WEB700 Assignment 5

Submission Deadline:Saturday, March 23rd, 2024 @ 11:59 PM

Assessment Weight:

9% of your final course Grade

Objective:

Build upon the code created in Assignment 4 by incorporating the Handlebars view engine to render our JSONdata visually in the browser using. hbs views and layouts. Additionally, update our collegeData module toallow for courses to be viewed individually using a (read only) web form

NOTE: If you are unable to start this assignment because Assignment 4 was incomplete - email your professorfor a clean version of the Assignment 4 files to start from. Please note: the "home", "about" and "htmlDemo"html files will not be included in the clean version of Assignment 4

Specification:

As mentioned above, this assignment will build upon your code from Assignment 4. To begin, make a copy ofyour assignment 4 folder and open it in Visual Studio Code. Note: this will copy your .git folder as well(including the "cyclic" remote for assignment 4). If you wish to start fresh with a new git repository, you willneed to delete the copied .git folder and execute "git init" again.

Part 1: Getting Express Handlebars & Updating your views

Step 1: Install & configure express-handlebars

* Use npm to install the "express-handlebars" module
* Wire up your server.js file to use the new "express-handlebars" module, ie:
  + "require" it as exphbs
  + add the app.engine() code using exphbs.engine({ ... }) and the "extname" property as ".hbs" and the"defaultLayout" property as "main" (See the Week 9 Notes)
  + call app.set() to specify the 'view engine' (See the Week 9 Notes)
* Inside the "views" folder, create a "layouts" folder

Step 2: Create the "default layout" & refactor home.html to use .hbs

* In the "layouts" directory, create a "main.hbs" file (this is our "default layout")
* Copy all the content of the "home.html" file and paste it into "main.hbs"
  + Quick Note: if your theme.css link looks like this href="css/theme.css", it must be modified to use aleading "/", ie href="/css/theme.css"
* Next, in your main.hbs file, remove all content INSIDE (not including) the single <div class="container">...</div>element and replace it with {{{body}}}
* Once this is done, rename home.html to home.hbs
* Inside home.hbs, remove all content EXCEPT what is INSIDE the single <div class="container">...</div> element(this should leave a single <div class="row">...</div> element)
* In your server.js file, change the GET route for "/" to "render" the "home" view, instead of sending home.html
* Test your server - you shouldn't see any changes. This means that your default layout ("main.hbs"), "home.hbs"and server.js files are working correctly with the express-handlebars module.

Step 3: Update the remaining "about", "addStudent" and "htmlDemo" files to use .hbs

* Follow the same procedure that was used for "home.html", for each of the above 3 files, ie:
  + Rename the .html file to .hbs
  + Delete all content EXCEPT what is INSIDE the single <div class="container">...</div> element
  + Modify the corresponding GET route (ie: "/about", "/htmlDemo" or "/students/add") to "res.render" theappropriate .hbs file, instead of using res.sendFile
* Test your server - you shouldn't see any changes, except for the fact that your menu items are no longerhighlighted when we change routes (only "Home" remains highlighted, since it is the only menu item within ourmain.hbs "default layout" with the class "active".

Step 4: Fixing the Navigation Bar to Show the correct "active" item

* To fix the issue we created by placing our navigation bar in our "default" layout, we need to make some small updates, including adding the following middleware function above your routes in server.js:
  + app.use(function(req,res,next){
  + let route = req.path.substring(1);
  + app.locals.activeRoute = "/" + (isNaN(route.split('/')[1]) ? route.replace(/\/(?!.\*)/, "") : route.replace(/\/(.\*)/, ""));
  + next();
  + });
  + This will add the property "activeRoute" to "app.locals" whenever the route changes, ie: if our route is"/students/add", the app.locals.activeRoute value will be "/students/add".
* Next, we must use the following handlebars custom "helper" (See the Week 9 notes for adding custom "helpers")"
  + navLink: function(url, options){
  + return '<li' + ((url == app.locals.activeRoute) ? ' class="nav-item active" ' : ' class="nav-item" ') +'><a class="nav-link" href="' + url + '">' + options.fn(this) + '</a></li>';
  + }
* This basically allows us to replace all of our existing navbar links, ie: <li class="nav-item"><a class="nav-link "href="/about">About</a></li> with code that looks like this {{#navLink "/about"}}About{{/navLink}}. The benefit here is that the helper will automatically render the correct <li> element add the class "active" if app.locals.activeRoute matches the provided url, ie "/about"
* Next, while we're adding custom "helpers" let's add one more that we will need later:
  + equal: function (lvalue, rvalue, options) {
  + if (arguments.length < 3)
  + throw new Error("Handlebars Helper equal needs 2 parameters");
  + if (lvalue != rvalue) {
  + return options.inverse(this);
  + } else {
  + return options.fn(this);
  + }
  + }
  + This helper will give us the ability to evaluate conditions for equality, ie {{#equal "a" "a"}} ... {{/equal}} will renderthe contents, since "a" equals "a". It's exactly like the "if" helper, but with the added benefit of evaluating asimple expression for equality
* Now that our helpers are in place, update all the navbar links in main.hbs to use the new helper, for example:
  + <li class="nav-item"><a class="nav-link" href="/about">About</a></li> will become {{#navLink"/about"}}About{{/navLink}}
  + NOTE: You can remove the "/tas" menu item from main.hbs and the "/tas" route from server.js, as wellas the "getTAs()" function from collegeData.js, as we will not be using these
* Test the server again - you should see that the correct menu items are highlighted as you navigate between views