**Website Page Routing with Node.js Express and EJS**

[Express](https://expressjs.com/) is pretty much the default framework to create a website with Node. Just like any website creation, the first step to create an Express website is to set up an template and routing. [EJS](http://ejs.co/) is an easy templating language to generate HTML with Javascript. It works well with Node without having to learn more complicated front-end frameworks like Angular or React. I started web development with PHP. EJS feels a lot more like PHP and learning how to use was much easier for me.

Another important component in Express is the Router class which is a middleware to make url path-based routing easy and maintainable.

Aim here is to create a 3-page websites with home, about and contact pages with header, nav and footer templates and routing traffic by the express server. Let’s get started!

**Project Folder**

We first create a main project folder and run npm init -y. Then, organise it as below.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

**Packages**

Dependencies are below. For the front-end, we will use bootstrap 4 which requires jquery and popper.js. The ejs package for creating template.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

**URL Routing**

We are using the Router class. This enable us to export the routing logic as a module and use it in the main app.js. There are a few different ways for routing (see documentation [here](https://expressjs.com/en/guide/routing.html)). When the logic gets complicated, using the Router class to compartmentalise routing makes it easier to maintain.

**index.js in the routes folder**

<pre class=" toolbar:2 lang:js decode:true" >

</pre>

**app.js in the main project folder**

To serve static files, we need to use the express.static. By default, express cannot access to the files in other folders. We also need to set the default views directory where we have ejs files. The router gets imported and used as middleware.

<pre class=" toolbar:2 lang:js decode:true" >

</pre>

**Serving Bootstrap and Javascripts files**

Once we set the folders to serve static files by using the express.static, the path to the css or javascript files should not include the folder path. If you do, you will get the horrible MIME type error as below:

**Refused to apply style from whatever-the-file-path because its MIME type (‘text/html’) is not a supported stylesheet MIME type, and strict MIME checking is enabled**).

You can replicate this error by changing the stylesheet path from href=”bootstrap/dist/css/bootstrap.min.css” to href=”../node\_modules/bootstrap/dist/css/bootstrap.min.css” in the head.ejs file for the link element.

**Using Templates with EJS**

This is the simplest part. Simply use the ejs syntax which is very similar to including templates with PHP.

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

This is pretty much it. Now we can create the front-end view in whatever way you want by writing html and css.

Here is the html and css example of simple 3 page websites. Note that we are using bootstrap 4 for most of the styling with a few custom css.

**HTML**

**index.ejs**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**about.ejs**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**contact.ejs**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**head.ejs in the include folder**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**nav.ejs in the include folder**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**footer.ejs in the include folder**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**scripts.ejs in the include folder**

[cc lang="html5" tab\_size="4" lines="-1"]

[/cc]

**CSS**

**style.css**

[cc lang="css" tab\_size="4" lines="-1"]

[/cc]