Angular and Spring Boot E-commerce App Steps

Format: shift + option + F

Overview

A diagram of a application

Description automatically generated with medium confidence

A close-up of a product list

Description automatically generated

09 SpringBoot Backend

Development Process:

1. Set up the database tables: MySQL
2. Create a SpringBoot starter project (start.spring.io)A screenshot of a computer

   Description automatically generated
3. Develop the Entities: Product and Product Category
   1. New package: entity -> new java class: Product and ProductCategory
   2. Note: ManyToOne, OneToMany
4. Create REST APIs with Spring Data JPA Repositories and Spring Data REST
   1. New package: dao -> new Interface: ProductRepository and ProductCategoryRepository
5. Make the REST API as READ-ONLY
   1. New package: config -> new java class: MyDataRestConfig
   2. Disable HTTP methods: POST, PUT, DELETE
   3. Test with Postman

10 Augular Frontend

Development process

1. Create Angular project
   1. Terminal; ng new angular-ecommerce -> n for routing, CSS
2. Create Angular component for product-list
   1. Ng generate component components/product-list
3. Develop TypeScript class for Product
   1. Ng generate class common/product
   2. Add constructor
4. Create Angular service to call REST APIs  
   A diagram of a service

   Description automatically generated
   1. Ng generate service services/product
   2. App.module.ts: add HttpClientModule and ProductService
   3. Product.service.ts: Inject HttpClient; Map the JSON data from SpringData REST to Product array; Unwraps the JSON from Spring Data REST \_embedded entry
5. Update Angular component to subscribe to data from Angular service
   1. Product-list.component.ts: Inject ProductService, subscribe, integrating service with angular component
6. Display the data in an HTML page
   1. Product-list.component.html
   2. Display list of products in an HTML table
7. Add CrossOrigin support to Spring Boot app
   1. Specific to JavaScript running on website.  
      Origin = protocol + hostname + port
   2. ProductRepository: add @CrossOrigin
   3. The same for ProductCategoryRepository
8. Add images to assets directory

11 Integrating online shop template

Release 2.0 – Plan

* Online shop template integration
* Search for products by category
* Search for products by text box
* Master / detail view of products
* Pagination
* Add products to shopping cart (CRUD)
* Shopping carts check out

Development Process

1. Download the HTML template starter files
2. Install Bootstrap CSS styles locally using npm
   1. Npm install bootstrap
   2. Npm install fortawesome/fontawesome-free
3. Add local custom CSS styles to Angular src/styles.css file
4. Integrate template files into Angular app
   1. Source -> app -> App.component.html
   2. Copy from home.html
      1. Menu sidebar
      2. Container – Desktop
      3. Footer
5. Add support for icons and logos
   1. copy product-list-table.component.html and rename to product-list-grid.component.html
6. Enhance our application with product images
   1. Copy asset pictures to assets -> images -> products folder

12 Search for products by category

Angular Route

A screenshot of a computer

Description automatically generated

Development Process

1. Define routes
   1. App.module.ts: define routes
   2. Specfic -> generic -> 404
2. Configure Router based on our routes
   1. @NgModule -> imports -> RouterModule.forRoot(routes)
3. Define the Router Outlet
   1. App.component.html
4. Set up Router Links to pass category id param
   1. App.component.html -> Menu sidebar -> href replace with routerLink
5. Enhance ProductListComponent to read category id param
   1. product-list.component.ts: add currentCategoryId: number = 1
   2. Constructor -> Inject the ActivatedRoute
   3. ngOnInit() this.route.paramMap.subscribe()…
6. Modify SpringBoot app – REST Repository needs new method
   1. ProductRepository add method
7. Update Angular Service to call new URL on Spring Boot app
   1. Product.service.ts -> getProductList()