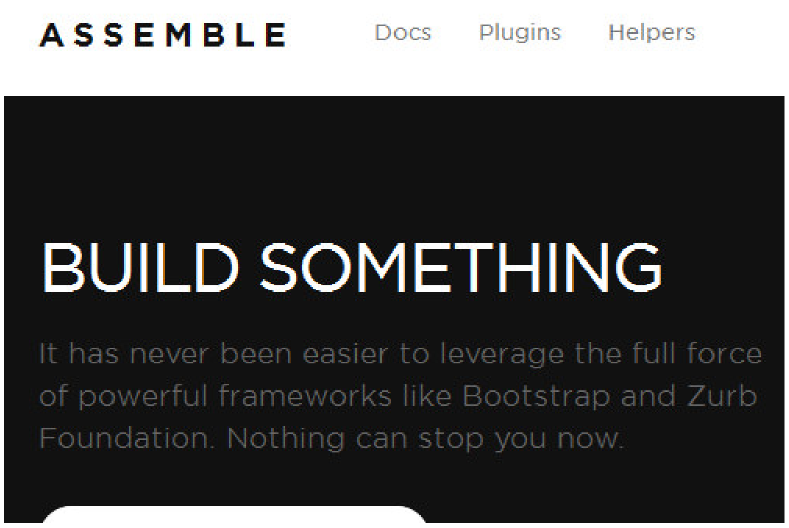
Agus (Http://Www.Hongkiat.Com/Blog/Author/Agus/) — Blogging (Http://Www.Hongkiat.Com/Blog/Category/Blogging/)

How To Build A Static Blog Using Assemble

Today, we are going to take a look at **Assemble (http://assemble.io/), a Grunt plugin that allows us create and manage static sites with ease**. Assemble may be slightly similar to Jekyll (http://www.hongkiat.com/blog/tag/jekyll/), but it brings more flexibility and features to the table that makes it more powerful.


ASSEMBLE
Docs
Plugins
Helpers
BUILD SOMETHING
It has never been easier to leverage the full force of powerful frameworks like Bootstrap and Zurb Foundation. Nothing can stop you now.

Permalink, Bootstrap Boilerplates, and LESS compiler are the features that makes Assemble a comparable tool to a full-fledged CMS application. Herein, we will show you **how to use Assemble to create a static blog**.

> **RECOMMENDED READING:**
> How To Unload Unnecessary CSS With Grunt (http://www.hongkiat.com/blog/unload-unnecessary-css/)

Step 1. Installing Project Dependency

Assemble requires Grunt to function (refer to our previous posts on Node.js (http://www.hongkiat.com/blog/node-js-server-side-javascript/) and Grunt (http://www.hongkiat.com/blog/tag/grunt/) if you need further assistance). Then, once Node and Grunt are all set, create a `package.json` file in the project folder to specify the Node packages that we will employ to build our blog.

Add the following code in package.json:

```
{
  "devDependencies": {
    "assemble": "~0.4.40",
    "grunt": "~0.4.5",
    "grunt-contrib-connect": "~0.8.0",
    "grunt-contrib-watch": "^0.6.1"
  }
}
```

These lines of code in package.json tells Node that our project will be dependent on Grunt (http://gruntjs.com/), Grunt Connect (https://github.com/gruntjs/grunt-contrib-connect), Grunt Watch (https://github.com/gruntjs/grunt-contrib-watch) and Assemble (http://assemble.io/). Now, we will install these packages by running this command via the Terminal.

5 Creative & Unique Ways People Document Their Travels (http://www.hongkiat.com/blog/travels-photography-documentation/)

10 Comic Themed Posters To Teach You Life Lessons (http://www.hongkiat.com/blog/life-lessons-posters/)

```
npm install
```

Step 2. Load and Register Grunt Tasks

After all the dependencies are downloaded, create **`grunfile.js`** and put the following lines in:

```
module.exports = function(grunt) {
    grunt.initConfig({
        pkg: grunt.file.readJSON('package.json')
    });

    grunt.loadNpmTasks('assemble');
    grunt.loadNpmTasks('grunt-contrib-connect');
    grunt.loadNpmTasks('grunt-contrib-watch');

    grunt.registerTask('default', ['connect:livereload','assemble','watch']);
};
```

The lines we put in gruntfile.js above merely **load and register the dependencies that we have just downloaded** through the **`npm install`** command. We will make these tasks “work” later in the following steps.

