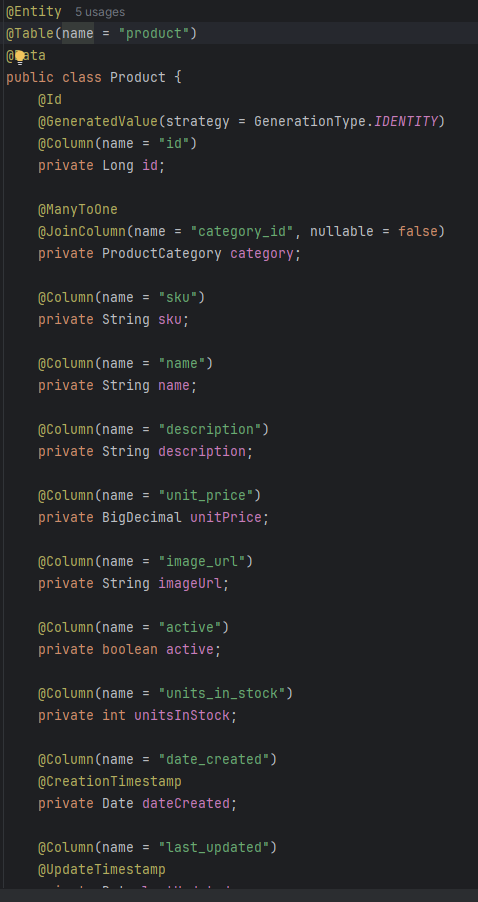
1. Create an angular project
2. Add bootstrap support to the project by adding header on index.html
3. Create Angular components for Product-list

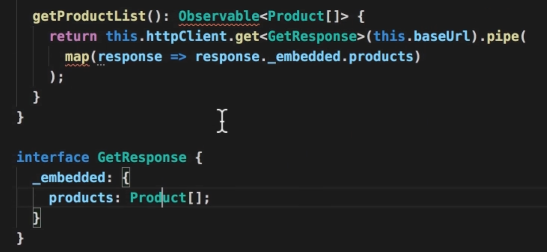
Remember a component consists of html for display and Ts file for calculation

1. We create a typescript class for Product. Use terminal window to create component in /common directory. This class is for Product entity similar to java entity

1. Create a Productservice class in the /service directory. The service class fetches product data from an API and provides it to multiple components. But first we let the app.module to import HttpClientModule and also add Productservice to it.

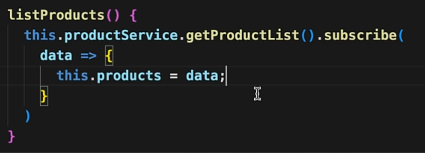
For now provide the baseurl static . Inject the httpclient to the product service constructor

In the service class create a method name getProductList() that returns an Observable object of type product array (Product[]). This function is basically gonna map the json data from spring data REST to a Product array. 

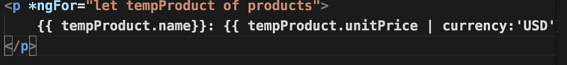
1. Now in productList.component.ts we are gonna inject the ProductService into the productListComponent through the constructor. Since we get the product[] array from the service injection. We create a Product property for the component class. To store the array

in the component class define a listProducts() function and call it inside the ngOnInit.

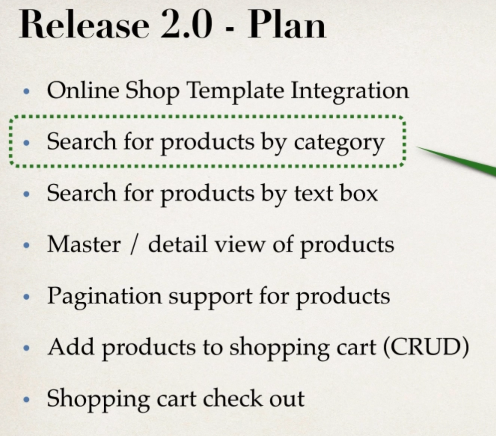
In the method listproducts use the product service to assign data to the Product array

