s 建议的[ISO 8601](http://www.w3.org/TR/NOTE-datetime)。

**注意：**   
在尝试使用不同的设置 (特别是信息) 时, 缓存可能会发生干扰，使更改可能不可见。在这种情况下, 禁用缓存 LOAD\_CONTENT\_CACHE = False 或使用--ignore-cache命令行开关忽略缓存。

modified应是上次更新文章的时间, 如果未指定, 则默认为date。除了可以在模板中显示modified外, 当您修改文章后将midified设置为当前日期时, feed readers 中的提要条目将自动更新。

authors是一个逗号分隔的文章作者列表。如果只有一个作者可以使用author。

如果你没有在 post 中指定摘要信息, 则可以使用 SUMMARY\_MAX\_LENGTH 设置来指定从文章开头的多少单词用作摘要。

还可以通过在 FILENAME\_METADATA 设置中设置的正则表达式从文件名中提取信息。所有匹配的命名组将在信息对象中设置。FILENAME\_METADATA 设置的默认值仅从文件名中提取日期。例如, 如果您想提取日期和slug, 可以设置如下内容:'(?P<date>\d{4}-\d{2}-\d{2})\_(?P<slug>.\*)'

请注意, 文件中可用的信息优先于从文件名中提取的信息。

### Pages 页面

如果你在 content 文件夹下建立了一个叫pages的文件夹，所有在这个文件夹内的文件都会用来生成静态页面 ( static pages)，像 About 和 Contact 页面等。

你可以使用设置 DISPLAY\_PAGES\_ON\_MENU 来控制是否所有这些页面都显示在主导航菜单中。(默认值为 True。)

如果要关闭某个页面在菜单中链接或列出, 请添加status: hidden属性到其信息中。这对于制作适合网站主题的错误页面(error pages)非常有用。

### Linking to internal content 内部链接

从 Pelican 3.1 开始, 现在可以指定站点内部链接 ( intra-site ) 到**源内容层次 (source content hierarchy)**结构中的文件, 而不是**生成的层次 (generated hierarchy)**结构中的文件。这使得从当前帖子 ( current post)链接到可能与该帖子并排的其他内容更容易 (而不必确定在站点生成之后其他内容的将放置位置)。

要链接到内部内容 (content目录中的文件), 请对链接目标使用以下语法: {filename}path/to/file，注: 正斜线（/）是所有操作系统 (包括 Windows) 的 {filename} 指令中的路径分隔符。

举例如下，加入有一个 Pelican 项目的结构像下面这样：

website/

├── content

│ ├── category/

│ │ └── article1.rst

│ ├── article2.md

│ └── pages

│ └── about.md

└── pelican.conf.py

其中article1.rst可能是这样：

The first article

#################

:date: 2012-12-01 10:02

See below intra-site link examples in reStructuredText format.

`a link relative to the current file <{filename}../article2.md>`\_

`a link relative to the content root <{filename}/article2.md>`\_

article2.md如下：

Title: The second article

Date: 2012-12-01 10:02

See below intra-site link examples in Markdown format.

[a link relative to the current file]({filename}category/article1.rst)

[a link relative to the content root]({filename}/category/article1.rst)

#### Linking to static files 链接到静态文件

像上文提到的使用{filename}标识来连接到既非 aritcle 也非 pages 的内容。必须记住, 这些文件不会被复制到output目录中, 除非包含它们的源目录包含在项目的 pelicanconf.py 文件的 STATIC\_PATHS 设置中。Pelican 的配置默认包括images目录，但必须手动添加其他内容。忘记这样做会导致链接断开。

举例来说，一个项目又如下目录结构：

content

├── images

│ └── han.jpg

├── pdfs

│ └── menu.pdf

└── pages

└── test.md

test.md中的链接：

![Alt Text]({filename}/images/han.jpg)

[Our Menu]({filename}/pdfs/menu.pdf)

pelicanconf.py文件设置如下：

STATIC\_PATHS = ['images', 'pdfs']

网站生成的时候就会复制han.jpg到output/image/han.jpg，menu.pdf到output/pdfs/menu.pdf，并且添加正确的链接到test.md。

#### Mixed content in the same directory 同一文件夹下的混合内容

从 Pelican 3.5 开始, 静态文件可以安全地与页面源文件共享源目录, 而不会在生成的站点中公开页面源。任何此类目录都必须同时添加到 STATIC\_PATHS 和 PAGE\_PATHS (或 STATIC\_PATHS 和 ARTICLE\_PATHS)。Pelican 通常会识别和处理页面源文件, 并将其余的文件复制到为静态文件保留的单独目录中。

Note: 将静态和内容源文件放在同一个源目录中并不能保证它们最终会在生成的站点中的同一位置。为确保如此的最简单方法是使用 {attach} 链接语法 (参见下一节)。或者, 可以将 STATIC\_SAVE\_AS、PAGE\_SAVE\_AS 和 ARTICLE\_SAVE\_AS 属性 (以及相应的 \* \_URL 属性) 设置为将不同类型的文件放在一起, 就像在早期版本的 Pelican 中一样。

#### Attaching static files 附件

从 Pelican 3.5 开始, 静态文件(static files)可以使用{attach}path/to/file将目标 “附加” 到一个网页或文章上，这类似于 {filename} 语法, 但也将静态文件重新定位到链接文档的output目录中。如果静态文件源自链接文档源下的子目录, 则该关系将保留在输出上。否则, 它将成为链接文档的同级。

这只适用于链接到静态文件, 而且仅当它们源自于 STATIC\_PATHS 属性中包含的目录时。

举个栗子，如下的目录结构：

content

├── blog

│ ├── icons

│ │ └── icon.png

│ ├── photo.jpg

│ └── testpost.md

└── downloads

└── archive.zip

pelicanconf.py设置如下：

PATH = 'content'

STATIC\_PATHS = ['blog', 'downloads']

ARTICLE\_PATHS = ['blog']

ARTICLE\_SAVE\_AS = '{date:%Y}/{slug}.html'

ARTICLE\_URL = '{date:%Y}/{slug}.html'

testpost.md内容如下：

Title: Test Post

Category: test

Date: 2014-10-31

![Icon]({attach}icons/icon.png)

![Photo]({attach}photo.jpg)

[Downloadable File]({attach}/downloads/archive.zip)

