Pelican - Part 2

Writing Articles With Markdown

Whenever we create new content for our site it will always follow the same basic process:

- Create a new markdown file in the *content* folder
- Define the file metadata
- Write the content
- Regenerate the website

To learn how to create content we are going to write multiple article posts. Let's begin by creating a new markdown file called *first_post.md* in the *content* folder. A markdown file is simply a plain .txt file with the extension changed to .md. If you are unsure about Markdown that's OK for now. We will discuss it shortly.

```
blog
└── source
     ├── content
     │   └── first_post.md (new)
     ├── plugins
     └── theme
```

Next we will define the metadata for first_post.md. Metadata is information about the article post. It gives Pelican direction on how to generate the static website files and some of the metadata will be displayed within the webpage itself. Open the blank *first_post.md* and add the following lines:

```
Title: My First Post
Author: Pat McCavana
Date: 2020-03-13 12:30
Category: Portfolio Post
Tags: python, markdown, pelican
Slug: my-first-post

The article content goes here
```

Metadata should always appear at the top of the file. Each piece of metadata must be on a new line and follows the format metadata_keyword: data. Here's a list of the most common keywords and their usage:

Keyword	Required?	Usage
Title	Yes	title of the blog post
Author	Yes	author of the blog post
Date	Yes	published date in the format YYYY-mm-dd hh:mm
Modified	No	last edit date in the format YYYY-mm-dd hh:mm
Category	Yes	general topic of the blog post

Keyword	Required?	Usage
Tags	No	topics covered in the blog post separated by a comma
Summary	No	brief one or two sentence synopsis of the blog post
Slug	No	name of the .html file to be generated. If none is defined the slug will be the post's title separated by "-" symbols
Status	No	choose one of published, draft or hidden

Regenerate the website to view the result. Our first post is now showing on the main page. Take a moment to browse the post and consider how the metadata we supplied affected the webpage that was generated.

```
pelican content
pelican --listen
```

Now we will use Markdown to write our post's content. Markdown is a language written for non-programmers that uses symbols to represent HTML. We apply markdown to a plain text file with the extension .md to do things like change the style (e.g. bold, italics), insert pictures and create links. When we use Pelican to generate our website it will read the markdown file and convert it to HTML.

Anyone can learn the basics of Markdown in about 10 minutes. The language's syntax is very well documented on the web so we will not discuss it in full detail here. Instead we will review a few sample posts to understand the basics which will include examples specific to Pelican. I recommend you check out either the interactive tutorial at <markdownguide.org> or Adam Pritchard's Markdown Cheatsheet on Github before moving on with the tutorial.

Create a file called *markdown_examples_1.md* in the same location as *first_post.md*. Insert the following text:

```
Title: Markdown Examples Part 1
Author: Pat McCavana
Date: 2019-01-02 12:00
Category: Introduction
Tags: markdown

Basic markdown examples are shown below

This text is **bold**
This text is *italic*
This text is **_italic and bold_**

### An h3 heading...

A list with bullets:
* Bullet a
* Bullet b
* Bullet c

Here's a blockquote:

Here's a table:
```

```
| Column1 | Column 2 | Column 3
|---|---|---|
| Value 1 | Value 2 | Value 3 |
| Value 4 | Value 5 | Value 6 |
| Value 7 | Value 8 | Value 9 |
```

Now generate the website and review our post.

```
pelican content
pelican --listen
```

The post we wrote shows the basics of how to use Markdown. When the website was generated the Markdown was converted into HTML. HTML elements can be used when writing a markdown file but the markdown syntax is much simpler to write.

Let's prepare to write our second post. Download the python icon image we will be using by clicking on this link. Copy the image to the content folder as shown below.

```
blog
└── source
     ├── content
     |   └── first_post.md
     |   └── markdown_example_1.md
     |   └── python_icon.png (new)
     ├── plugins
     └── theme
```

Create a file called *markdown_examples_2.md* in the same location as *first_post.md* and *markdown_examples_1.md*. Insert the following text:

```
Title: Markdown Examples Part 2
Author: Pat McCavana
Date: 2020-03-14 13:00
Category: Intoduction
Tags: markdown

Images can be displayed in Markdown.
Text within the square brackets is the image name. The path to the image goes between
The {static} tag indicates the image is stored in the content folder. This setting ca

![python logo]({static}/python_icon.png)

Links to downloadable content such as PDF files are written similarly to image files

[Pelican Documenation]({static}/pelican.pdf)

A link to a different blog post on our website is written exactly the same.
Text within the square brackets can be clicked on to travel to the website between th
The {filename} tag indicates we want to follow the link to a webpage rather than the
```

```
[First Post]({filename}/first_post.md)

Or we can link to another external website by supplying the web address.

[Python Package Index](https://pypi.org)
Generate the website and review the result.
```

```
pelican content
pelican --listen
```

For our final blog post we will write a Python code block. Pelican displays code blocks using the popular Pygments snytax highlighter. You will recall we setup the Pygments theme 'monokai' in pelicanconf.py earlier. Also, Pygments is not limited to only Python. It can handle almost any language you can think of. Check out the list of supported languages here.

Create a file called *markdown_examples_3.md* in the *content* folder. Insert the following text:

```
Title: Markdown Examples Part 3
Author: Matthew Devaney
Date: 2019-01-04 12:00
Category: blogging
Tags: markdown, pygments

Code blocks are preceeded by an indent, three : symbols and the name of the language.
All of the following code will be highlighted while the text is indented.

    :::python3
    def do_twice(func):
    def wrapper_do_twice(*args, **kwargs):
        return func(*args, **kwargs).lower()
    return wrapper_do_twice

    @do_twice
    def say_whee(some_text):
        print(some_text)

    x = 'Whee!'
    say_whee(x)
```

Generate the site one last time and review the code block.

```
pelican content
pelican --listen
```

Writing Pages vs. Articles

Written content in Pelican takes one of two forms: articles or pages. Articles are frequently updated content that can be ordered by date. Our blog posts are all examples of articles. Pages are more permanent in

nature and are not associated with dates such as an about me page. They only appear on the navigation bar of our blog and not the list of recent posts. We are going to make an about me page for our blog.

