Extracted from:

Build Websites with Hugo

Fast Web Development with Markdown

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Build Websites with Hugo

Fast Web Development with Markdown

Brian P. Hogan

edited by Tammy Coron

Build Websites with Hugo

Fast Web Development with Markdown

Brian P. Hogan

The Pragmatic Bookshelf

Raleigh, North Carolina

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Tammy Coron
Copy Editor: Jasmine Kwityn
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-726-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—May 2020

CHAPTER 4

Working with Data

Modern sites are driven by data. That data might come in the form of content stored in Markdown files, or from external APIs.

You've already used some data in your site. For example, you've included the site's title in your templates, and you've displayed each page's title. But you're not restricted to global configuration data. Hugo has built-in support for incorporating external data files into your site, and it can fetch data from remote sources to generate content in your layouts.

Using Site Configuration Data in Your Theme

Your layouts use {{ .Site.Title }} to display the name of the site in the browser title bar and in your header navigation. You can place all kinds of other data in your site's configuration file and use it as well.

For example, let's use the site's configuration to build additional <meta> tags in the site's header. The site configuration file has built-in fields like Title, but you can add anything you'd like to this file if you add it to a params section of the file.

Open config.toml and add a new params section that defines the author of the site and a description of the site:

working_with_data/portfolio/config.toml
```
[params]
  author = "Brian Hogan"
  description = "My portfolio site"
```

Then, open the file themes/basic/layouts/partials/head.html and add a new author meta tag to the page:

working_with_data/portfolio/themes/basic/layouts/partials/head.html
```
➤ <meta name="author" content="{{ .Site.Params.author }}">
  <link rel="stylesheet" href="{{ "css/style.css" | relURL }}">
```

Below that, add one for the description:

```
<meta name="description" content="{{ .Site.Params.description }}">
```

Now all of your pages will have these tags filled in. If you open your page in a browser and look at its source, you'll see the data:

```
...
  <title>Brian&#39;s Portfolio</title>
➤ <meta name="author" content="Brian Hogan">
➤ <meta name="description" content="My portfolio site">
  <link rel="stylesheet" href="/css/style.css">
...
```

The description field isn't page-specific yet, but you'll fix that shortly after you explore how to use front matter data to pull in content.

Adding Google Analytics

While we're on the subject of data, you might want to collect data about how people are interacting with your site. Hugo has built-in support for working with Google Analytics,[a] a platform from Google that lets you track unique page views, browser data, and other metrics from your site's visitors.

To incorporate this into your site, grab your Google Analytics tracking ID and add it to your site's configuration file using the googleAnalytics field:

```
    title = "Brian's Portfolio"
➤   googleAnalytics = "UA-XXXXXXXXXXXX"
```

Then, in your themes/basic/layouts/partials/head.html file, use Hugo's built-in template:

```
{{ template "_internal/google_analytics_async.html" . }}
```

When you generate your site, the appropriate JavaScript will be added to your pages, and it'll pull the tracking ID out of your site's configuration.

If you are going to use Google Analytics or other tracking software, be sure to consult local privacy laws, as you may need to obtain permission from your users to collect data.[b]

a. https://analytics.google.com/
b. https://www.cookiepro.com/knowledge/google-analytics-and-gdpr/

Populating Page Content Using Data in Front Matter

When you created your Project archetype, you created some placeholder content that you could replace. Over time, you might want to change the content on these pages, and going through each one manually to get them all looking the same could be a lot of work.

Let's refactor the project pages so they're more data-driven.

Modify the default archetype for Projects to include the image, the image's alternative text, and the "Tech Used" content. You'll also add a summary that you can use in various places:

working_with_data/portfolio/archetypes/projects.md

```
---
title: "{{ replace .Name "-" " " | title }}"
draft: false
image: //via.placeholder.com/640x150
alt_text: "{{ replace .Name "-" " " | title }} screenshot"
summary: "Summary of the {{ replace .Name "-" " " | title }} project"
tech_used:
- JavaScript
- CSS
- HTML
---

Description of the {{ replace .Name "-" " " | title }} project...
```

Notice that you're using the name of the file to generate some of the default text for the summary, the image alt text, and even the main content.