生成器生成的output目录结构如下：

output

└── 2014

├── archive.zip

├── icons

│ └── icon.png

├── photo.jpg

└── test-post.html

请注意, 所有使用 {attach} 链接的文件最终都在文章的输出目录中或下面。

如果静态文件 ( static file )被多次链接, 则 {attach} 的重新定位功能将只在要处理的第一个链接中起作用。在第一个链接之后, Pelican将像 {filename} 一样处理 {attach}。这样可避免中断已处理的链接。

**从多个文档链接到文件时要小心**： 由于 Pelican 没有定义处理文档的顺序，第一个指向文件的链接确定了其最终位置，因此在由多个文档链接的文件上使用 {attach}时会导致在不同的版本生成时文件位置发生更改（ cause its location to change from one site build to the next）。(这是否在实际中发生受操作系统、文件系统、Pelican 版本以及从项目中添加、修改或删除的文档的影响。）任何链接到文件旧位置的外部站点都可能会发现它们的链接已失效。**因此, 仅当在链接的文档在同一个目录下时，才建议使{attach}，并且要同时全部使用{attach}。It is therefore advisable to use {attach} only if you use it in all links to a file, and only if the linking documents share a single directory.** 在这样的情况下, 文件的输出位置在将来的生成中不会更改。在无法采取这些预防措施的情况下, 请考虑使用 {filename} 链接而不是 {attach} , 并让该文件的位置由项目的 STATIC\_SAVE\_AS 和 STATIC\_URL 设置确定。(每个文件 save\_as 和 url 仍然可以在 EXTRA\_PATH\_METADATA 中设置重写。) 此段落翻译不好，大家将就看下

#### Linking to authors, categories, index and tags

你可以使用{author}name连接到作者；{category}foobar连接到类别；{index}到索引；{tag}tagname到标签。

#### Deprecated internal link syntax 快要过时的内部链接语法

为了与早期版本保持兼容, Pelican 仍然支持垂直条 (| |)的内部链接。例如: | filename | an\_article.rst， | tag | tagname |， |category | foobar。语法从 | |到 {} 的改变是为了避免与扩展 Markdown 或 reST 的指令冲突。可能最终会删除对旧语法的支持。

### Importing an existing site 从已存在的站点导入

可以用一个简单的脚本实现从 WordPress、Tumblr、Dotclear 和Rss feeds的导入，详见[站点导入](http://docs.getpelican.com/en/stable/importer.html" \l "import)

### Translations 翻译

提供多语种的文章是可行的，为了做到这些，你需要在文章或是页面中添加lang信息，并设置DEFAULT\_LANG属性（默认英语[en])。有了这些设置, 将只列出具有默认语言的文章, 并且每篇文章都附有该文章的可用翻译列表。

Note：   
Pelican 的核心功能不创建每个语言的翻译模板的子站点 (例如 example.com/de)。对于这样的高级功能, 可以使用 i18n\_subsites 插件。

Pelican 使用文章的 URL “slug” 来确定两个或更多的文章是否是互相的翻译。可以在文件的信息(metadata)中手动设置slug;如果没有明确设置, Pelican 将通过标题自动生成slug。

下面给个例子，这里有两篇文章，一个英语一个法语。

英语文章：

Foobar is not dead

##################

:slug: foobar-is-not-dead

:lang: en

That's true, foobar is still alive!

法语版：

Foobar n'est pas mort !

#######################

:slug: foobar-is-not-dead

:lang: fr

Oui oui, foobar est toujours vivant !

发布内容后, 您可以看到这两篇文章中唯一的共同点就是slug, 它在这里是一个标识符。如果您不希望以这种方式显式定义该slug, 则必须确保翻译的文章标题相同, 因为将从文章标题自动生成slug。

如果不希望 DEFAULT\_LANG 检测到某个特定文章的原始版本, 请在信息（metadata）中使用translation指定哪些帖子是翻译的版本：

Foobar is not dead

##################

:slug: foobar-is-not-dead

:lang: en

:translation: true

That's true, foobar is still alive!

### Syntax highlighting 语法高亮

Pelican 可以为您的代码块提供彩色的语法高亮。为此, 您必须在内容文件中使用以下约定。

对于 reStructuredText, 请使用code-block指定要突出显示的代码类型 (在此示例中, 我们将使用 python)：

.. code-block:: python

print("Pelican is a static site generator.")

对于使用 CodeHilite 扩展来提供语法高亮的Markdown, 请在代码块的正上方包含语言标识符, 同时缩进标识符和代码：

There are two ways to specify the identifier:

:::python

print("The triple-colon syntax will \*not\* show line numbers.")

To display line numbers, use a path-less shebang instead of colons:

#!python

print("The path-less shebang syntax \*will\* show line numbers.")

指定的语言标识符 (如 python、ruby) 应该出现在[list of available lexers](http://pygments.org/docs/lexers/)中。

使用 reStructuredText 时, 代码块指令 (code-block directive) 中提供了以下选项:

| **选项** | **有效值** | **描述** |
| --- | --- | --- |
| anchorlinenos | N/A | 如果在<a>标记中显示行号 |
| classprefix | String | 标记类名 |
| hl\_lines | numbers | 要突出显示的行的list, 其中要突出显示的行号由空格分隔。这类似于在Sphinx中的emphasize-lines, 但它不支持用连字符分隔的行号范围, 或逗号分隔的行号。 |
| lineanchors | string | 在lineanchors和-linenumber之间对其中的每一行进行包装。 |
| ………太多了了 | 去看原文吧 | <http://docs.getpelican.com/en/stable/content.html#syntax-highlighting> |

Note：不同的版本中, 您的 Pygments 模块可能没有所有可用的选项。有关每个选项的详细信息, 请参阅 [Pygments 文档](http://pygments.org/docs/formatters/)的 HtmlFormatter 部分。

例如, 下面的代码块启用行号, 从153开始, 并使用 pgcss 前缀 Pygments css 类, 以使名称更加独特, 并避免可能的 css 冲突:

.. code-block:: identifier

:classprefix: pgcss

:linenos: table

:linenostart: 153

<indented code block goes here>

### Publishing drafts 草稿

如果要将文章发布为草稿 (例如, 在发布前供好友审阅)，可以将Status:draft 属性添加到其信息（metadata) 中。然后, 该文章将输出到 draft 文件夹, 而不在索引页上或任何类别或标记页上列出。

如果您想要所有文章自动作为草稿发布 (在文章完成前不会意外地发布), 请在 DEFAULT\_METADATA 中包括以下状态:

