Week of 14 Homework Node Express Handlebars

Overview

In this assignment, you'll create a burger logger with MySQL, Node, Express, Handlebars and a homemade ORM (yum!). Be sure to follow the MVC design pattern; use Node and MySQL to query and route data in your app, and Handlebars to generate your HTML.

Remember

You will be fully capable of doing this homework by the end of Saturday's class.

Before You Begin

- Eat-Da-Burger! is a restaurant app that lets users input the names of burgers they'd like to eat.
- Whenever a user submits a burger's name, your app will display the burger on the left side of the page -- waiting to be devoured.
- Each burger in the waiting area also has a `Devour it!` button. When the user clicks it, the burger will move to the right side of the page.
- Your app will store every burger in a database, whether devoured or not.
- Check out this video of the app for a run-through of how it works.

Instructions

App Setup

1. Create a GitHub repo called `burger` and clone it to your computer.
2. Make a package.json file by running `npm init` from the command line.
3. Install the Express npm package: `npm install express --save`.
4. Create a server.js file.
5. Install the Handlebars npm package: `npm install express-handlebars --save`.
6. Install the method-override npm package: `npm install method-override --save`.
7. Install the body-parser npm package: `npm install body-parser --save`.
8. Install MySQL npm package: `npm install mysql --save`.
9. Require the following npm packages inside of the server.js file:
   - express

    - •method-override
    - •body-parser

DB Setup

1.Inside your `burger` directory, create a folder named `db`.

2.In the `db` folder, create a file named `schema.sql`. Write SQL queries this file that do the following:

•Create the `burgers_db`.

•Switch to or use the `burgers_db`.

•Create a `burgers` table with these fields:

- •**id**: an auto incrementing int that serves as the primary key.
- •**burger_name**: a string.
- •**devoured**: a boolean.
- •**date**: a TIMESTAMP.

1.Still in the `db` folder, create a `seeds.sql` file. In this file, write insert queries to populate the `burgers` table with at least three entries.

2.Run the `schema.sql` and `seeds.sql` files into the mysql server from the command line

3.Now you're going to run these SQL files.
•Make sure you're in the `db` folder of your app.

•Start MySQL command line tool and login: `mysql -u root -p`.

•With the `mysql>` command line tool running, enter the command `source schema.sql`. This will run your schema file and all of the queries in it -- in other words, you'll be creating your database.

•Now insert the entries you defined in `seeds.sql` by running the file: `source seeds.sql`.

•Close out of the MySQL command line tool: `exit`.

Config Setup

1.Inside your `burger` directory, create a folder named `config`.

2.Create a `connection.js` file inside `config` directory.

•Inside the `connection.js` file, setup the code to connect Node to MySQL.

•Export the connection.
1.Create an `orm.js` file inside `config` directory.

- Import (require) `connection.js` into `orm.js`
- In the `orm.js` file, create the methods that will execute the necessary MySQL commands in the controllers. These are the methods you will need to use in order to retrieve and store data in your database.
    - `selectAll()`
    - `insertOne()`
    - `updateOne()`
- Export the ORM object in `module.exports`.

Model setup

- Inside your `burger` directory, create a folder named `models`.
    - In `models`, make a `burger.js` file.
    - Inside `burger.js`, import `orm.js` into `burger.js`
    - Also inside `burger.js`, create the code that will call the ORM functions using burger specific input for the ORM.
    - Export at the end of the `burger.js` file.

Controller setup

1. Inside your `burger` directory, create a folder named `controllers`.
2. In `controllers`, create the `burgers_controller.js` file.
3. Inside the `burgers_controller.js` file, import the following:

- Express
- `burger.js`

1. Create the `router` for the app, and export the `router` at the end of your file.

View setup

1. Inside your `burger` directory, create a folder named `views`.

- Create the `index.handlebars` file inside `views` directory.
- Create the `layouts` directory inside `views` directory.
    - Create the `main.handlebars` file inside `layouts` directory.
    - Setup the `main.handlebars` file so it's able to be used by Handlebars.
    - Setup the `index.handlebars` to have the template that Handlebars can render onto.

•Create a button in `index.handlebars` that will submit the user input into the database.

Directory structure

All the recommended files and directories from the steps above should look like the following structure:

```
.
├── config
│   ├── connection.js
│   └── orm.js
│ 
├── controllers
│   └── burgers_controller.js
│
├── db
│   ├── schema.sql
│   └── seeds.sql
│
├── models
│   └── burger.js
│ 
├── node_modules
│ 
├── package.json
│
├── public
│   ├── assets
│   │   ├── css
│   │   │   └── burger_style.css
│   │   └── img
│   │       └── burger.png
│   └── test.html
│
├── server.js
│
└── views
    ├── index.handlebars
    └── layouts
        └── main.handlebars
```

One More Thing

This is a really tough homework assignment, but we want you to put in your best effort to finish it.

If you have any questions about this project or the material we have covered, please post them in the community channels in slack so that your fellow developers can help you! If you're still having trouble, you can come to office hours for assistance from your instructor and TAs.
Good Luck!