写作内容

文章和页面

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

“页面”背后的想法是，它们通常不是时间性的，用于不经常更改的内容（例如“关于”或“联系”页面）。

您可以在存储库的`samples/content/`上找到示例内容。

文件元数据

Pelican试图足够聪明，从文件系统获取所需的信息（例如，关于您文章的类别），但您需要以文件中元数据的形式提供一些信息。

如果您以reStructuredText格式编写内容，您可以通过以下语法在文本文件中提供此元数据（为您的文件提供`.rst`扩展名）：

```
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
```

作者和标签列表可以分号分隔，这允许您编写包含逗号的作者和标签：

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

Pelican implements an extension to reStructuredText to enable support for the `abbr` HTML tag. To use it, write something like this in your post:

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

您还可以使用Markdown语法（文件以`.md`、`.markdown`、`.mkd`或`.mdown`结尾）。Markdown生成要求您首先显式安装Python-Markdown软件包，这可以通过`pipinstallMarkdown`完成。

Pelican还支持Markdown扩展，如果它们不包含在默认的`Markdown`软件包中，则可能需要单独安装，并且可以通过`MARKDOWN`设置进行配置和加载。

Markdown帖子的元数据语法应遵循以下模式：

```
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.
```

您还可以拥有自己的元数据密钥（只要它们与保留的元数据关键字不冲突）在模板中使用。下表包含保留的元数据关键字列表：

元数据	描述
`title`	文章或页面的标题
`date`	Publication date (e.g., `YYYY-MM-DD HH:SS`)
`modified`	Modification date (e.g., `YYYY-MM-DD HH:SS`)
`tags`	内容标签，用逗号分隔

`keywords`	内容关键词，用逗号分隔（仅限HTML内容）
`category`	内容类别（仅限一个，而不是多个）
`slug`	URL和翻译中使用的标识符
`author`	内容作者，当只有一个
`authors`	内容作者，当有多个
`summary`	索引页面内容的简要描述
`lang`	内容语言ID（`en`、`fr`等）
`translation`	如果内容是另一个翻译（`true`或`false`）
`status`	内容状态：`draft`、`hidden`或`published`
`template`	用于生成内容的模板名称（不扩展）
`save_as`	将内容保存到此相对文件路径
`url`	用于本文/页面的URL

其他格式（如AsciiDoc）的阅读器可以通过插件获得。有关这些，请参阅鹈鹕插件存储库。

Pelican还可以处理以`.html`和`.htm`结尾的HTML文件。Pelican以非常直截了当的方式解释HTML，从`meta`标签中读取元数据，从`title`标签中读取标题，从`body`标签中读取正文：

```
<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，标准元数据有一个简单的例外：标签可以通过`tags`元数据指定，就像鹈鹕的标准一样，也可以通过`keywords`元数据指定，就像HTML中的标准一样。两者可以互换使用。

请注意，除了标题外，这些内容元数据都不是强制性的：如果未指定日期，并且`DEFAULT_DATE`设置为`'fs'`”，Pelican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`的文件将具有`foobar`的类别。如果您想以其他方式组织文件，而子文件夹的名称不是一个好的类别名称，您可以将设置`USE_FOLDER_AS_CATEGORY`设置为`False`。在解析页面元数据中给出的日期时，Pelican支持W3C建议的子集ISO 8601。

So the title is the only required metadata. If that bothers you, worry not. Instead of manually specifying a title in your metadata each time, you can use the source content file name as the title. For example, a Markdown source file named `Publishing via Pelican.md` would automatically be assigned a title of *Publishing via Pelican*. If you would prefer this behavior, add the following line to your settings file:

```
FILENAME_METADATA = '(?P<title>.*)'
```

笔记

When experimenting with different settings (especially the metadata ones) caching may interfere and the changes may not be visible. In such cases disable caching with `LOAD_CONTENT_CACHE = False` or use the `--ignore-cache` command-line switch.

`modified` should be last time you updated the article, and defaults to `date` if not specified. Besides you can show `modified` in the templates, feed entries in feed readers will be updated automatically when you set `modified` to the current date after you modified your article.

`authors`是以逗号分隔的文章作者列表。如果只有一个作者，您可以使用`author`字段。

如果您没有明确指定给定帖子的摘要元数据，则可以使用`SUMMARY_MAX_LENGTH`设置来指定文章开头有多少单词用作摘要。

您还可以通过在`FILENAME_METADATA`设置中设置的正则表达式从文件名中提取任何元数据。所有匹配的命名组都将在元数据对象中设置。`FILENAME_METADATA`设置的默认值将仅从文件名中提取日期。例如，如果您想提取日期和弹头，您可以设置以下内容：`'(?P<date>\d{4}-\d{2}-\d{2})_(?P<slug>.*)'`

请注意，文件中可用的元数据优先于从文件名中提取的元数据。

页面

如果您在内容文件夹中创建一个名为`pages`的文件夹，其中的所有文件都将用于生成静态页面，例如**关于页面**或**联系人**页面。（请参阅下面的文件系统布局示例。）

您可以使用DISPLAY`DISPLAY_PAGES_ON_MENU`设置来控制所有这些页面是否显示在主导航菜单中。（默认为`True`。）

如果您想排除任何页面链接到或列在菜单中，请在其元数据中添加`status:hidden`属性。这对于制作适合您网站生成主题的错误页面等内容非常有用。

静态内容

静态文件是文章和页面以外的文件，按原样复制到输出文件夹，无需处理。您可以使用项目的`pelicanconf.py`文件的`STATIC_PATHS`设置来控制复制哪些静态文件。Pelican的默认配置包括此的`images`目录，但其他配置必须手动添加。此外，还包括显式链接的静态文件（见下文）。

同一目录中的混合内容

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，而不会暴露生成站点中的页面源。任何此类目录都必须添加到`STATIC_PATHS`和`PAGE_PATHS`（或`STATIC_PATHS`和`ARTICLE_PATHS`）。Pelican将正常识别和处理页面源文件，并复制剩余的文件，就像它们生活在为静态文件保留的单独目录中一样。

注意：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的同一位置。最简单的方法是使用`{attach}`链接语法（如下所述）。或者，`STATIC_SAVE_AS`、`PAGE_SAVE_AS`和`ARTICLE_SAVE_AS`设置（以及相应的`*_URL`设置）可以配置为将不同类型的文件放在一起，就像在早期版本的鹅鹕中一样。

链接到内部内容

从Pelican 3.1开始，现在可以指定指向*源内容*层次结构中文件的站点内链接，而不是*生成*层次结构中的文件。这使得从当前帖子链接到可能与该帖子并列的其他内容变得更容易（而不必确定网站生成后其他内容将放置在哪里）。

To link to internal content (files in the `content` directory), use the following syntax for the link target: `{filename}path/to/file` Note: forward slashes, `/`, are the required path separator in the `{filename}` directive on all operating systems, including Windows.

例如，鹈鹕项目可能是这样构建的：

```
website/
├── content
│   ├── category/
│   │   └── article1.rst
│   ├── article2.md
│   └── pages
│       └── about.md
└── pelican.conf.py
```

在本例中，`article1.rst`可以如下所示：

```
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>`_
```

和`article2.md`：

```
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)
```

链接到静态文件

您可以使用`{static}path/to/file`链接到静态内容。使用此语法链接的文件将自动复制到输出目录，即使包含它们的源目录不包含在项目`pelicanconf.py`文件的`STATIC_PATHS`设置中。

例如，项目的内容目录可能结构如下：

```
content
├── images
│   └── han.jpg
├── pdfs
│   └── menu.pdf
└── pages
    └── test.md
```

`test.md`将包括：

```
![Alt Text]({static}/images/han.jpg)
[Our Menu]({static}/pdfs/menu.pdf)
```

然后，网站生成将`han.jpg`复制到`output/images/han.jpg`，`menu.pdf`到`output/pdfs/menu.pdf`，并在`test.md`中编写适当的链接。

如果您使用`{static}`链接到文章或页面，这将变成其源代码的链接。

附加静态文件

从Pelican 3.5开始，静态文件可以使用链接目标的此语法“附加”到页面或文章：`{attach}path/to/file`这类似于`{static}`语法，但也将静态文件重新定位到链接文档的输出目录中。如果静态文件来自链接文档源代码下方的子目录，则该关系将在

输出时保留。否则，它将成为链接文档的兄弟姐妹。

这仅适用于链接到静态文件。

例如，项目的内容目录可能结构如下：

