What's on this Page

CONTENT MANAGEMENT FUNDAMENTALS

Content Organization

Hugo assumes that the same structure that works to organize your source content is used to organize the rendered site.

Page Bundles

Hugo `0.32` announced page-relative images and other resources packaged into `Page Bundles`.

These terms are connected, and you also need to read about Page Resources and Image Processing to get the full picture.

The illustration shows three bundles. Note that the home page bundle cannot contain other content pages, although other files (images etc.) are allowed.

> The bundle documentation is a **work in progress**. We will publish more comprehensive docs about this soon.

Organization of Content Source

In Hugo, your content should be organized in a manner that reflects the rendered website.

While Hugo supports content nested at any level, the top levels (i.e. `content/<DIRECTORIES>`) are special in Hugo and are considered the content type used to determine layouts etc. To read more about sections, including how to nest them, see sections.

Without any additional configuration, the following will automatically work:

.

```
|   └── index.md  // <- https://example.com/about/
├── posts
|   ├── firstpost.md   // <- https://example.com/posts/firstpost/
|   ├── happy
|   |   └── ness.md  // <- https://example.com/posts/happy/ness/
|   └── secondpost.md  // <- https://example.com/posts/secondpost/
└── quote
    ├── first.md       // <- https://example.com/quote/first/
    └── second.md      // <- https://example.com/quote/second/
```

Path Breakdown in Hugo

The following demonstrates the relationships between your content organization and the output URL structure for your Hugo website when it renders. These examples assume you are using pretty URLs, which is the default behavior for Hugo. The examples also assume a key-value of `baseURL = "https://example.com"` in your site's configuration file.

Index Pages: `_index.md`

`_index.md` has a special role in Hugo. It allows you to add front matter and content to your list templates. These templates include those for section templates, taxonomy templates, taxonomy terms templates, and your homepage template.

> **Tip:** You can get a reference to the content and metadata in `_index.md` using the `.Site.GetPage` function.

You can create one `_index.md` for your homepage and one in each of your content sections, taxonomies, and taxonomy terms. The following shows typical placement of an `index.md` that would contain content and front matter for a `posts` section list page on

```
.         url
.       ⊢--^-⊣
.        path    slug
.       ⊢--^--⊩---^---⊣
.           filepath
.       ⊢------^------⊣
content/posts/_index.md
```

At build, this will output to the following destination with the associated values:

```
                     url ("/posts/")
                    ⊢-^-⊣
       baseurl      section ("posts")
⊢--------^---------⊩-^-⊣
         permalink
⊢----------^-------------⊣
https://example.com/posts/index.html
```

The sections can be nested as deeply as you want. The important thing to understand is that to make the section tree fully navigational, at least the lower-most section must include a content file. (i.e. `_index.md`).