Generate a third project using this template. Stop your server and execute the following command to create a page for a project called "Linkitivity."

```
$ hugo new projects/linkitivity.md
/Users/brianhogan/portfolio/content/projects/linkitivity.md created
```

Open content/projects/linkitivity.md and you'll see this content:

working_with_data/portfolio/content/projects/linkitivity.md

```
---
title: "Linkitivity"
draft: false
image: //via.placeholder.com/640x150
alt_text: "Linkitivity screenshot"
summary: "Summary of the Linkitivity project"
tech_used:
- JavaScript
- CSS
- HTML
---

Description of Linkitivity project...
```

The name "Linkitivity" is placed throughout the front matter, making it easier for you to maintain consistency.

Before moving on, modify the two existing project pages you created to reflect these changes. Remove the placeholder text and move it all to the front matter section instead, under those keys. For reference, here's what your Awesomeco project should look like:

working_with_data/portfolio/content/projects/awesomeco.md

```
---
title: "Awesomeco"
draft: false
image: //via.placeholder.com/640x150
alt_text: "Awesomeco screenshot"
summary: "Summary of the AwesomeCo project"
tech_used:
- JavaScript
- CSS
- HTML
---

Description of the Awesomeco project...
```

Next, modify the project layout to use this data. Open themes/basic/layouts/projects/single.html and locate the {{ .Content }} section. Below that line, add code that adds the image and alternative text from the front matter, and then iterates through the values in the tech used field to display those values on the page:

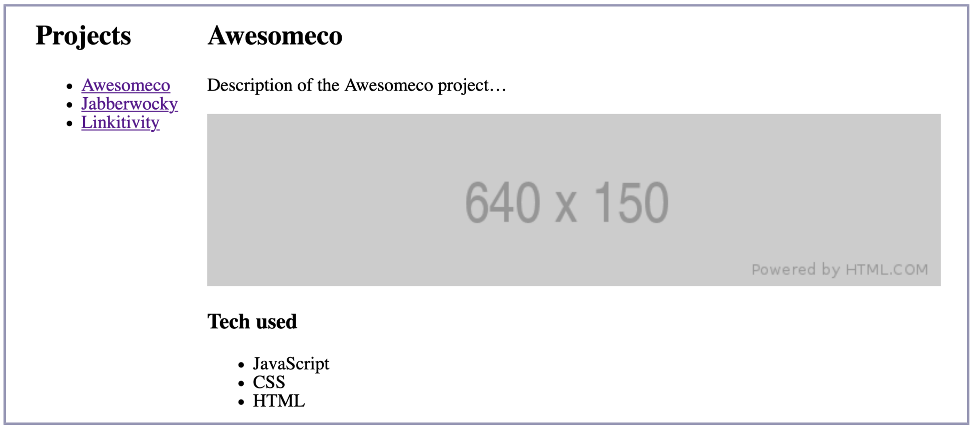
working_with_data/portfolio/themes/basic/layouts/projects/single.html

```
<section class="project">
  <h2>{{ .Title }}</h2>

  {{ .Content }}
➤ <img alt="{{ .Params.alt_text }}" src="{{ .Params.image }}">
➤
➤ <h3>Tech used</h3>
➤ <ul>
➤   {{ range .Params.tech_used }}
➤     <li>{{ . }}</li>
➤   {{ end }}
➤ </ul>

</section>
```

Notice that the image, alt_text, and tech_used fields are prefixed by .Params. Any custom fields you add to the front matter get added to this .Params collection. If you don't add that prefix, Hugo won't be able to find the data. Fields like description and title are predefined fields that Hugo knows about, so you don't need the params prefix when referencing them. You can find the list of predefined fields in Hugo's documentation.[1]

1. https://gohugo.io/content-management/front-matter/#predefined

Save the layout and start the development server with hugo server. Visit http://localhost:1313/projects/awesomeco.md and your project displays, but this time it's driven by front matter data instead of hard-coded content:


Projects

- Awesomeco
- Jabberwocky
- Linkitivity

Awesomeco

Description of the Awesomeco project…

Tech used

- JavaScript
- CSS
- HTML