```
content
├── blog
│   ├── icons
│   │   └── icon.png
│   ├── photo.jpg
│   └── testpost.md
└── downloads
    └── archive.zip
```

`pelicanconf.py` 将包括：

```
PATH = 'content'
ARTICLE_PATHS = ['blog']
ARTICLE_SAVE_AS = '{date:%Y}/{slug}.html'
ARTICLE_URL = '{date:%Y}/{slug}.html'
```

`testpost.md` 将包括：

```
Title: Test Post
Category: test
Date: 2014-10-31

![Icon]({attach}icons/icon.png)
![Photo]({attach}photo.jpg)
[Downloadable File]({attach}/downloads/archive.zip)
```

然后，站点生成将产生一个结构如下的输出目录：

```
output
└── 2014
    ├── archive.zip
    ├── icons
    │   └── icon.png
    ├── photo.jpg
    └── test-post.html
```

请注意，使用`{attach}`链接的所有文件最终都位于文章的输出目录中或下方。

如果静态文件被多次链接，`{attach}`的迁移功能将仅在要处理的第一个链接中起作用。在第一个链接之后，鹈鹕将把`{attach}`视为`{static}`。这避免了破坏已经处理过的链接。

从多个文档链接到文件时要小心：由于文件的第一个链接最终确定了其位置，而鹈鹕没有定义处理文档的顺序，因此在多个文档链接的文件中使用`{attach}`可能会导致其位置从一个站点构建更改为下一个站点构建。（在实践中是否发生这种情况将取决于操作系统、文件系统、Pelican版本以及正在从项目中添加、修改或删除的文档。）任何链接到文件旧位置的外部网站都可能会发现其链接已损坏。**因此，建议仅当您在文件的所有链接中使用{attach}时，并且仅当链接文档共享单个目录时，才使用{attach}。**在这种情况下，文件的输出位置在未来的构建中不会改变。在无法采取这些预防措施的情况下，请考虑使用`{static}`链接而不是`{attach}`，并让文件的位置由项目的`STATIC_SAVE_AS`和`STATIC_URL`设置确定。（每文件`save_as`和`url`覆盖仍然可以在`EXTRA_PATH_METADATA`中设置。）

笔记

When using `{attach}`, any parent directory in `*_URL` / `*_SAVE_AS` settings should match each other. See also: URL settings

链接到作者、类别、索引和标签

您可以使用`{author}name`、`{category}foobar``{index}`和`{tag}tagname`语法链接到作者、类别、索引和标签。

不建议使用的内部链接语法

为了与早期版本保持兼容，除了内部链接的卷曲大括号（`{}`）外，Pelican仍然支持垂直条（`||`）。例
如：`|filename|an_article.rst`，`|tag|tagname`，`|category|foobar`。语法从`||`更改为`{}`，以避免与Markdownextensions或reST指令发生冲突。同样，Pelican仍然支持链接到`{filename}`的静态内容。语法更改为`{static}`，以便链接到生成的文章和页面及其静态源。

对旧语法的支持最终可能会被删除。

包括其他文件

Markdown和reStructuredText语法都为此提供了机制。

以下是使用include指令的**reStructuredText**的一些示例：

```
.. include:: file.rst
```

包括一个由两个标识符分隔的文件片段，突出显示为C++（也可以根据行号进行切片）：

```
.. include:: main.cpp
    :code: c++
    :start-after: // begin
    :end-before: // end
```

包含一个原始的HTML文件（或内联SVG），并将其直接放入输出中，而无需任何处理：

```
.. raw:: html
    :file: table.html
```

对于**Markdown**，必须依赖扩展。例如，使用mdx_include插件：

```
```html
{! template.html !}
```
```

导入现有站点

可以使用简单的脚本从WordPress、Tumblr、Dotclear和RSS提要导入您的网站。请参阅导入现有站点。

翻译

可以翻译文章。为此，您需要在文章/页面中添加`lang`元属性，并设置`DEFAULT_LANG`设置（默认情况下为英语[en]）。有了这些设置，将只列出具有默认语言的文章，每篇文章都将附有该文章的可用翻译列表。

> **笔记**
>
> 此核心鹈鹕功能不会为每种语言创建带有翻译模板的子站点（例如`example.com/de`）。对于此类高级功能，可以使用i18n_subsites插件。

By default, Pelican uses the article's URL "slug" to determine if two or more articles are translations of one another. (This can be changed with the `ARTICLE_TRANSLATION_ID` setting.) The slug can be set manually in the file's metadata; if not set explicitly, Pelican will auto-generate the slug from the title of the article.

这是两篇文章的例子，一篇是英文文章，另一篇是法文。

英文文章：