DEFAULT\_METADATA = {

'status': 'draft',

}

若要在以上情况，即设置了默认状态为 “草稿” 时发布帖子, 请更新帖子的信息以包括Status: published。

### 学习小结

由于对rst格式并不熟悉，重点还是看了看怎么使用目录结构以及markdown。   
文章中应该包括的信息有：

| **标记** | **内容** |
| --- | --- |
| Title | 标题 |
| Date | yyyy-mm-dd hh:mm |
| Modified | 修改时间，格式同上 |
| Tags | 标签 |
| Slug | Slug用来指定文件名 |
| Authors / Author | 作者，作者 |
| Summary | 概要 |

链接是个重要的东西，会影响build以后的目录结构。让我欣喜的是居然还能够提供附件，这个比我预期要好很多。   
多篇文章指向同一个文件那里没有看懂怎么回事，只能待将来实操多实验几次来总结了。

## Publish Your Site

### Site generation 网站生成

一旦安装了 Pelican , 并且您有一些内容 (例如, 以Markdown 或 reST 格式), 您可以通过 Pelican 命令将内容转换为 HTML, 并指定内容的路径和设置文件的路径 (可选):

pelican /path/to/your/content/ [ -s path/to/your/settings.py]

上面的命令将使用默认的主题生成一个简单的站点，并将其保存在output/文件夹中。默认主题由非常简单的 HTML 组成, 没有样式, 因此人们可以使用它作为创建他们自己的主题的基础。

在处理单个项目或页面时, 使用 --write-selected可以只生成与该内容对应的文件。如下:

pelican --write-selected output/posts/my-post-title.html

NOTE：必须指定生成的输出文件的路径, 而不是源内容。要确定输出文件路径, 请使用 --debug标志来确定正确的文件名和位置。如果需要, --write-selected 可以采取逗号分隔的路径列表, 也可以配置为设置。(请参见: [Writing only selected content](http://docs.getpelican.com/en/stable/settings.html#writing-only-selected-content))

您还可以运行 Pelican时添加使用 -r 或 --autoreload 选项，告诉 Pelican 监视您的修改, 而不是在每次希望看到更改时手动重新运行它。

Pelican 有其他可用的命令行开关。请查看 “帮助” 以查看可以使用的所有选项:

pelican --help

生成网站

pelican content

[Publish your site](http://docs.getpelican.com/en/3.5.0/publish.html#publish-your-site) 有三种方法可以发布博客： - pelican命令 - make 详细用法查看Makefile文件 - fabric

**生成页面**

pipenv run pelican content

执行make html命令进行博客的生成

make html

(pelican e:/blog/content/ -o e:/blog/output -s e:/blog/pelicanconf.py)

Title等信息一定要在第一行开始，不然会提示找不到。

[Site generation](http://docs.getpelican.com/en/3.5.0/publish.html" \l "site-generation)

|  |  |
| --- | --- |
| 1 | pelican /path/to/your/content/ [-s path/to/your/settings.py] |

或

|  |  |
| --- | --- |
| 1 | make html |

生成的站点放在output/目录下。

## Viewing the generated files 预览

Pelican 生成的是静态页面，你并不需要什么特殊的软件来预览。你可以直接用浏览器来打开页面查看：

firefox output/index.html

浏览器不能定位CSS，所以上面的方法会让你困惑。你可以使用 Python 运行一个简单的 web 服务，当使用 Python 3时：

cd output

python -m http.server

一旦服务开始运行，你可以在 http://localhost:8000/预览你生成的静态站。

打开一个新的 terminal ，切换目录到output，然后开启 Pelican 的服务器：

cd /x..xx/yoursite/output

pipenv run python -m pelican.server

make html命令将把刚才写的博文生成html，存放到output目录下，如果你没有make命令，也可执行第二行的pelican命令。接着执行make serve开启测试服务器

make serve

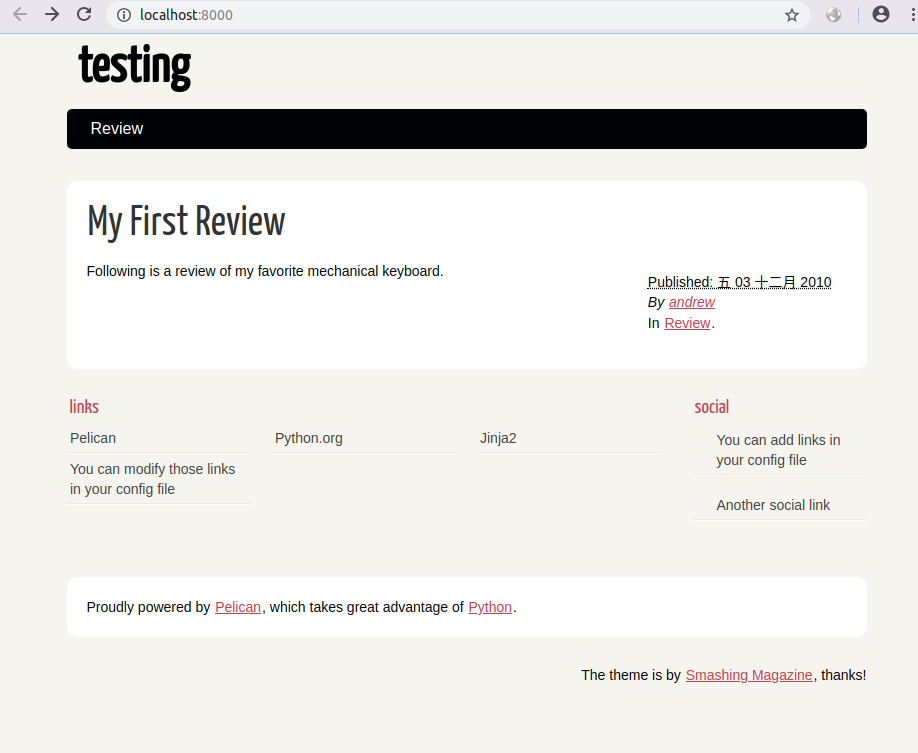
(cd e:/blog/output/ && python -m pelican.server)

make serve命令也可由第二行替代，在浏览器中输入http://localhost:8000即可看到博文效果。

浏览http://localhost:8000/页面预览以下。

pelican --listen

访问：[http://localhost:8000/](http://localhost:8000/" \t "_blank)。



图片.png

For Python 2, run:

|  |  |
| --- | --- |
| 1  2 | cd output  python -m SimpleHTTPServer |

For Python 3, run:

|  |  |
| --- | --- |
| 1  2 | cd output  python -m http.server |

或

|  |  |
| --- | --- |
| 1 | make serve |

浏览<http://localhost:8000/>地址预览效果.

