Capstone Project - Phase 4 of 6 (ReST Interface Integration)

Getting Started

Recall that you already have both a local and a remote repository for your Capstone Library project. In Phase 1 of the Capstone Project, you built the engine that will drive your application using JavaScript, in Phase 2 you built the User Interface for your digital library application, and in Phase 3, you connected your engine to your UI with the help of jQuery, init(), bindEvents(), and eventHandlers(). **Before you begin Phase 4, please merge your final copy of Phase 3 into your develop branch if you have not already done so.** As we enter Phase 4, we will create a new feature branch within our feature folder, titled **phase-four**.

Your task is to set up a database for your capstone project and then create HTTP/AJAX requests to take the place of your HTML5 local storage so that the information in your database can be displayed in your DataTableUI. We have spent time getting your mLabs, Robo 3T, and Postman accounts/apps set and ready for you. If any of these aren't working, please let Kyle or Shambre know so we can help. To do this project, you will first need to be able to navigate to the library-service directory through the terminal and open up your node server.js. This will get your Library API (application programming interface) up and running. From here, although unnecessary and entirely optional, it can be nice to have Robo 3T open so that you can verify when a data entry is created through your POST method or retrieved through your GET method.

Set Up

I'm working on a formal write-up of the procedures we went through to get our databases set up. These will be shared with you separately from this document at a later time as a reference.

Your file structure at this point should not need to be modified, however, you may want to open up the library-service folder inside of your Atom/VSCode window so that you can reference both your library project and your service layer as you work. You can do this by opening your library project in Atom/VSCode and then navigating to File → Add Project Folder → library-service directory. It is also fine to have the two directories open in separate windows.

Requirements

- Create jQuery Ajax or HTTP requests (it's up to you which to use) to perform the following 5 jobs by interacting with your Library UI
    a. GET
        - Remove your local storage functionality by commenting it out, or deleting it
        - Render your book data in your table through a GET/Library request
        - This will happen on page load. Reach out to your GET/Library API at your local host address using either (http://127.0.0.1:3002/Library OR localhost:3002/Library)
        - Store the results of the GET somewhere, such as in your bookShelf array
    b. GET:ID
        - Render an individual random book
        - Remember to reference the MongoDB through the id
    c. POST
        - Create a new book that displays on your DataTableUI view
    d. DELETE:ID
        - Remove a book from your library database array and from your DataTableUI
        - Add your id on as a data attribute so that you can hook into it to delete that entry
    e. PUT:ID
        - Update a book object's properties in your DataTableUI view and your database array
        - Remember to reference the MongoDB through the id

What to Do

1. Review the requirements
2. Plan out how you think you can best accomplish these targets and determine various approaches you can try
3. Research and read about asynchronous callbacks and how they work, AJAX requests, and HTTP requests in the documentation from Helpful Materials below
4. Start writing your code
5. Discuss issues with your peers and ask for help early and often!
6. Slack Shambre if you have a pressing issue that you cannot solve
7. Commit and push every time a new module is completed and functional

Helpful Materials

https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82
https://www.upwork.com/hiring/development/intro-to-apis-what-is-an-api/
https://medium.freecodecamp.org/understanding-asynchronous-javascript-callbacks-through-household-chores-e3de9a1dbd04
https://medium.freecodecamp.org/javascript-callbacks-explained-using-minions-da272f4d9bcd
http://callbackhell.com/
https://restfulapi.net/
https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/
https://www.w3schools.com/jquery/jquery_ajax_get_post.asp
https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them
http://mongoosejs.com/docs/api.html
https://docs.mongodb.com/manual/
https://www.codementor.io/wapjude/creating-a-simple-rest-api-with-expressjs-in-5min-bbtmk51mq
https://expressjs.com/en/guide/routing.html

Submitting

- Add Kyle and Shambre as collaborators (kbrothis and ShambreSW) on GitHub
- Project due at **9:00 am on Monday, July 30th, 2018**
    - If you do not think you can finish by this time, Kyle and I must receive a Slack message from you saying so by **9:00 am** on Friday, July 27th, 2018 (it's okay to tell us you won't finish and then discover that you are able to, but we want to know if this pace is too strenuous and adjust if necessary)