```
Foobar is not dead
##################

:slug: foobar-is-not-dead
:lang: en

That's true, foobar is still alive!
```

法文版本：

```
Foobar n'est pas mort !
#######################

:slug: foobar-is-not-dead
:lang: fr

Oui oui, foobar est toujours vivant !
```

尽管帖子内容质量很高，但您可以看到，两篇文章之间唯一的共同点是弹头，它在这里用作标识符。如果您不想以这种方式明确定义弹头，那么您必须确保翻译的文章标题相同，因为弹头将从文章标题自动生成。

如果您不希望`DEFAULT_LANG`设置检测到特定文章的原始版本，请使用`translation`元数据指定哪些帖子是翻译：

```
Foobar is not dead
##################

:slug: foobar-is-not-dead
:lang: en
:translation: true

That's true, foobar is still alive!
```

语法高亮显示

Pelican可以为您的代码块提供彩色语法高亮显示。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`指令指定要突出显示的代码类型（在这些示例中，我们将使用`python`）：

```
.. code-block:: python

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

对于使用CodeHilite扩展提供语法高亮显示的Markdown，请包括代码块正上方的语言标识符，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.")
```

指定的标识符（例如`python`、`ruby`）应该出现在可用词典列表中。

使用reStructuredText时，代码块指令中提供以下选项：

选项	有效值	描述
anchorlinenos	不适用	如果在<a>标签中存在包装行号。
类前缀	符号串	字符串前于令牌类名
hl_lines	数字	要突出显示的行列表，其中要突出显示的行号由空格分隔。这类似于狮身人面像中的`emphasize-lines`，但它不支持由连字符或逗号分隔的行号。
线锚	符号串	使用此字符串和-行号将每行包裹在锚中。
亚麻布	符号串	如果在表中存在或设置为“表”输出行号，如果设置为“内联”，则内联输出它们。“无”表示不输出此表的行号。
linenospecial	号码	如果设置了第n行，将获得“特殊”css类。
linenostart	号码	第一行的行号。
linenostep	号码	打印每个n行编号。
线条分隔符	符号串	字符串在代码行之间打印，默认情况下为“n”。
线跨度	符号串	使用此和-行号将每行包裹在跨度中。
没有背景	不适用	如果设置，不要为包装元素输出背景颜色
报废	不适用	如果设置，则完全不要包装令牌。
标签文件	符号串	ctags文件用于名称定义。
tagurlformat	符号串	ctag链接的格式。

请注意，根据版本，您的Pygments模块可能没有所有这些选项可用。有关每个选项的更多详细信息，请参阅Pygments文档的*HtmlFormatter*部分。

例如，以下代码块启用行号，从153开始，并将Pygments CSS类的前缀为*pgcss*，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>
```

也可以在鹈鹕设置文件中指定`PYGMENTS_RST_OPTIONS`变量，以包含将自动应用于每个代码块的选项。

例如，如果您想为每个代码块显示行号和CSS前缀，您将此变量设置为：

```
PYGMENTS_RST_OPTIONS = {'classprefix': 'pgcss', 'linenos': 'table'}
```

如果指定，单个代码块的设置将覆盖设置文件中的默认值。

发布草稿

如果您想将文章或页面作为草稿发布（例如，供朋友在发布前查看），您可以在其元数据中添加`Status:draft`属性。然后，该文章将输出到`drafts`文件夹，并且没有列在索引页面或任何类别或标签页面上。

如果您的文章应该自动以草稿形式发布（在文章完成之前不会意外发布），请在`DEFAULT_METADATA`中包含状态：

```
DEFAULT_METADATA = {
    'status': 'draft',
}
```

要在默认状态为`draft`时发布帖子，请将帖子的元数据更新为包括`Status:published`。

隐藏帖子

与页面一样，帖子也可以使用`Status:hidden`属性标记为隐藏。隐藏的帖子将按预期输出到`ARTICLE_SAVE_AS`，但默认情况下不会包含在标签或类别索引中，也不会包含在主文章提要中。这具有创建“未上市”帖子的效果。