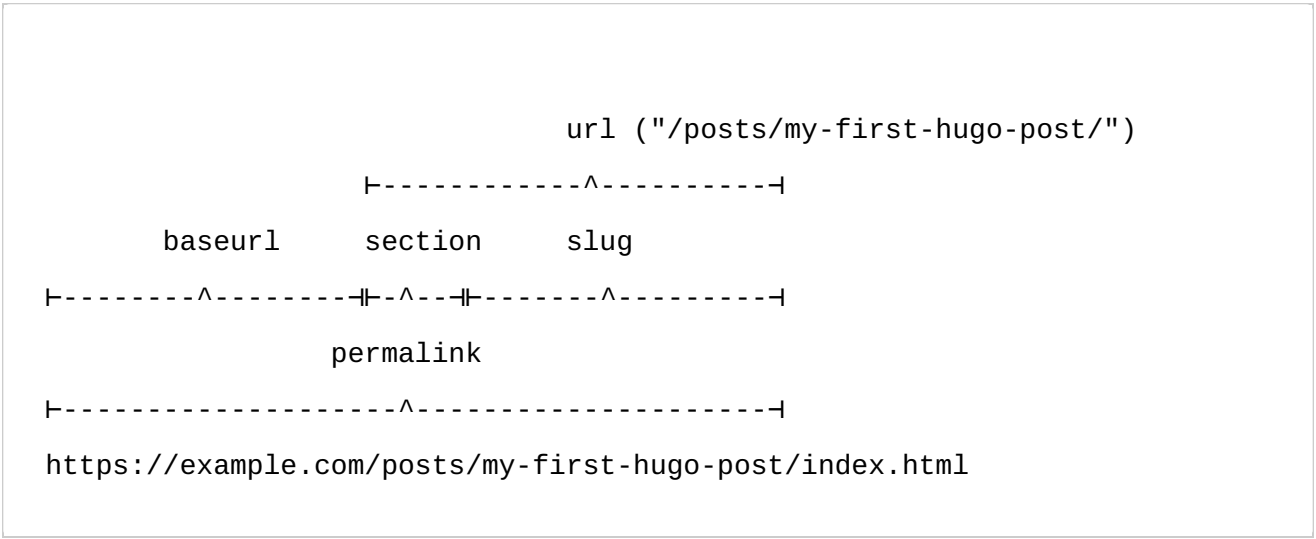
Single Pages in Sections

Single content files in each of your sections will be rendered as single page templates. Here is an example of a single `post` within `posts`:

```
                    path ("posts/my-first-hugo-post.md")
.       ⊢----------^------------⊣
```

```
content/posts/my-first-hugo-post.md
```

When Hugo builds your site, the content will be output to the following destination:



Paths Explained

The following concepts provide more insight into the relationship between your project's organization and the default Hugo behavior when building output for the website.

`section`

A default content type is determined by the section in which a content item is stored. `section` is determined by the location within the project's `content` directory. `section` *cannot* be specified or overridden in front matter.

`slug`

A content's `slug` is either `name.extension` or `name/`. The value for `slug` is determined by

- the name of the content file (e.g., `lollapalooza.md`) OR

`path`

A content's `path` is determined by the section's path to the file. The file `path`

- is based on the path to the content's location AND
- does not include the slug

`url`

The `url` is the relative URL for the piece of content. The `url`

- is based on the content item's location within the directory structure OR
- is defined in front matter, in which case it *overrides all the above*

Override Destination Paths via Front Matter

Hugo assumes that your content is organized with a purpose. The same structure that you use to organize your source content is used to organize the rendered site. As displayed above, the organization of the source content will be mirrored at the destination.

There are times when you may need more fine-grained control over the content organization. In such cases, the front matter field can be used to determine the destination of a specific piece of content.

The following items are defined in a specific order for a reason: items explained lower down in the list override higher items. Note that not all items can be defined in front matter.

`filename`

slug

When defined in the front matter, the `slug` can take the place of the filename in the destination.

content/posts/old-post.md Copy

```
---
title: A new post with the filename old-post.md
slug: "new-post"
---
```

This will render to the following destination according to Hugo's default behavior:

```
example.com/posts/new-post/
```

section

`section` is determined by a content item's location on disk and c*a*nnot be specified in the front matter. See sections for more information.

type

A content item's `type` is also determined by its location on disk but, unlike `section`, it c*a*n be specified in the front matter. See types. This can come in especially handy when you want a piece of content to render using a different layout. In the following example, you can create a layout at `layouts/new/mylayout.html` that Hugo will use to render this piece of content, even in the midst of many other posts.

content/posts/my-post.md Copy

```
---
```

```
layout: mylayout
---
```

`url`

A complete URL can be provided. This will override all the above as it pertains to the end destination. This must be the path from the baseURL (starting with a `/`). `url` will be used exactly as it is defined in the front matter, and will ignore the `--uglyURLs` setting in your site configuration:

`content/posts/old-url.md` Copy

```
---
title: Old URL
url: /blog/new-url/
---
```

Assuming your `baseURL` is configured to `https://example.com`, the addition of `url` to the front matter will make `old-url.md` render to the following destination:

```
https://example.com/blog/new-url/
```

You can see more information on how to control output paths in URL Management.

See Also

- Comments
- Page Resources
- Content Sections

Last updated: November 8, 2021: Update index.md (#1570) (5fbe741d7)

IMPROVE THIS PAGE

By the Hugo Authors

File an Issue Get Help Discuss Source Code

@GoHugoIO @spf13 @bepsays