Step 3. Folder and File Structure

We will now **organize the folder and file structure of our blog**, as follows:

```
MyBlog/
    package.json
    gruntfile.js
    app/
        layout/
            default.hbs
        content/
            page/
                index.hbs
            blog/
                first-posting.hbs
        partials/
```

Assemble allows us to configure the file and directory organization through the gruntfile.js. But, for now, let's just keep up with the default configuration, as shown above.

Step 4. The Blog Layout

In Assemble, **Layouts set the foundation of a page**. In Step 3, we have created a layout file named `default.hbs` in the `MyBlog/app/layout/` folder. The `.hbs` extension is used because Assemble uses the Handlebars (http://handlebarsjs.com) templating language.

REAS ALSO:

A Look Into: Handlebars.Js (http://www.hongkiat.com/blog/a-look-into-handlebarsjs/)

The `default.hbs` will be used by all pages in the blog which refers to this file. Herein, we will use Bootstrap (http://getbootstrap.com/) via the BootstrapCDN (http://bootstrapcdn.com) to set the styling base for our blog. We then add in the following codes in `default.hbs` :

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My Blog</title>
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/b
</head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h1 class="page-header text-center">MY BLOG</h1>
            </div>
            <div class="col-md-9 main">
            {{> body }}
            </div>
        </div>
    </div>
</body>

</html>
```

Step 5. Configuring the Grunt Tasks

As the next step, create a `Gruntfile.js` to **configure directories and files for Assemble to compile**. Open `Gruntfile.js` and add the following codes in the `Grunt.initConfig` section:

```
grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    watch: {
        assemble: {
            files: [
                'app/content/blog/*.hbs',
                'app/content/pages/*.hbs',
                'app/layouts/*.hbs',
                'app/partials/*.hbs'
            ],
            tasks: ['assemble']
        },
        livereload: {
    options: {
      livereload: '<%= connect.options.livereload %>'
    },
    files: [
      './dist/*.html'
    ]
  },
    },
    assemble: {
        options:{
            layoutdir: 'app/layouts',
            flatten: true,
            layout: 'default.hbs',
            partials: 'app/partials/*.hbs'
        },
        page: {
            files: {
                'dist/': ['app/content/page/*.hbs']
            }
        },
        blog: {
            files: {
                'dist/': ['app/content/blog/*.hbs']
            }
        }
    },
connect: {
  options: {
    port: 8800,
    // change this to '0.0.0.0' to access the server from outside
    hostname: 'localhost',
    livereload: 35728
  },
  livereload: {
    options: {
      open: true,
      base: './dist'
    }
  }
}
});
```

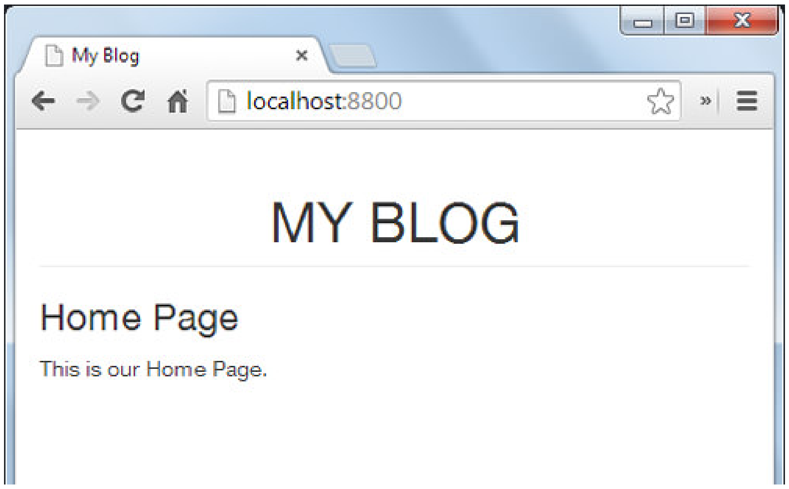
Step 6. Generating Page and First Post

We can now **build a page**. Let's open index.hbs file in `MyBlog/app/content/page/` folder and add the content.