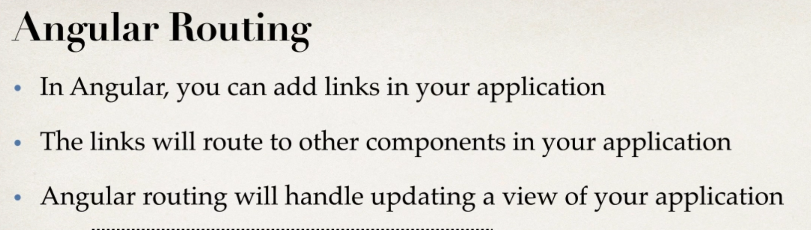


1. Now we do need to display the data that is in the product array so in productComponent.html

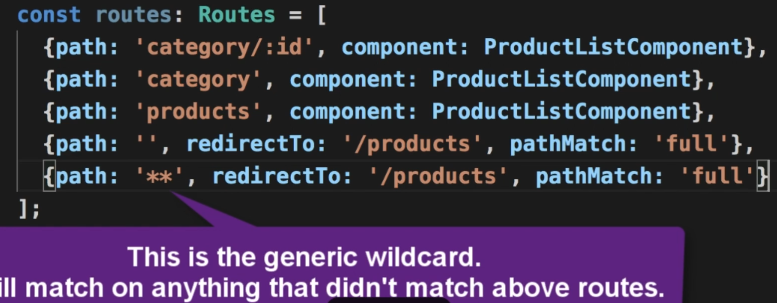








1. **Defining Routes and Configuring Router based on our routes**

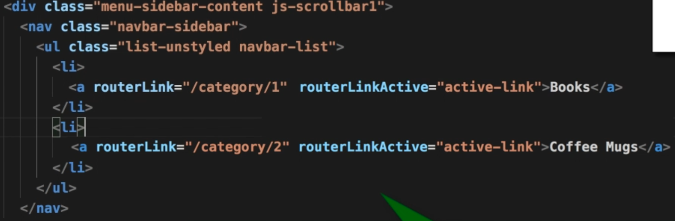
In app.module.ts we create an array of routes

remember the order of routes is important. Now in app.module.ts add the array by using RouterModule.forRoot(routes) in the imports[]

1. **Defining Router Outlets and Setting up Router Links to pass category id param**

Now in the main app.component.html we are gonna replace the

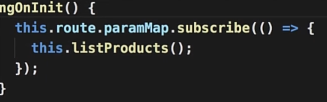
<app-product-list></app-product-list> with <router-outlet></router-outlet>

Then we setup our menu links 

1. **Enhance ProductListComponent to read category id param**

Lets add another property for the ProductListComponent. This time it’s the currentCategoryId which is initially equal 1

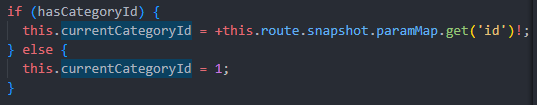
We also inject the route: ActivetedRoute to the component class. We need this to access the categoryid (aka “/categoryId/1”). Since the Activated route(this.route) has the category id we need we get it using the ngOnInIt Function



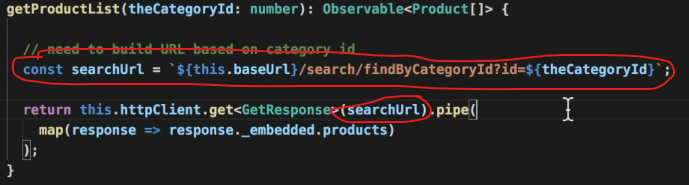
We make some changes to the listProducts() function by first checking if there is a catagoty id or not in the route



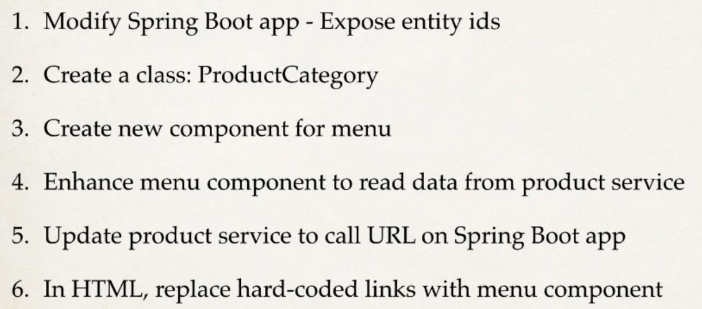
If it has an category id we convert it into number to string



Now we pass the currentCategoryId to the getProductlist() function so we make changes in the product service so that getProductlist accepts



Product Category Component.



1. To expose the entity IDs, in MyDataRestConfig firstly inject the MyEntityManger Object to the MyDataRestConfig class. In the configureRepositoryRestConfiguration function to expost the id create a function exposeId(config) . this function takes the RepositoryRestConfiguration config as parameter.



This line fetches a Set of all entity types known to the JPA EntityManager. (e.g., classes marked with @Entity).

Then create an empty array list.



Loop over the entites set and add then to the Class arraylist by converting them from entity type to java type.

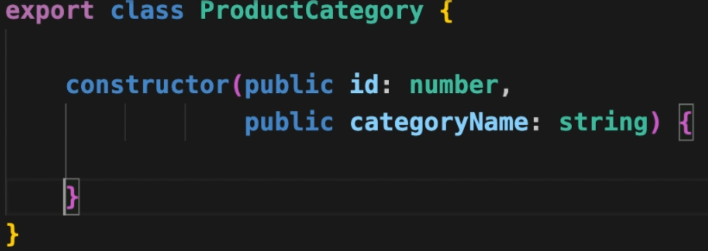


Convert the arraylist into an array of Type Class[]



We take the parameter config and expose the id in it by using the Class[] domainTypes

1. Create the class ProductCategory



1. Create a new component For ProductCategory menu

Similarly in ProductListComponent create a properties array for Product Categories.

