CMPS 356 Enterprise Application Development - Spring 2020
Lab 7 - Web API

Objective

The objective of this lab is to practice implementing Web API (also known as REST API) using Node.js and Express. Then you will test it using Postman and mocha.

Overview

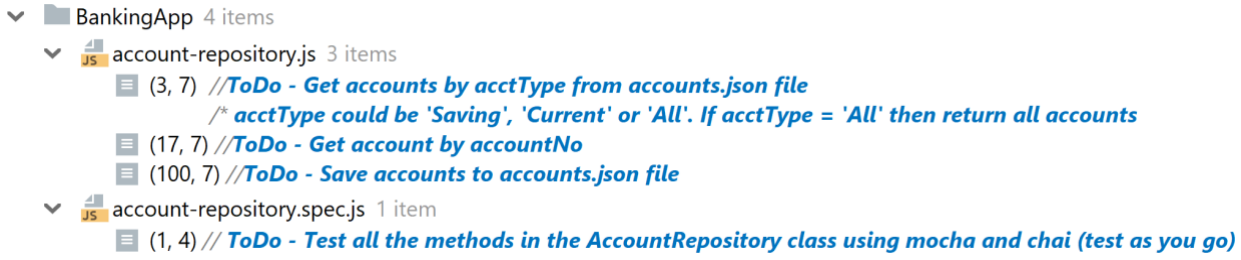
This lab has two parts:

- **Part A**: Extend the Banking App to asynchronously read/write data from the accounts.json file and make the App functionality accessible via Web API.
- **Part B**: Extend the Book Donation App to asynchronously read/write data from the book-donation.json file and make the App functionality accessible via Web API.

Part A - Extend the Banking App to asynchronously read/write data from the accounts.json file and make the App functionality accessible via Web API

1. Sync cmps356-lab repo to get the Lab files.
2. Copy ***Lab7-WebAPI*** folder from cmps356s20-Lab repo to your repository.
3. Open ***Lab7- WebApi \BankingApp*** in WebStorm. Run **npm install** to install the packages.

   Open the ToDo window and complete the pending ToDo tasks.

4. Develop Web API to make the App functionality accessible via the web:
   - Run **npm install express** to install express package.
   - Run **npm install body-parser** to install the body-parser package. This middleware extracts the body portion of an incoming request and assigns it to req.body.
   - Create **app.js** file, import express package, instantiate an express app, configure it then start it at port 9090.
   - Create a **bank-service.js** file and implement handlers for the Urls listed in the table below. The Url handlers should use the methods from the Bank class.

HTTP Verb	Url	Functionality
Get	/api/accounts/:acctType	Returns accounts by acctType
Get	/api/accounts?accountNo=XXX	Returns an account by accountNo
Post	/api/accounts	Adds an account
Put	/api/accounts	Updates an account
Delete	/api/accounts?accountNo=XXX	Deletes an account by accountNo

5. Test the Web API using Postman. First download and install it from https://www.getpostman.com/
6. Create a **bank-service.spec.js** file. Test the Web API using chai-http. Documentation available at https://github.com/chaijs/chai-http

Part B - Make the Book Store App functionality accessible via Web API.

getBook(name)	Returns the book object if found otherwise "Not found" exception.
getBooksByPageCount(pageCount)	Returns the books with pages >= the pageCount parameters. E.g. Calling the function with pageCount=200 should return all the books with pages >= 200.
getBooksByAuthor(author)	Returns all the books authored by that specific author. **Note**: some books have more than one author. You should consider those too and return them as well.
getBooksbyCatagory(category)	Returns the books for a particular category. E.g. Calling the function with *category = Programming* should return all the programming books.
getAuthorsBookCount()	Returns a map that contains the author name and the number of books they have authored. E.g. **Author Name** | **Book Count** James | 2 Ali | 4

Develop Web API to make the Book App functionality accessible via the web:

- Create **app.js** file, import express package, instantiate an express app, configure it then start it at port 8090.

- Create a **book-service.js** file and implement **BookService** class with methods to handle the requests listed in the table below. The handlers should use the provided methods from the **BookRepository** class.
- Create a **book-router.js** file and implement handlers for the Urls listed in the table below. The Url handlers should use the methods from the BookService class.

<table>
<tr><th>HTTP Verb</th><th>Url</th><th>Functionality</th></tr>
<tr><td>Get</td><td>/api/books?name=</td><td>Returns the book by name</td></tr>
<tr><td>Get</td><td>/api/books?pageCount=</td><td>Returns the books with pages >= the pageCount parameters. E.g. Calling the function with pageCount=200 should return all the books with pages >= 200.</td></tr>
<tr><td>Get</td><td>/api/books?author=</td><td>Returns all the books authored by that specific author.</td></tr>
<tr><td>Get</td><td>/api/books?category=</td><td>Returns the books for a particular category.

E.g. Calling the function with category = Programming should return all the programming books.</td></tr>
<tr><td>Get</td><td>/api/books/summary</td><td>Returns a map that contains the author name and the number of books they have authored. E.g.

<table><tr><th>Author Name</th><th>Book Count</th></tr><tr><td>James</td><td>2</td></tr><tr><td>Ali</td><td>4</td></tr></table></td></tr>
<tr><td>Post</td><td>/api/books/</td><td>Adds a book</td></tr>
<tr><td>Put</td><td>/api/books/:isbn</td><td>Updates a book</td></tr>
<tr><td>Delete</td><td>/api/books/:isbn</td><td>Deletes a book</td></tr>
</table>

1. Test the Web API using Postman.
2. Create a **book-service.spec.js** file. Test the Web API using chai-http.

After you complete the lab, fill in the ***Lab7-TestingDoc-Grading-Sheet.docx*** and save it inside **Lab6-WebApi** folder. Sync your repository to push your work to Github.