```html
<h3>Home Page</h3>

<section>
<p>This is our Home Page. </p>
</section>
```

Through the Command Prompt or Terminal, run `grunt` command. This command will generate the `index.hbs` file into a `html` file and immediately launch the file in the browser. Let's look at the result in the browser.



MY BLOG

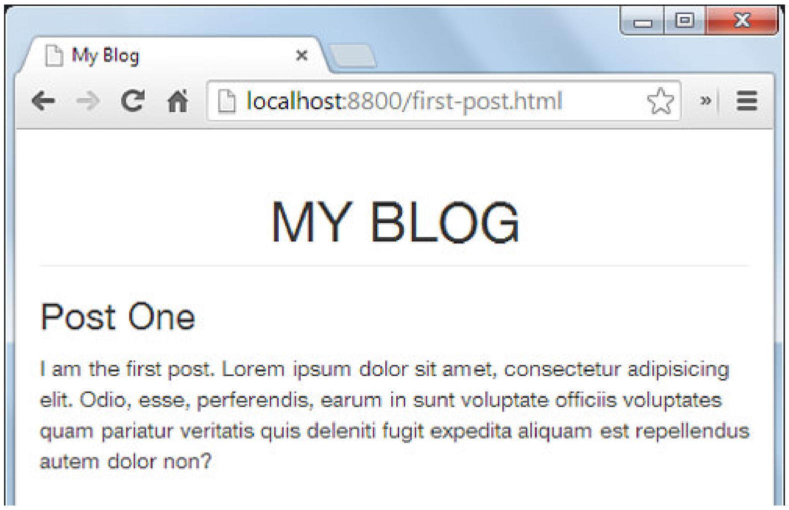
Home Page

This is our Home Page.

We will also **generate the first post** of our blog. Open the `first-post.hbs` inside the `MyBlog/app/content/blog/` folder and lay out the content, like so.

```html
<h3>First Post</h3>
<section>
<p>I am the first post. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
</section>
```

Once again run the `grunt` command and you will see the `first-post.html` file generated in a newly created folder named `dist` . Navigate to `localhost:8800/first-post.html` on the browser, you should find the first post to be the same as the image below.



MY BLOG

Post One

I am the first post. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Odio, esse, perferendis, earum in sunt voluptate officiis voluptates quam pariatur veritatis quis deleniti fugit expedita aliquam est repellendus autem dolor non?

You can create more posts by creating more **`.hbs`** files and place them inside in the **`MyBlog/app/content/blog/`** folder.

Step 7. Create a List of Blog Posts

Now, we will create a list of posts and put it **in the blog sidebar**. To do so, we will use the **Partial** feature of Assemble. A “Partial” is a reusable fragment of codes that can be included into the other pages.

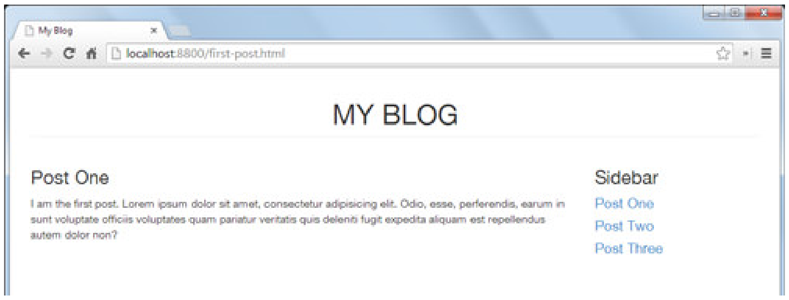
The Sidebar is meant to contain a list of our blog posts as well as the link to the respective post. Let’s make a new file named **`sidebar.hbs`**. Add the following code in and save it inside the **`MyBlog/app/partials/`** folder.

```
<h3>Sidebar</h3>
{{#each pages}}
<li class="list-unstyled">
    <a href="{{relative dest this.dest}}">{{ data.title }}</a>
</li>
{{/each}}
```

Then, call the Sidebar partial in **`default.hbs`**, as follows:

```
<div class="col-md-3 sidebar">
{{> sidebar }}
</div>
```

The **`#each`** is a loop that will list all of our blog posts in **`MyBlog/app/content/blog/`** folder. The result is shown below:



MY BLOG

Post One

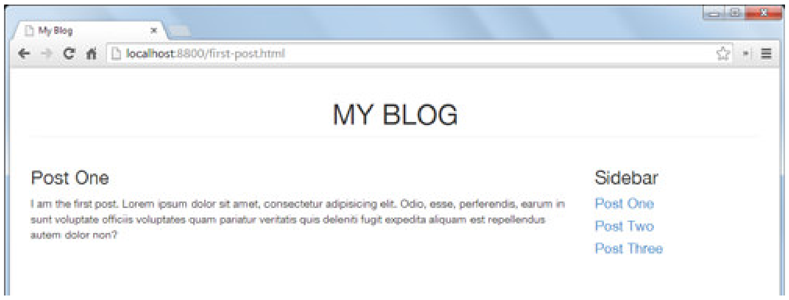
I am the first post. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Odio, esse, perferendis, earum in sunt voluptate officiis voluptates quam pariatur veritatis quis deleniti fugit expedita aliquam est repellendus autem dolor non?

Sidebar

Post One
Post Two
Post Three

Step 8. Using Variables

With Assemble, we can use a variable using YAML front matter. YFM (YAML front matter) is **an optional section that is placed at the top of a page and is used for maintaining metadata for the page and its contents**. We will use it to specify the post title; open `first-post.hbs` , and modify the code like so:



MY BLOG

Post One

I am the first post. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Odio, esse, perferendis, earum in sunt voluptate officiis voluptates quam pariatur veritatis quis deleniti fugit expedita aliquam est repellendus autem dolor non?

Sidebar

Post One
Post Two
Post Three