## Deployment 部署

生成站点后, 在本地开发环境中预览它, 并准备将其部署到生产中, 您可能首先使用您定义的任何特定于生产的设置 (例如, 分析提要等) 重新生成站点:

pelican content -s publishconf.py

若要以 pelicanconf.py 作为发布配置的基础, 您可以通过在publishconf.py中导入 pelicanconf 设置:

from pelicanconf import \*

* 1

如果你是通过运行 pelican-quickstart生成的 publishconf.py，以上以默认存在于文件中。

部署站点的步骤将取决于它将部署在何处。如果您对运行 Nginx 或 Apache 的服务器具有 SSH 访问权限, 则可以使用 rsync 工具来传输您的站点文件：

rsync -avc --delete output/ host.example.com:/var/www/your-site/

* 1

还有许多其他部署选项, 其中一些可以在首次通过 pelican-quickstart 命令设置站点时进行配置。有关通过 GitHub 页面发布的详细信息, 请参阅[提示页面](http://docs.getpelican.com/en/stable/tips.html)。

[Deployment](http://docs.getpelican.com/en/3.5.0/publish.html" \l "deployment) 原理：将output目录下生成的站点部署到自己的github pages.

## github部署

比如<https://github.com/china-testing/china-testing.github.io>，这个仓库名要求为[username.github.io](http://username.github.io" \t "_blank)。

上传代码，就可以访问你的网站了。

#!/usr/bin/env bash

DATE=$(date +%Y%m%d)

cd /home/andrew/code/yoursite

rm -rf output/\*

pelican content

cd /home/andrew/code/china-testing.github.io

git rm -rf \*

cp -rf /home/andrew/code/yoursite/output/\* .

git add \*

git commit -m $DATE

git push

网站示例：　[https://china-testing.github.io/](https://china-testing.github.io/" \t "_blank)

## Automation 自动化

虽然 pelian 命令是生成站点的规范化方法, 但自动化工具可以用来简化生成和发布流程。在 pelican-quickstart 过程中询问的问题之一是您是否希望自动生成和发布站点。如果您对该问题回答 “yes”, 则会在项目的根目录中生成 fabfile.py 和 Makefile 文件。这些文件预先填写了从 pelican-quickstart 过程中提供的其他答案收集的某些信息, 这意味着作为一个起点, 应自定义以适合您的特定需求和使用模式。如果您发现这些自动化工具作用有限, 这些文件可以随时删除, 不会影响 pelican 的使用。

下面是 “wrap” pelican 的自动化工具, 可以简化生成、预览和上传站点的过程。

### Fabric

[Fabric](http://fabfile.org/) 的优点是它是用 Python 编写的, 因此可以在多种环境中使用。缺点是必须单独安装。使用下面的命令来安装 Fabric , 您有可能需要用到 “sudo”：

pip install Fabric

* 1

Note: **Windows 需要安装 PyCrypto**   
  
Fabric 依赖 PyCrypto，如果您的系统没有 C 编译器, 则很难安装。对于 Windows 用户, 在安装Fabric之前, 请使用   
easy\_install http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-py2.7.exe   
根据[此 StackOverflow 建议](http://stackoverflow.com/a/11405769/6364)您使用 Python 2.7 的 Win32 版本和PyCrypto比64位版本的Win64—even更容易成功。

请花些时间打开在项目根目录中生成的 fabfile.py 文件。您将看到许多命令, 其中每一个都可以根据您的需要来重命名、删除和/或自定义。使用 out-of-the-box 配置, 您可以通过以下方式生成站点：

fab build

* 1

如果您希望 Pelican 在每次检测到更改时自动重新生成站点 (这在本地测试时很方便), 请改用以下命令:

fab regenerate

* 1

运行本地服务以供 <http://localhost:8000/>预览:

fab serve

* 1

如果在 pelican-quickstart中, 当被问及是否要通过 ssh 上传站点时回答 “是”, 可以使用以下命令通过 ssh 通过 rsync 发布站点:

fab publish

* 1

这些只是一些默认情况下可用的命令, 所以请随时浏览 fabfile.py。更重要的是, 不要犹豫自定义 fabfile.py , 以满足您的特定需求和喜好。

### Make

当您在 pelican-quickstart对相关问题说 “是” 时, 也会自动为您创建 Makefile。这种方法的优点是, 使命令被内置到大多数 POSIX 系统中, 因此不需要安装其他任何东西来使用它。缺点是, 非 POSIX 系统 (例如, Windows) 不包括 make, 并且在这些系统上安装它可能是一个艰巨的任务。

如果你想使用 make根据pelicanconf.py生成你的站：

make html

* 1

利用publishconf.py里的设置将生成站点布置到生产环境：

make publish

* 1

如果你希望当文件更改后不用手动生成，而是 Pelican 自动监控处理，可以：

make regenerate

* 1

提供 <http://localhost:8000> 预览服务：

make serve

* 1

通常情况下, 您需要在两个单独的终端中运行 make regenerate 和 make serve , 但您可以通过以下方式同时运行两个：

make devserver

* 1

上述命令将同时运行 Pelican 的再生模式, 以及输出在 <http://localhost:8000> 的服务。完成对更改的测试后, 应通过以下方式停止开发服务器：

./develop\_server.sh stop

当您准备发布网站时, 您可以通过在 pelican-quickstart选择的方法将其上传。对于本例, 我们将在 ssh 上使用 rsync:

make rsync\_upload

（默认的 Makefiel 和 devserver.sh 使用 python 和 pelican 来完成其任务。如果要使用不同的可执行文件 (如 python3), 则可以分别设置 py 和 PELICAN 环境变量来重写默认的可执行文件名称。）

* Fabric

|  |  |
| --- | --- |
| 1  2  3  4  5 | pip install Fabric  fab build  fab regenerate  fab serve  fab publish |

* Make

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | make html  make regenerate  make serve  make devserver  make stopserver  ./develop\_server.sh stop |

备注：发布的简单流程：pelican content生成页面至output目录，然后git push将output目录推送到github站点的gh-pages分支，即可自动渲生成染博客。

参考资料： <http://www.xycoding.com/articles/2013/11/21/blog-create/>

## Pelican Plugins

Beginning with version 3.0, Pelican supports plugins. Plugins are a way to add features to Pelican without having to directly modify the Pelican core. Starting with 3.2, all plugins (including the ones previously in the core) have been moved to this repository, so this is the central place for all plugins.

