Newer Post
Being Productive on the Command Line

Older Post
How to Unleash Yourself in Just 2 Simple Steps

Alberta Williams Jul 24, 2017

Build a Website With Jekyll and GitHub Pages

If you would like to publish a personal website, start a blog, or host any other static website, Jekyll and GitHub Pages offer a minimalistic solution. Other options for building a website are website builders like SquareSpace or content management systems like Wordpress. Both provide templates to design your site and require you pay for hosting. The other option is to build your own content management system. But if you are lazy like me, that is not even an option. However, if you just need a site to show off your portfolio or projects, I recommend using Jekyll to build the site and host it online for free using GitHub pages.

Table of Contents

Getting started with Jekyll

Jekyll is a command line tool that will generate our pages for us. Earlier, I told you we would just be using Jekyll to build the site. However, that was a lie. You will need other software installed on your computer in order to use Jekyll. Before you get started, fix yourself a cup of hot tea and turn on some soothing music. I like to listen to the peaceful piano playlist on Spotify. It is important not to omit this step, because this is the most pleasurable part of the process.

To use Jekyll, we need Ruby and RubyGems. Jekyll is written in Ruby and RubyGems is a package manager for Ruby. To install Ruby and RubyGems, we will use a tool called Homebrew. Homebrew is a command line tool that allows us to install programs on a Mac. (Note: you may have to install these programs as the root user.) (Another note: sorry, Windows users. No instructions for you.)

Install Homebrew:

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew
/install/master/install)"
```

Install Ruby:

```
$ brew install ruby
```

Next, install Jekyll and Bundler:

```
$ gem install jekyll bundler
```

Create a Jekyll site named `blog`:

```
$ jekyll new blog
```

Change into the directory:

SHARES

```
$ cd blog
```

Build the site and watch for changes:

```
$ bundle exec jekyll serve
```

Navigate to http://localhost:4000 in your browser to see your site. If everything is fine, you should see a welcome page. Now, let us take a step back and look at the files in our project. When we entered the `jekyll new blog` command, it created the following folders and files in our directory:

```
|-- _config.yml
|-- _posts/
|-- _site/
|-- 404.html
|-- about.md
|-- Gemfile
|-- Gemfile.lock
|_  index.md
```

`_config.yml` - The configuration settings for our site

`Gemfile` - A list of Gem dependencies

`Gemfile.lock` - A list of Gems AKA packages for your project

`_posts` - Directory for blog posts

`_site` - Directory where files are written

`404.html` - Error page

`about.md` - About page

`index.md` - Home page

SHARES

Customizing the site

Our site came with some default text. You can start by changing the title and description. These values have been set in the `_config.yml` file. When we restart our Jekyll server and refresh the page, the text in the header and footer will have changed.

Notice also that our site is already styled. This is because it uses a theme named Minima. The theme's files were installed on your computer when you installed Jekyll. To override the theme's styles, we need to copy the files to our project folder. You can see the file path of the theme by running the command `bundle show minima`.

To get started customizing our site, copy all of the folders inside the minima directory to the project directory. Next, remove the reference to `gem "minima", "~> 2.0"` from the `Gemfile`, and remove `theme: minima` from `_config.yml`. This removes the Gem theme and lets us use our own theme. Restart the server to register the changes we made.

Layouts

Files in the `_layouts` directory contain templates for creating pages. The purpose of the layouts is to abstract away the structure of the pages from the content. When you create a page, like `about.md`, the content that is inside will be injected into the layout where `{{ content }}` is specified. We can also create a layout that inherits another layout. For example, the `page.html` is based on the default layout.

Let us add another page to our site. Create a file named `projects.md`, and save it to the root of your directory. Add the following inside:

```
---
layout: page
title: Projects
permalink: /projects/
---
```

This is called the front matter and tells Jekyll how to build the page. This

SHARES

page will use the `page.html` template from the `_includes` directory. Where the `page.title` variable is used in the template, it will be replaced with the text "Projects". And permalink is the relative path for the page. For demonstration purposes, we can add some example text below the front matter. When you refresh your browser, you will see the new projects page. Add as many pages as you like.

Includes

The templates for our header, footer, and head of our site are located in the `_includes` directory. Includes are useful for separating sections of your pages that will be used in multiple places. You create the template and include it in your layout or pages with `{% include file_name.html %}`, where `file_name` is replaced with the name of the file.

Posts

The `_posts` folder contains our blog posts. To create a new post, create a file with the format `yyyy-mm-dd-title-of-post.ext`. For example: `2017-07-06-build-a-website.md`. Add the following front matter to the file:

```
---
layout: post
title:  "Build a Website"
date:   2017-07-06 06:18:08
author: Alberta Williams
permalink: blog/build-a-website.html
---
```

Since our example post used an `md` extension, we can write our post using Markdown. When we refresh the browser, we will see the post added to the front page. Unlike in the example post provided, we added an author variable and a permalink. If a permalink had not been specified, the filename would be used as the path. Other page variables you can specify for posts include tags and categories.

Sass

The [illegible] directory contains the styles for the site. Sass is a CSS

SHARES

framework for creating stylesheets. Sass files use a `.scss` extension. Files that begin with an underscore, such as `_base.scss`, are partials and can be imported in your main sass file. We can also define variables to replace hard-coded values. Variables begin with a `$`. When your site is built, these sass files are automatically converted into CSS.

Assets

The `assets` folder is where you will store your images, fonts, scripts, and styles. The minima theme includes a `main.scss` file. Below the import statement, we can add custom styles. Alternatively, we can just create a `main.css` file to add our styles. You may want to consider using a framework like Bootstrap for designing your site, if you have no interest in writing CSS.

Deploying

We will be using GitHub Pages to host our site. I like Github Pages because I can use git to update my site with each commit I push. To get started, we will need to have a github account and have git installed on our computer. In your dashboard, click the link to create a new repository and name it username.github.io where username is replaced with the username of your account.

From your terminal inside the root of your project, initialize git:

```
$ git init
```

Add all of the files:

```
$ git add --all
```

Save the changes:

```
$ git commit -m "first commit"
```

SHARES

Set up the remote repository to push to:

```
$ git remote add origin https://github.com/username/repo_name.git
```

The url will be replaced with the url you were given when you created the repository. Finally, push your changes to the remote repository:

```
$ git push -u origin master
```

You can now see your site online at [http://username.github.io].

Conclusion

We saw how we can use Jekyll to build a website. Jekyll is especially useful, if you intend to create a simple blog, because adding posts just requires saving the file to the folder. Our site used a theme that gave us default pages and styling. However, you can create your own theme by copying the same files to your project and adding your own CSS. Finally, we deployed our site, by pushing it to our GitHub repository. You should now be able to publish your own website.

Resources

- Homebrew
- Ruby
- RubyGems
- Bundler
- Jekyll
- Minima Jekyll theme
- Bootstrap
- Git
- GitHub Pages
- Customizing GitHub Pages

SHARES

2 Comments X-Team Login

Recommend 1 Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Anelda van der Walt • a month ago

Hi there, thanks for the post. Just to note that there's a typo just underneath "bundle exec jekyll serve". The site is available at http://localhost:4000 (there's a 0 missing in your instructions).

• Reply • Share ›

Alberta Williams Anelda van der Walt • a month ago

Thanks for letting me know! The post will be updated.

• Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy

SHARES

SCALE YOUR DEVELOPMENT TEAM

We help you execute projects by providing trusted developers who can join your team and immediately start delivering high-quality code.

HIRE DEVELOPERS

code 155 community 151 remoteworking 54 drupal 39 javascript 29
unleash 27 react 20 superheroes 18 productivity 11 mobile 9

Show All ...

X-Team · Commit.Push.Unleash **TOP**

SHARES