```
---
title: Post One
---

<h3>{{ title }}</h3>
<section>
blahblah...
</section>
```

The `{{title}}` tag will be filled with “Post One” that we’ve defined on top.

Step 9. Ordering list of posts

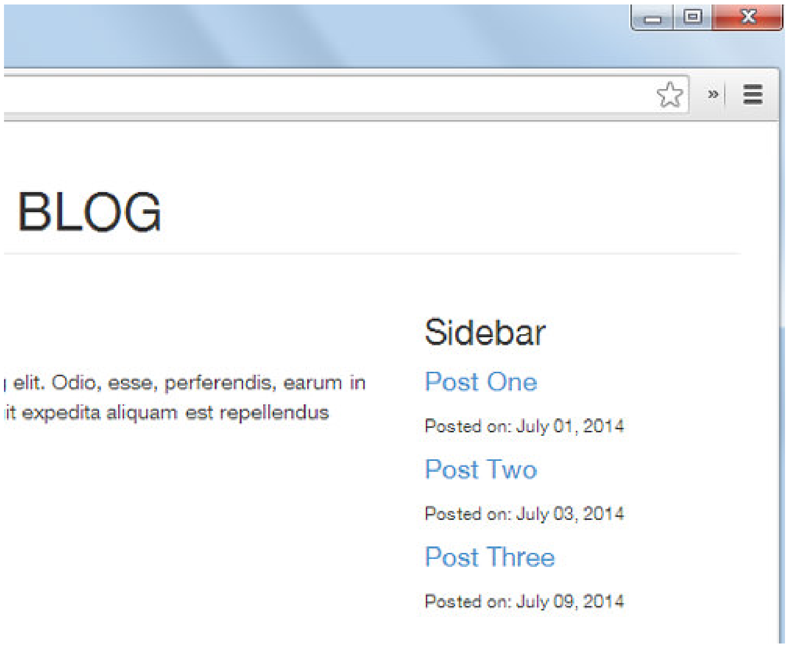
Assemble allows us to order and sort the list of post based on the ‘term’ specified. As an example, here we will order our blog posts on sidebar by the date. Let’s modify our post by adding date on YML front matter like below:

```
---
title: Post One
date: 2014-07-10
---
```

Also modify other post files in `MyBlog/app/content/blog/` . Then, on the `sidebar.hbs` , we will display the date below the post title. Modify the code like this:

```
<ul class="list-unstyled">
{{#withSort pages "data.title"}}
    <li>
        <h4><a href="{{relative dest this.dest}}">{{ data.title }}</a></h4>
        <small>Posted on: {{formatDate data.date "%B %d, %Y"}}</small>
    </li>
{{/withSort}}
</ul>
```

The result is the post list in the sidebar which is ordered by date.



BLOG

elit. Odio, esse, perferendis, earum in
it expedita aliquam est repellendus

Sidebar

Post One

Posted on: July 01, 2014

Post Two

Posted on: July 03, 2014

Post Three

Posted on: July 09, 2014

Conclusion

Now we have a simple blog generated with Assemble. **Assemble can be used as an alternative tool to build websites** as we've already shown you. And should you want to, you can use a free web hosting service like Github Pages or servers that support Node.js like Heroku (https://www.heroku.com/) to put your site online.

Readers also read:

Writing Content That Convert Readers & Deliver Sales (http://www.hongkiat.com/blog/writing-content-that-converts/)

Turn Your Blog posts Into A Video Easily With Wibbitz (http://www.hongkiat.com/blog/blog-posts-into-video/)

Top Tips & Resources To Creating Content People Want To Share (http://www.hongkiat.com/blog/create-viral-content/)