Sabrina Macmanus

Final Capstone

Documentation

To Load E-Commerce Program:

1. In Workbench: create a SQL database named “shopping\_cart”
2. In IntelliJ: change application properties from “none” to “create”, and make sure username and password matches your local server
3. Run finalCapstoneApplication in IntelliJ to create the table in Workbench;
4. You can use the included SQL file to help load starter objects.
5. IntelliJ: change “create” back to “none”
6. Run finalCapstoneApplication again to load server.
7. In Angular, start finalCapstone by “npm start” in the Terminal, or “ng serve” in Bash.

Routes included:

1. GET/READ inventory “/inventory”
2. GET/READ inventory by id “/inventory/{id}”
3. POST/CREATE inventory “/inventory”
4. PUT/UPDATE inventory “/inventory/{id}”
5. DELETE inventory by id “/inventory/{id}”
6. POST to change inventory quantity “/purchase”
7. GET/READ tax amount for items by id “tax/{id}”

Business Logic in Java:

1. getImportTax
2. getCategoryTax
3. getTotalTax

In Angular:

1. Main page includes navbar and main animation
2. Admin component/page includes navbar and form for editing/updating inventory
3. Items page includes navbar, animation, search bar that filters by category and title, and inventory list
4. Item cards contain a routerLink that will go to ItemDetailComponent, where adding to Cart occurs
5. Checkout is the Cart component, which lists the items added to the cart, buttons to adjust quantities, and a modal receipt

Logic in Angular:

1. Items service contains Item object/class , the connecting apiUrl to the local server, and the methods connecting server routes to angular
2. Cart service contains methods for adding and removing items from cart.

Custom Pipes in Angular:

1. Item-Search pipe is used for filtering the search box
2. Rounding pipe is used for rounding tax

Assets:

1. Picture files for inventory are located here.
2. Portal-text.js includes JavaScript code for animating Navbar title