### How to use plugins

The easiest way to install and use these plugins is to clone this repo:

git clone --recursive https://github.com/getpelican/pelican-plugins

and activate the ones you want in your settings file:

PLUGIN\_PATHS = ['path/to/pelican-plugins']

PLUGINS = ['assets', 'sitemap', 'gravatar']

PLUGIN\_PATHS can be a path relative to your settings file or an absolute path.

Alternatively, if plugins are in an importable path, you can omit PLUGIN\_PATHS and list them:

PLUGINS = ['assets', 'sitemap', 'gravatar']

or you can import the plugin directly and give that:

import my\_plugin

PLUGINS = [my\_plugin, 'assets']

### Plugin descriptions

添加评论系统

开启个人博客的原因在于分享知识，分享就需要交流，评论模块当然少不了。大多数主题默认支持Disqus。在[Disqus](https://disqus.com/)上申请帐号，按照流程Disqus会分配给你站点的Shortname，记牢Shortname，如果忘了请进入admin/settings中查看。然后同理，在pelicanconf.py添加 国内也可以选择[多说](http://duoshuo.com/)和[友言](http://www.uyan.cc/).

|  |  |
| --- | --- |
| 1 | DISQUS\_SITENAME = Shortname |

## 配置文件设置 重点

# all the following settings are \*optional\*

# HTML metadata

SITEDESCRIPTION = ''

# all defaults to True.

DISPLAY\_HEADER = True

DISPLAY\_FOOTER = True

DISPLAY\_HOME = True

DISPLAY\_MENU = True

# provided as examples, they make 'clean' urls. used by MENU\_INTERNAL\_PAGES.

TAGS\_URL = 'tags'

TAGS\_SAVE\_AS = 'tags/index.html'

AUTHORS\_URL = 'authors'

AUTHORS\_SAVE\_AS = 'authors/index.html'

CATEGORIES\_URL = 'categories'

CATEGORIES\_SAVE\_AS = 'categories/index.html'

ARCHIVES\_URL = 'archives'

ARCHIVES\_SAVE\_AS = 'archives/index.html'

# use those if you want pelican standard pages to appear in your menu

MENU\_INTERNAL\_PAGES = (

('Tags', TAGS\_URL, TAGS\_SAVE\_AS),

('Authors', AUTHORS\_URL, AUTHORS\_SAVE\_AS),

('Categories', CATEGORIES\_URL, CATEGORIES\_SAVE\_AS),

('Archives', ARCHIVES\_URL, ARCHIVES\_SAVE\_AS),

)

# additional menu items

MENUITEMS = (

('GitHub', 'https://github.com/'),

('Linux Kernel', 'https://www.kernel.org/'),

)

## pelican-themes

回到blog目录下，按如下步骤下载pelican官方主题，从里面挑选出自己喜欢的主题吧，大多数主题预览界面你可以打开这个网页[主题预览](http://pelicanthemes.com/)进行查看。不过如今pelican又新出了很多主题，所以你需看看[pelican主题开源库](https://github.com/getpelican/pelican-themes)。

* 克隆主题到本地

|  |  |
| --- | --- |
| 1 | git clone https://github.com/getpelican/pelican-themes.git |

* 使用主题 打开pelicanconf.py配置文件，更改或添加THEME为自己喜欢的主题，例如本博客所挑选的elegant，更多的配置含义请关注[官方文档](http://docs.getpelican.com/en/3.5.0/index.html)。

|  |  |
| --- | --- |
| 1 | THEME = 'pelican-themes/pelican-elegant' |

### pelican-themes 插入主题

### Description 描述

pelican-themes是 Pelican 用来管理主题的命令行工具。

#### Usage 用法

pelican-themes [-h] [-l] [-i theme path [theme path ...]]

[-r theme name [theme name ...]]

[-s theme path [theme path ...]] [-v] [-version]

* 1
* 2
* 3

#### Optional arguments 参数

| **参数** | **说明** |
| --- | --- |
| -h, –help | 显示帮助 |
| -l, –list | 列出已经安装的主题 |
| -i theme\_path, –install theme\_path | 安装主题（可多个同时安） |
| -r theme\_path, –remove theme\_path | 删除主题 |
| -s theme\_path, –symlink theme\_path | 以符号链接形式安装主题，而非复制，常用于主题开发 |
| -v, –verbose | 详细输出 |
| –version | 版本 |
| git clone https://github.com/getpelican/pelican-themes.git  打开pelicanconf.py配置文件，更改或添加THEME为自己喜欢的主题，例如本博客所挑选的gum，更多的配置含义请关注[官方文档](http://docs.getpelican.com/en/3.3.0/settings.html)。  THEME = 'pelican-themes/gum' |  |

#### Examples 例子

##### Listing the installed themes

$ pelican-themes -l

notmyidea

two-column@

simple

$ pelican-themes --list

notmyidea

two-column@

simple

可以看到主题 two-column后跟着@，表明此主题没有复制安装而是链接安装。

$ pelican-themes -v -l

/usr/local/lib/python2.6/dist-packages/pelican-2.6.0-py2.6.egg/pelican/themes/notmyidea

/usr/local/lib/python2.6/dist-packages/pelican-2.6.0-py2.6.egg/pelican/themes/two-column (symbolic link to '/home/skami/Dev/Python/pelican-themes/two-column')

/usr/local/lib/python2.6/dist-packages/pelican-2.6.0-py2.6.egg/pelican/themes/simple

