Module 4 Coding Assignment

Click here to SIGN UP for the ***Single Page Web Applications with AngularJS*** course on Coursera. It's FREE!

Time to code something cool! Yay!!!

Time to Complete

It should take about an 1.5 hours or less. (This is just an estimate. People's backgrounds differ, so for some people it will take longer to complete.)

Ask questions in Week 4 Discussion Forum if you get stuck! We are all learning, and going through getting stuck and then unstuck (even with someone's help) can be a very valuable learning experience!

DO NOT be scared by the length of this assignment! It's not long at all! I just wanted to explain everything as clearly as I could and break it down into smaller steps for your benefit.

Assignment Instructions

General Idea

Super simple idea behind this week's assignment: use the restaurant menu REST API to create a master/detail view pair.

Your application should have 3 views (i.e., 3 view states): home (`home`), categories list (`categories`), items list (`items`).

As long as the views are functional and readable, the styling does not really matter and is not graded.

When the user goes to `/` path in your application, a home screen should be shown. It's up to you what you place on the home screen. You can just say "Welcome to our Restaurant". The home screen should have a link to a list of categories of menu items in the restaurant. Clicking that link would obviously take the user to the `/categories` view.

The categories view should list all available categories of items on the menu. Each listing should be a link. Clicking that link should take the user to the `/items` view. Note that since what the `items` view state shows is dependent on which category the user clicked, the URL mapping will need to have a parameter in it. I.e., don't take the URLs I am listing in the assignment description as literal URLs. They are just naming hints.

Make sure that if, while viewing the list of menu items for a particular category, the user copies the URL in the browser's address bar and pastes it into a new tab or a different browser, that the view is the same as the original one.

Rules

Breaking one of these rules will cause you to fail the assignment:

- You are not allowed to use regular HTML `onclick` attribute to bind behavior to the button. You **must** use the AngularJS way of binding behavior.
- At no point should your Javascript code look up *anything* in the DOM of the browser.

Steps

Here is what you will need to do to complete the assignment:

Steps for Setups (similar to all other assignments)

1. (If you haven't already) Create a GitHub.com account and a repository that you will use for this class.
2. (If you haven't already) Follow the Development Setup Video (beginning of Module 1) instructions on how to create a repository and set it up such that you can host and view your finished web pages on GitHub Pages, i.e., GitHub.io domain name. You will need to provide that URL for your peer review.
3. Create a folder in your repository that will serve as a container folder for your solution to this assignment. You can call it whatever you want. For example, `module4-solution` or `mod4_solution`, etc.
    - You can do this by 'cloning' your repository with `git clone https://www.github.com/your_repo_url` to your local machine, creating `module4-solution` folder in the root of the repository directory along with a README.txt inside of the `module4-solution` directory. Then, you would do `git add .`, followed by `git commit -m 'New folder'`, followed by `git push` to upload the new folder with the README.txt to the GitHub repository.
4. HTML/CSS for the assignment are very simple and I'll leave it up to you to generate. Again, this class is not about styling, so as long as things are functional and readable, styling or layout doesn't matter much.
5. Import AngularJS and ui-router library into your HTML page. You may want to simply copy the JS files from the `examples/Lecture36/lib` folder of this repository.

General Steps for Implementing Assignment Requirements

Suggestion: Read through all the requirements first and then come back to #1 again. These are **not** necessarily listed in the order you should create them. Choose whatever order works for you. This list is simply an extended hint of how to implement a solution for this assignment.

1. Declare `ng-app` either on the `html` or the `body` element. Name your app `MenuApp`.

2. You must follow the 1 artifact per file rule. That means if the JS file declares a controller, it should *not* declare anything else like a service or component and vice versa. Use a separate JS file to declare other artifacts in your application. *Don't forget to include each JS file in your* `index.html`*!*
3. Create a file called `menuapp.module.js` and declare an Angular module to match your `ng-app` declaration.
4. Create `data.module.js` file and declare another module in it called `data`. Make sure the `MenuApp` module lists the `data`module as a dependency.
5. Create `menudata.service.js` file and create a service called `MenuDataService` in it. This service should be declared on the `data` module, *not* on the `MenuApp` module. The `MenuDataService` should have 2 methods:
   - `getAllCategories` - this method should return a promise which is a result of using the `$http` service, using the following REST API endpoint: https://davids-restaurant.herokuapp.com/categories.json
   - `getItemsForCategory(categoryShortName)` - this method should return a promise which is a result of using the `$http`service, using the following REST API endpoint: https://davids-restaurant.herokuapp.com/menu_items.json?category=, where, before the call to the server, your code should append whatever `categoryShortName` value was passed in as an argument into the `getItemsForCategory` method.
6. Create `categories.component.js` file and create component called `categories` that shows all available categories in the menu to the user.
7. Create `items.component.js` file and create a component called `items` that shows all of the menu items for a particular category.
8. The `categories` and the `items` components should *NOT* directly use the `MenuDataService` to fetch their own data. Instead, the proper data should be simply passed into the component. (Hint: use the one-way `<` binding).
9. Create `routes.js` file and configure your routes and view states in it. These routes should be defined in the `MenuApp`module.
   - *Hint:* don't try to define the states all at once. Define one state, including whatever it needs and make sure it works all the way to the point when you can see the results on the screen. Then, move on to the next view state. That *does* mean that you will have to leave `routes.js` and define all the other artifacts listed below and then come back to it, etc.
   - *Hint:* The `home` state will not need a controller. Just a template.
   - *Hint:* The `categories` state can have a `controller` as well as a `resolve`. The resolve will use the `MenuDataService`to retrieve categories and inject them into the controller. The controller can then expose the retrieved categories object such that it can be simply passed into the `categories` component.
   - *Hint:* The `items` state can have the same type of setup as the `categories` state.

Important Implementation Notes

1. Make sure all of your Javascript code is inside of an IIFE. (*If you don't know what that is or why we'd want to use it, brush up on it by looking through module 4 of HTML, CSS, and Javascript for Web Developers course I teach.*)
2. Make sure all of your dependency injections are protected from minification.

3. After you are done and satisfied with your solution, don't forget to add/commit/push your code to your repository.

IMPORTANT REMINDERS:

- Closely follow the submission guidelines for this assignment on Coursera.org
- Make **sure** you provide the correct URL in your submission (it should be GitHub**.io**, *not* GitHub**.com**)
- Make **sure** to TEST your assignment not just on your local machine, but ALSO once you deploy it on GitHub, using the URL you are providing as part of your submission.
- This assignment will be peer-reviewed (and graded). The guidance will be given such that if submission instructions are not followed, the assignment is to be failed. This includes providing the wrong URL for your deployment. Following instructions is very much part of software development. After all, that's what software requirements are - instructions to follow.