Pelican determines which content is pages vs. articles based on where it is located in the filesystem. Make a new folder called pages within the content folder and then create a new markdown file about_me as shown below.

```
blog
└── source
     └── content
          └── pages
              └── about_me.md
```

Page content is written exactly the same as articles, however, less metadata is required. We do not need to include a category or tags. Date and modified are also not necessary but we will still include them in case we want to know this information in the future.

```
title: About Me
date: 2019-02-01
modified: 2019-02-01
status: published

My name is Pat McCavana.  I work as an medical physicst in Dublin, Ireland.  My inter
```

Generate the website and navigate to the about me page to view the result of our work.

```
pelican content
pelican --listen
```

Keeping The Content Folder Organized A small content-oriented website such as our blog can quickly grow into a large website as time goes on. Organizing the file structure of our website in a logical manner will help us minimize the difficulty of maintaining our blog, leaving us more time to spend productively on creating new content. Look at the diagram below which shows our current setup of the content folder. A flat file structure like this one might be fine to use for a blog with 10 posts but it would become significantly harder after 100 posts.

```
blog
└── source
     └── content
          └── pages
          |   └── about_me.md
          └── first_post.md
          └── markdown_example_1.md
          └── markdown_example_2.md
          └── markdown_example_3.md
          └── pelican.pdf
          └── python_icon.png
```

We are going to create a file-structure that is organized by content type instead. Create a folder folder named articles in the content folder. Move the article files into the folder exactly as shown below.

```
blog
└── source
    └── content
        └── articles
        |   └── first_post.md
        |   └── markdown_example_1.md
        |   └── markdown_example_2.md
        |   └── markdown_example_3.md
        └── pages
        |   └── about_me.md
        └── pelican.pdf
        └── python_icon.png
```

Pelican expects articles to be located in the content folder by default. To tell Pelican where to find our articles add a new line to pelicanconf.py. The article path is built by combining the PATH and ARTICLE_PATHS variables into a file path.

```
ARTICLE_PATHS = ['articles']
```

We've changed the location of our articles so any links in our blog posts must be manually updated to the new path. Open the file *markdown_examples_2.md* and change the link to *first_post.md* as follows:

```
[First Post]({filename}/articles/first_post.md)
```

Next we will organize our blog's media files. Create two new folders: img and pdf within the content folder. Place the appropriate files in each folder as shown below.

```
blog
└── source
    └── content
        └── articles
        |   └── first_post.md
        |   └── markdown_example_1.md
        |   └── markdown_example_2.md
        |   └── markdown_example_3.md
        └── img
        |   └── python_icon.png
        └── pages
        |   └── about_me.md
        └── pdf
            └── pelican.pdf
```

Once again we must tell Pelican where to look for the static media files we just moved. Add the following code to pelicanconf.py.

```
```

```
STATIC_PATHS = ['img', 'pdf']
```

Open markdown_examples_2.md and change the path of the static files there as well.

```
![python logo]({static}/img/python_icon.png)
[Pelican Documenation]({static}/pdf/pelican.pdf)
```

Lastly, we have the option to define where Pelican should look for our blog's pages. By default Pelican expects them to be in the content/pages folder. It is not necessary to state the path but it is a good practice to do so. Add the following code to pelicanconf.py.

```
 PAGE_PATHS = ['pages']
```

The site and check that the changes we made do not raise any errors during the process. Our blog should output in exactly the same manner as before we started making alterations and no changes to the layout or content of our website should be evident.

```
pelican content
pelican --listen
```

Now that we've got the content folder organized let's turn our attention to the file structure that gets output when we generate our site.

Defining the URL Structure

We should aim to make the file structure of our generated website, and consequently the URL, as simple as possible. Logically constructed URLs that are readable by humans are more likely to be shared and to be successfully indexed by a search engine.

Let's use the post Markdown Examples 3 as an example. Browse to the post and you will see the URL below. The title of the post is showing as a slug after the domain.

```
http://127.0.0.1:8000/markdown-examples-part-3.html.
```

We should change the URL to show the content type and date as well. The ARTICLE_URL variable states what should display in the web browser's address bar while the ARTICLE_SAVE_AS variable defines where the article being generated should be output to. Add the following code to pelicanconf.py

```
ARTICLE_URL = 'articles/{date:%Y}/{date:%m}/{date:%d}/{slug}/'
ARTICLE_SAVE_AS = 'articles/{date:%Y}/{date:%m}/{date:%d}/{slug}/index.html'
```

Generate the webpage after making changes.

```
pelican content
pelican --listen
```

Now the article URL appears as:

```
http://127.0.0.1:8000/articles/2019/01/04/markdown-examples-part-3/
```

Go ahead do the same procedure for pages, categories, and tags. Add the following code to pelicanconf.py

```
PAGE_URL = 'pages/{slug}/'
PAGE_SAVE_AS = 'pages/{slug}/index.html'
CATEGORY_URL = 'category/{slug}'
CATEGORY_SAVE_AS = 'category/{slug}/index.html'
TAG_URL = 'tag/{slug}'
TAG_SAVE_AS = 'tag/{slug}/index.html'
```

Generate the blog again and review how the URLs have changed.

```
pelican content
pelican --listen
```