##### Installing themes 安装主题

你可以安装一个或多个主题：

$ pelican-themes --install ~/Dev/Python/pelican-themes/notmyidea-cms --verbose

# pelican-themes --install ~/Dev/Python/pelican-themes/notmyidea-cms\

~/Dev/Python/pelican-themes/martyalchin \

--verbose

# pelican-themes -vi ~/Dev/Python/pelican-themes/two-column

* 1

##### Removing themes 移除主题

# pelican-themes --remove two-column

* 1

# pelican-themes -r martyachin notmyidea-cmd -v

* 1

##### Creating symbolic links 符号安装

# pelican-themes --symlink ~/Dev/Python/pelican-themes/two-column

* 1

上面的例子，two-column主题作为符号安装进了主题目录，我们可以使用它，同时我们可以开发改进主题而不用reinstall。

这在开发主题的过程中很有用：

$ sudo pelican-themes -s ~/Dev/Python/pelican-themes/two-column

$ pelican ~/Blog/content -o /tmp/out -t two-column

$ firefox /tmp/out/index.html

$ vim ~/Dev/Pelican/pelican-themes/two-column/static/css/main.css

$ pelican ~/Blog/content -o /tmp/out -t two-column

$ cp /tmp/bg.png ~/Dev/Pelican/pelican-themes/two-column/static/img/bg.png

$ pelican ~/Blog/content -o /tmp/out -t two-column

$ vim ~/Dev/Pelican/pelican-themes/two-column/templates/index.html

$ pelican ~/Blog/content -o /tmp/out -t two-column

##### Doing several things an once

--install, --remove和 --symlink可以同时出现在一个命令中：

# pelican-themes --remove notmyidea-cms two-column \

--install ~/Dev/Python/pelican-themes/notmyidea-cms-fr \

--symlink ~/Dev/Python/pelican-themes/two-column \

--verbose

### pelican主题更换

首先在[pelican主题开源库](https://github.com/getpelican/pelican-themes)中找到自己想要的主题，本次替换的主题是[pelican-bootstrap3](https://github.com/getpelican/pelican-themes/tree/master/pelican-bootstrap3)，这个主题的配置好像跟其他的不一样，需要多配置些东西。

在根目录下的pelicanconf.py文件中添加或更改自己喜欢的主题。

THEME = 'pelican-themes/pelican-bootstrap3'

PLUGIN\_PATHS = ['pelican-plugins', ]

PLUGINS = ['i18n\_subsites', ]

JINJA\_ENVIRONMENT = {'extensions': ['jinja2.ext.i18n'],}

1. 然后clone <https://github.com/getpelican/pelican-plugins> 到根目录下(本主题所需)，配置已经在上一步骤完成了。
2. 最后在命令行输入pip install python-gettext安装成功后再输入pelican content 即可输入至output文件夹。

`make html`编译网页

### git提交代码

先输入`git status`可查看修改状态。

git add.

git pull origin master

git commit -m '注释'

git push origin master

最后在网页打开自己的github博客地址即可查看。 例:<https://udbbbn.github.io/> (现已替换成jekyll模板) [参考链接1](https://github.com/getpelican/pelican-themes/issues/507) [参考链接2](https://github.com/getpelican/pelican-themes/tree/master/pelican-bootstrap3)

### 学习笔记

假设我们将主题放在 ~/pelican-themes文件夹下，先将主题拿到本地：

git clone --recursive https://github.com/getpelican/pelican-themes ~/pelican-themes

* 1

现在在本地我们拥有了pelican-themes的分支。安装主题，在blog文件夹下运行：

pipenv run pelican-themes -i ~/pelican-themes/themes-name

pelican-themes -i tuxlite\_tbs

还记得吗？我为我的环境配置了pipenv。所以运行的时候要注意。   
安装完成后用pipenv run pelican-themes -l查看主题   
选用主题时，修改 pelicanconf.py文件，添加：

THEME = "themename"

保存文件并生成站点。

开发主题的时候注意修改~/pelican-themes下的文件，output文件夹下的修改在站点生成的时候会被覆盖.

## Pelican参考资源

### Pelican4.1.0博客系统官方文档中文版

https://pelican-docs-latest-cn.readthedocs.io/zh\_CN/latest/

https://github.com/elvynlee/pelican\_docs\_cn

https://github.com/getpelican/pelican

1. [使用github搭建网站](http://blog.csdn.net/pipisorry/article/details/51707366)   
2. [一步步在GitHub上创建博客主页](http://www.pchou.info/ssgithubPage/2013-01-03-build-github-blog-page-01.html)   
3. [Github Help](https://help.github.com/categories/customizing-github-pages/)   
4. [pelican 的官方页](https://blog.getpelican.com/)   
5. [pelican 中文版](http://pelican-zh.readthedocs.io/en/latest/zh-cn/)，资料太古老，只做参考，不要实操   
6. [Pelican＋Github博客搭建详细教程](https://www.cnblogs.com/cciejh/archive/2016/02/04/blog_building.html)

