Project 5 - Fetch

New Attempt

Due Jul 6, 2021 by 11:59pm **Points** 20 **Submitting** a website url

Due: Tuesday, July 6 [This is a hard due date and will *not* change. I will take off 2.5 points for every day late]
Points: 20 points (+10 points for peer reviews)
Deliverables: Upload [Name]BookstoreFetch.war to **cs5244.cs.vt.edu:8080/ArchiveUpload/ (http://cs5244.cs.vt.edu:8080/ArchiveUpload/)**
Resources: **CorsFilter.java (https://drive.google.com/file/d/11Pqll8djz376mwvDY20t1z0KhUYHYu03/view?usp=sharing)**

Note: The following requirements have been added to this project on Monday, June 28:

- **Fix any problems that I mentioned in my comments to Project 3 (Vue project). If you are still working on issues, let me know in Canvas when you submit Project 5**
- **Implement a transition somewhere in your project. If the transition is not obvious (ex: dropdown menu or add-to-cart button), please let me know what it is in Canvas when you submit Project 5**
- **If you implement a hamburger menu similar to the one in the hamburger-demo, please do add or change something about the menu that distinguishes it from the one in the demo. For example, maybe the hamburger icon changes to an "X", or maybe the sub-menu comes out from the side.**
- **Ensure style change (ex: darker background) when you hover over buttons**
- **Style read-me button differently from add-to-cart button**

Overview

In this project, you will combine Project 3 (Vue) with Project 4 (Rest). The result will be a home page and category page that dynamically update depending on the selected category. To do this, we are going to use the Fetch API built into JS6 to retrieve data from the REST API created in Project 5.

Setup your project

Create a new (no-plugin) Gradle project in IntelliJ called [Name]BookstoreFetch. Create two modules:

- server - Gradle with Java and web plugins
- client - Vue with Linter and Router (as in P3)

Once you've done this, **I recommend quitting IntelliJ** before copying the following files into your new project.

From your Vue project (P3) copy the following into the **client** side:

- src folder
- vue.config.js

From your REST project (P4) copy the following into the **server** side;

- build.gradle
- src folder

Set up a Tomcat run configuration for the server (with application context [Name]BookstoreFetch) and **run the Tomcat configuration**. The REST API should work as in Project 4.

Run the client using "npm run serve". The Home and Category pages should look and behave just as they did in Project 3. Look at the address bar. Does it still have "[Name]BookstoreVue" in the path? If so, modify the vue.config.js file to change that to "[Name]BookstoreFetch".

Fetch Categories for your CategoryNav component

Follow along with these slides to complete this portion of the assignment:

- **Bookstore Fetch slides** **(https://drive.google.com/file/d/1x-DbqIdMvCTFyjoD1-xLiTSsm7KOmP_G/view?usp=sharing)**

Fetch Categories as needed

In addition to CategoryNav, fetch the categories for HeaderCategoryDropdown. If you have categories on your front page, also fetch the categories for HomeCategoryList. Use the ApiService that has already been created.

Fetch Books for your Category component

In your ApiService, create a method called "fetchSelectedCategoryBooks". It should take the category name as a parameter. Use this method in your Category view — similar to how we used ApiService.fetchCategories in the CategoryNav component — to display the book boxes for the selected category. This will involve passing the fetched books as properties to the CategoryBookList and then CategoryBookListItem components.

To load the book images, create a custom JavaScript method that takes a the book title from the database, and returns the image file name. It should look something like this:

```
bookImageFileName: function(book) {
  let name = book.title.toLowerCase()
  name = name.replace(/ /g, '-')
  return `${name}.jpg`
}
```

Modify this code to work with your book-image filenames. For example, my book-image files were "gif" files. Also, I had books with punctuation in the title, so I replaced all punctuation with an empty string.

Finally, to enable the navigation between categories to reload the books on the category page, in App.vue add :key="$route.fullPath" to the router view tag:

```
<router-view id="router-view" :key="$route.fullPath"></router-view>
```

This tells Vue to reload each page when the path changes.

Now navigating to different categories should load the books for each category.