


Lab 6

Due Oct 31, 2016 by 5pm **Points** 100 **Submitting** a file upload

CS-546 Lab 6

HTML and Templating

For this lab, you will be templating some basic data. You will download the [lab 6.zip](#)  zip file that will be provided via Canvas to use that codebase.

There are several things that I want for you to notice about the repository:

- The files in your data module do not access database calls, but still return promises; despite being a simple read-only application, I want you to understand what the workflow *will* be once you begin fitting databases back into the equation
- The middlewares to setup static assets, as well as the view engine have been setup for you.
- I have provided a handy helper that is **not** part of handlebars by default: `asJSON` which will print out the template variables as a JSON string

Objectives

1. Each route in the *routes* folder is setup to simply pass nonsense data to a *debug* view that will print out the data for easy debugging purposes. You will first acquire the associated data needed (detailed in the comments for each route) using the provided data modules. You will then make a view for each route and change the route from passing data to the *debug* view to the new view; the files can be named as per your choosing, but I personally recommend a directory for each route and some standardized convention (ie, index for when you are listing all objects of a certain data type).
 1. **Your route will specify what specifically must be included for that page**
 2. You can lay the content out using any tags that make semantical sense.
 3. You must print out all properties of the model that are passed down to the template, in some way. You can add any amount of text or other content to the page, as well.
2. You will create a static 404 error page; any url that is not matched by another route will show this 404 page and issue a 404 status code. You can setup this behavior easily in your `routes/index.js` file, which provides a small hint hint.
3. You will add 10+ CSS rules to the `public/styles/main.css` file that will style the HTML you have created using a combination of classes and element selectors (you can use more than that, but must at least use several class / element selectors).

You must save all dependencies to your package.json file

References and Packages

Basic CSS info can easily be referenced in the [MDN CSS tutorial](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started) (https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started). If you need a quick CSS reference,

You will use the **express-handlebars** package as your templating engine.

You can reference the [express-handlebars repository](https://github.com/ericf/express-handlebars/) (<https://github.com/ericf/express-handlebars/>) for details on adding the module; you may also want to check out the [handlebars website](http://handlebarsjs.com/) (<http://handlebarsjs.com/>) for details.

You will use the **express** package as your server.

You can read up on [express](http://expressjs.com/) (<http://expressjs.com/>) on its home page. Specifically, you may find the [API Guide section on requests](http://expressjs.com/en/4x/api.html#req) (<http://expressjs.com/en/4x/api.html#req>) useful.

You may use the [lecture 4 code](https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_4) (https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_4) as a guide.

You may use the [lecture 5 code](https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_5) (https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_5) as a guide.

You may use the [lecture 6 code](https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_6) (https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_6) as a guide.

You may use the [lecture 8 code](https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_8) (https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_8) as a guide.

Requirements

1. You **must not submit** your node_modules folder
2. You **must remember** to save your dependencies to your package.json folder
3. You must do basic error checking in each function
 1. Check for arguments existing and of proper type.
 2. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)
 3. If a function should return a promise, instead of throwing you should return a rejected promise.
4. You **must remember** to update your package.json file to set `app.js` as your starting script!
5. **Your HTML must be valid** (https://validator.w3.org/#validate_by_input) or you will lose points on the assignment.
6. Your HTML must make semantical sense; usage of tags for the purpose of simply changing the style of elements (such as `i`, `b`, `font`, `center`, etc) will result in points being deducted; think in terms of content first, then style with your CSS.
7. **You can be as creative as you'd like to fulfill front-end requirements**; if an implementation is not explicitly stated, however you go about it is fine (provided the HTML is valid and semantical). Design is not a factor in this course.
8. **You cannot update the data modules at all.**