Github:<https://github.com/>

Github Ppages:<http://pages.github.com/>

git:<http://git-scm.com/>

python:<http://www.python.org/>

pip:<https://pypi.python.org/pypi>

pelican:<http://blog.getpelican.com/>

markdown:<http://daringfireball.net/projects/markdown/syntax>

## Pelican Plugins资源

| **Plugin** | **Description** |
| --- | --- |
| Ace Editor | Replace default **<code>** by an [Ace](https://ace.c9.io) code editor with settings configure on pelicanconf.py. |
| Always modified | Copy created date metadata into modified date for easy "latest updates" indexes |
| AsciiDoc reader | Use AsciiDoc to write your posts. |
| Asset management | Use the Webassets module to manage assets such as CSS and JS files. |
| Auto Pages | Generate custom content for generated Author, Category, and Tag pages (e.g. author biography) |
| Backref Translate | Add a new attribute (is\_translation\_of) to every article/page (which is a translation) pointing back to the original article/page which is being translated |
| Better code line numbers | Allow code blocks with line numbers to wrap |
| Better code samples | Wraps table blocks with div > .hilitewrapper > .codehilitetable class attribute, allowing for scrollable code blocks. |
| Better figures/samples | Adds a style="width: ???px; height: auto;" attribute to any <img> tags in the content |
| bootstrap-rst | Provides most (though not all) of Bootstrap's features as rst directives |
| bootstrapify | Automatically add bootstraps default classes to your content |
| Category Order | Order categories (and tags) by the number of articles in that category (or tag). |
| CJK auto spacing | Inserts spaces between Chinese/Japanese/Korean characters and English words |
| Clean summary | Cleans your summary of excess images |
| Code include | Includes Pygments highlighted code in reStructuredText |
| Collate content | Makes categories of content available to the template as lists through a collations attribute |
| Creole reader | Allows you to write your posts using the wikicreole syntax |
| CSS HTML JS Minify | Minifies all CSS, HTML and JavaScript files in the output path after site generation. |
| Custom article URLs | Adds support for defining different default URLs for different categories |
| CTags generator | Generates a "tags" file following the CTags in the "content/" directory, to provide autocompletion for code editors that support it. |
| Dateish | Treat arbitrary metadata fields as datetime objects |
| Dead Links | Manage dead links (website not available, errors such as 403, 404) |
| Disqus static comments | Adds a disqus\_comments property to all articles. Comments are fetched at generation time using disqus API |
| Encrypt content | Password protect pages and articles |
| Events | Add event start, duration, and location info to post metadata to generate an iCalendar file |
| Extract table of content | Extracts table of contents (ToC) from article.content |
| Figure References | Provides a system to number and references figures |
| Filetime from Git | Uses Git commit to determine page date |
| Filetime from Hg | Uses Mercurial commit to determine page date |
| Footer Insert | Add standardized footer (e.g., author information) at end of every article |
| GA Page View | Display Google Analytics page views on individual articles and pages |
| Gallery | Allows an article to contain an album of pictures |
| Gist directive | This plugin adds a gist reStructuredText directive. |
| GitHub wiki | Converts a flat github wiki into a structured read only wiki on your site |
| GitHub activity | On the template side, you just have to iterate over the github\_activity variable |
| Global license | Allows you to define a LICENSE setting and adds the contents of that license variable to the article's context |
| Glossary | Adds a variable containing definitions extracted from definition lists in articles and pages. This variable is visible to all page templates. |
| Goodreads activity | Lists books from your Goodreads shelves |
| GooglePlus comments | Adds GooglePlus comments to Pelican |
| Gravatar | Assigns the author\_gravatar variable to the Gravatar URL and makes the variable available within the article's context |
| Gzip cache | Enables certain web servers (e.g., Nginx) to use a static cache of gzip-compressed files to prevent the server from compressing files during an HTTP call |
| Headerid | This plugin adds an anchor to each heading so you can deeplink to headers in reStructuredText articles. |
| HTML entities | Allows you to enter HTML entities such as &copy;, &lt;, &#149; inline in a RST document |
| HTML tags for rST | Allows you to use HTML tags from within reST documents |
| I18N Sub-sites | Extends the translations functionality by creating internationalized sub-sites for the default site |
| ical | Looks for and parses an .ics file if it is defined in a given page's calendar metadata. |
| Image Process | Automates the processing of images based on their class attributes |
| Interlinks | Lets you add frequently used URLs to your markup using short keywords |
| Jinja2 Content | Allows the use of Jinja2 template code in articles, including include and import statements. Replacement for pelican-jinja2content. |
| JPEG Reader | Create image gallery pages based on content of JPEG metadata |
| Just table | Allows you to easily create and manage tables. You can embed the tables into posts with a simple way. |
| Libravatar | Allows inclusion of user profile pictures from libravatar.org |
| Link Class | Allows the insertion of class attributes into generated <a> elements (Markdown only) |
| Linker | Allows the definition of custom linker commands in analogy to the builtin {filename}, {attach}, {category}, {tag}, {author}, and {index} syntax |
| Liquid-style tags | Allows liquid-style tags to be inserted into markdown within Pelican documents |
| Load CSV | Adds csv Jinja tag to display the contents of a CSV file as an HTML table |
| Markdown Inline Extend | Enables you to add customize inline patterns to your markdown |
| Markdown-metaYAML | Pelican reader to enable YAML-style metadata in markdown articles |
| Math Render | Gives pelican the ability to render mathematics |
| Mbox Reader | Generate articles automatically via email, given a path to a Unix mbox |
| More Categories | Multiple categories per article; nested categories (foo/bar, foo/baz) |
| Multi Neighbors | Adds a list of newer articles and a list of older articles to every article's context. |
| Multi parts posts | Allows you to write multi-part posts |
| Neighbor articles | Adds next\_article (newer) and prev\_article (older) variables to the article's context |
| Open graph | Generates Open Graph tags for your articles |
| Optimize images | Applies lossless compression on JPEG and PNG images |
| Org Reader | Create posts via Emacs Orgmode files |
| Page View | Pull page view count from Google Analytics. |
| Panorama | Creates charts from posts metadata |
| PDF generator | Automatically exports articles and pages as PDF files |
| PDF Images | If an img tag contains a PDF, EPS or PS file as a source, this plugin generates a PNG preview which will then act as a link to the original file. |
| Pelican Cite | Produces inline citations and a bibliography in articles and pages, using a BibTeX file. |
| Pelican Comment System | Allows you to add static comments to your articles |
| pelican-ert | Allows you to add estimated reading time of an article |
| Pelican-flickr | Brings your Flickr photos & sets into your static website |
| Pelican Genealogy | Add surnames and people so metadata and context can be accessed from within a theme to provide surname and person pages |
| Pelican Gist tag | Easily embed GitHub Gists in your Pelican articles |
| Pelican Github Projects | Embed a list of your public GitHub projects in your pages |
| pelican\_javascript | Allows you to embed Javascript and CSS files into individual articles |
| Pelican Jinja2Content | Allows the use of Jinja2 template code in articles, including include and import statements |
| Pelican Link Class | Set class attribute of <a> elements according to whether the link is external or internal |
| Pelican Meetup Info | Include your Meetup.com group and event information on generated pages and articles |
| Pelican Page Hierarchy | Creates a URL hierarchy for pages that matches the filesystem hierarchy of their sources |
| Pelican Page Order | Adds a page\_order attribute to all pages if one is not defined. |
| Pelican Themes Generator | Generates theme screenshots from the Pelican Themes repository |
| pelican-rdf | Allows the processing of .rdf vocabularies, and the generation of a lightweight documentation. |
| pelican-toc | Generates a Table of Contents and make it available to the theme via article.toc |
| Pelican Vimeo | Enables you to embed Vimeo videos in your pages and articles |
| Pelican YouTube | Enables you to embed YouTube videos in your pages and articles |
| pelicanfly | Lets you type things like i ♥ :fa-coffee: in your Markdown documents and have it come out as little Font Awesome icons in the browser |
| Photos | Add a photo or a gallery of photos to an article, or include photos in the body text. Resize photos as needed. |
| permalink | Enables a kind of permalink using html redirects. |
| Pin to top | Pin Pelican's article(s) to top "Sticky article" |
| PlantUML | Allows you to define UML diagrams directly into rst documents using the great PlantUML tool |
| Post Revision | Extract article and page revision information from Git commit history |
| Post statistics | Calculates various statistics about a post and store them in an article.stats dictionary |
| Random article | Generates a html file which redirect to a random article |
| Read More link | Inserts an inline "read more" or "continue" link into the last html element of the object summary |
| Readtime | Adds article estimated read time calculator to the site, in the form of '<n> minutes'. |
| Related posts | Adds the related\_posts variable to the article's context |
| Render Math | Render mathematics in content via the MathJax Javascript engine |
| Replacer | Replace a text of a generated HTML |
| Representative image | Extracts a representative image (i.e, featured image) from the article's summary or content |
| RMD Reader | Create posts via knitr RMarkdown files |
| Section number | Adds section numbers for article headers, in the form of 2.3.3 |
| Series | Groups related articles into a series |
| Shaarli poster | Upload newly redacted articles onto a specified [Shaarli](https://github.com/shaarli/Shaarli) instance. |
| Share post | Creates share URLs of article |
| Shortcodes | Easy and explicit inline jinja2 macros |
| Show Source | Place a link to the source text of your posts. |
| Similar Posts | Adds a list of similar posts to every article's context. |
| Simple footnotes | Adds footnotes to blog posts |
| Sitemap | Generates plain-text or XML sitemaps |
| Slim | Render theme template files via Plim, a Python port of Slim, instead of Jinja |
| Static comments | Allows you to add static comments to an article |
| Subcategory | Adds support for subcategories |
| Sub parts | Break a very long article in parts, without polluting the timeline with lots of small articles. |
| Summary | Allows easy, variable length summaries directly embedded into the body of your articles |
| tag\_cloud | Provides a tag\_cloud |
| Textile Reader | Adds support for Textile markup |
| Thumbnailer | Creates thumbnails for all of the images found under a specific directory |
| Tipue Search | Serializes generated HTML to JSON that can be used by jQuery plugin - Tipue Search |
| Touch | Does a touch on your generated files using the date metadata from the content |
| Twitter Bootstrap | Defines some rst directive that enable a clean usage of the twitter bootstrap CSS and Javascript components |
| txt2tags\_reader | Reader that renders txt2tags markup in content |
| Unity WebGL | Easily embed Unity3d games into posts and pages |
| Video Privacy Enhancer | Increases user privacy by stopping YouTube, Google, et al from placing cookies via embedded video |
| Webring | Add a webring to your site from a list of web feeds (e.g. RSS/Atom) |
| W3C validate | Submits generated HTML content to the W3C Markup Validation Service |
| Yuicompressor | Minify CSS and JS files on building step |