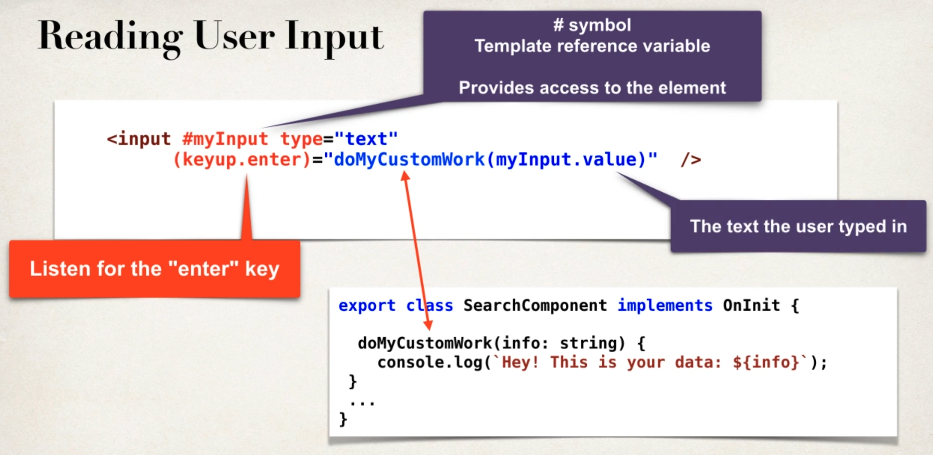
Inject the product service to this component to get the data using the listProductCategories() method. Which uses the getProductCategories of the service class.

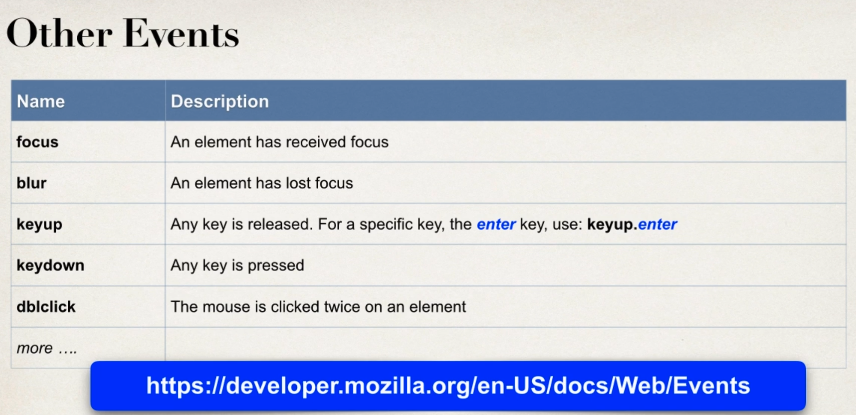
Populate the data in the productCategories array.

1. Code the function getProductCategories in the service class.
2. Implement the component in HTML using \*ngfor

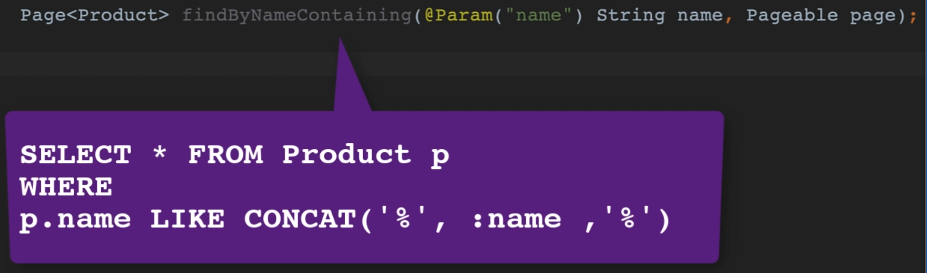
Product Search Component.







1. **Modifying Spring Boot App (Adding a new search method)**

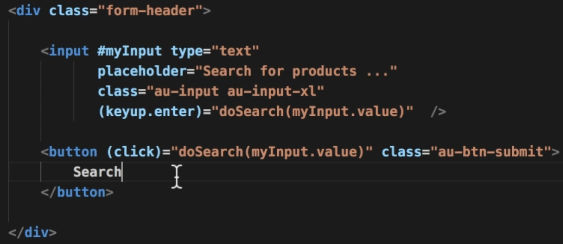
In the java backend in the product repository add a new Query method 

1. **Adding a new route for searching and creating a new component for search.**

In module.ts create a new route for search

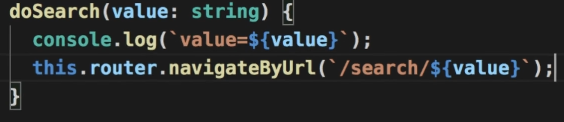
Generate a new conponent name search.

1. **Updating the search component to send data to search route.**

Place the search component <app-search /> in the app.component.htm 

Inject the router to our search component.

In the vomponent.ts implement the event function doSearch()



1. **Enhancing ProductListComponent to search for products with ProductService**

Create a property for product list componont searchMode: boolean; then add a function handleListProducts() do some refactoring. 