Please refer to the Readme file in a plugin's folder for detailed information about that plugin.

## Pelican主题资源

### 资源一览

pelican 的主题在 <https://github.com/getpelican/pelican-themes>，页面中有使用说明。选取主题的时候可以看<http://www.pelicanthemes.com/>页面挑选。

|  |  |  |
| --- | --- | --- |
| 主题名称 | GitHub | 维护状态 |
| elegant | https://github.com/Pelican-Elegant/elegant |  |
|  |  |  |
| buruma | https://github.com/ivanhercaz/buruma.git | 2019 |
|  |  |  |
|  |  |  |
| niu-x2 | https://github.com/wilbur-ma/niu-x2 | (中文的，不用调整CSS）最后更新2013年 |
|  |  |  |

[attila](https://github.com/arulrajnet/attila/tree/02dcad911ba1eb2d797a79ec008a810d89a2fde1),

[backdrop](https://github.com/getpelican/pelican-themes/tree/master/backdrop),

[blueidea](https://github.com/blueicefield/pelican-blueidea/tree/8f11c0e3b4b8e9ef45d1243b0175db91b7b42ba8),

[clean-blog](https://github.com/gilsondev/pelican-clean-blog/tree/ea156f8f1741e473bc0ab848b7c8898112d6ffb5),

,

[voidy-bootstrap](https://github.com/robulouski/voidy-bootstrap/tree/83f4d802710bbfa20123a9b7f1921424e292ef53),

### 主题gum修改

[**现状**](https://www.iloveanan.com/pelicanzhu-ti-xiu-gai.html#_1)

现在使用的主题是[gum](https://github.com/getpelican/pelican-themes/tree/master/gum)，不喜欢的地方有：

1. 侧边栏：没有添加pages，但有Pages标题；每篇文章都添加tag，但Tag下没有内容
2. Home导航太单调，希望可以有各个分类的链接

[**修改**](https://www.iloveanan.com/pelicanzhu-ti-xiu-gai.html#_2)

因为这个主题是很早就选定的，所以一开始我是直接去看gum的源码，内容倒也不多，两个文件夹，一个css一个是html模版。但是，这样的方式是错误滴～正确的方式是先过一遍pelican的使用指南，然后再看gum的使用指南。中间遇到的问题就挑重点记录了。

[**菜单显示分类**](https://www.iloveanan.com/pelicanzhu-ti-xiu-gai.html#_3)

pelican内置变量 DISPLAY\_CATEGORIES\_ON\_MENU，决定是否在菜单中显示分类。此外，如果希望同时显示Pages，除了需要在本地添加pages外，还需要在配置文件中添加

MENUITEMS = (

('About', '/pages/About.html'),

)

添加pages的方法是，现在content所在文件夹下新建 pages，然后在其中创建形如article的文件。这里，我使用的是md格式，那么我就创建一个 About.md文件。  
此外，还修改了category.html，将以下部分删除，因为我希望点击分类的时候和首页一样，直接显示文章列表，而不是先出现如下的字样：

{% **block** content\_title %}

**<h2>**Articles in the {{ category }} category**</h2>**

{% **endblock** %}

[**侧边栏**](https://www.iloveanan.com/pelicanzhu-ti-xiu-gai.html#_4)

我希望分类能同时在menu和侧边栏出现，但是查看 sidebar.html 发现，两者分别是 if 和else中，因此将其中的else内容移除。  
同时发现tag下的内容为空，但是html模版文件中有tag内容。此时怀疑是否是变量使用出了问题，排查了一遍将目光定位到

{% **for** tag in tag\_cloud %}

搜索了下，tag\_cloud是一个[插件](http://docs.getpelican.com/en/3.6.3/faq.html" \l "my-tag-cloud-is-missing-broken-since-i-upgraded-pelican)。我的工程文件下已有 plugins 文件夹，赶紧pull了一下，更新到了tag\_cloud。大喜，在配置文件中添加了tag\_cloud的属性，build后发现tag还是依次向下逐个排列，即使我修改模版文件添加了 <ul class="tagcloud">。后来在tag\_cloud库的readme中发现原因：需要将tag\_cloud.py文件直接放置在plugins下，同时要添加相应的css。

现在的形式，差强人意。以后还会陆续修改，借此熟悉pelican和html/css。

### 主题pelican-alchemy修改

https://github.com/nairobilug/pelican-alchemy

### 主题XXXX修改