Please note there are so many files in Angular and spring boot, so I am copying the main codes, please find the entire application at - <https://github.com/vishnugautam/KitchenStory>

**App.routing.module:**

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { AdminLogoutComponent } from './admin-logout/admin-logout.component';

import { AdminloginComponent } from './adminlogin/adminlogin.component';

import { CheckoutComponent } from './checkout/checkout.component';

import { ErrorComponent } from './error/error.component';

import { HomepageComponent } from './homepage/homepage.component';

import { InventoryComponent } from './inventory/inventory.component';

import { ItemComponent } from './item/item.component';

import { LandingpageComponent } from './landingpage/landingpage.component';

import { OrderoverviewComponent } from './orderoverview/orderoverview.component';

import { SearcheditemsComponent } from './searcheditems/searcheditems.component';

import { RouteGuardService } from './service/route-guard.service';

const routes: Routes = [

  {path:'', component: LandingpageComponent},

  {path:'login', component: AdminloginComponent},

  {path:'home/:name', component: HomepageComponent, canActivate: [RouteGuardService]},

  {path: 'inventory', component: InventoryComponent, canActivate: [RouteGuardService]},

  {path: 'logout', component: AdminLogoutComponent, canActivate: [RouteGuardService]},

  {path: 'inventory/:id', component: ItemComponent, canActivate: [RouteGuardService]},

  {path: 'searcheditems/:foodname', component: SearcheditemsComponent},

  {path: 'overview/:id', component: OrderoverviewComponent},

  {path: 'checkout', component: CheckoutComponent},

  {path: '\*\*', component: ErrorComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**Inventory data service to connect with the backend:**

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Inventory } from 'src/app/inventory/inventory.component';

@Injectable({

  providedIn: 'root'

})

export class InventoryDataService {

  constructor(private httpClient : HttpClient) { }

  retrieveAllFoods(hotel\_address: any) {

    return this.httpClient.get<Inventory[]>(`http://localhost:8080/admin/inventory/${hotel\_address}`) // which returns array of inventory

  }

  retrieveFoodByName(food\_name : String){

    return this.httpClient.get<Inventory[]>(`http://localhost:8080/admin/inventory/search/${food\_name}`)

  }

  retrieveFoodById(hotel\_address : String, id : number){

    return this.httpClient.get<Inventory>(`http://localhost:8080/admin/inventory/${hotel\_address}/${id}`)

  }

  updateItem(inventory : Inventory, id : number){

    return this.httpClient.put(`http://localhost:8080/admin/inventory/${id}`, inventory);

  }

  createItem(food\_name : String, inventory : Inventory){

    return this.httpClient.post(`http://localhost:8080/admin/inventory/${food\_name}`, inventory);

  }

  deleteFoodById(id: number){

    return this.httpClient.delete(`http://localhost:8080/admin/inventory/${id}`)

  }

}

**Inventory component.ts,**

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { InventoryDataService } from '../service/data/inventory-data.service';

@Component({

  selector: 'app-inventory',

  templateUrl: './inventory.component.html',

  styleUrls: ['./inventory.component.css']

})

export class InventoryComponent implements OnInit{

  // member variable

  // inventory = [{

  //   id: 1, description: 'Idli'},

  //   {id: 2, description: 'dosa'}

  //   ]

  // foodName = this.route.snapshot.params["foodName"];

  // foodDescription = this.route.snapshot.params["foodDescription"];

  // date = this.route.snapshot.params["date"];

  // price = this.route.snapshot.params["price"];;

  // hotelName = this.route.snapshot.params["hotelName"];

  // hotelAddress = this.route.snapshot.params["hotelAddress"];

  constructor(private inventoryDataService : InventoryDataService, private router : Router, private route : ActivatedRoute){

  }

  ngOnInit(): void {

    this.refreshInventory();

  }

  inventory : Inventory[] | undefined;

  refreshInventory(){

    this.inventoryDataService.retrieveAllFoods('tirupur').subscribe(

      response => {

        // console.log(response);

        this.inventory = response;

        // console.log(this.inventory);

      }

    )

  }

  updateInventory(id : any){

    // console.log(id);

    this.router.navigate(['inventory', id])

  }

  id! : number;

  deleteFood(id : number){

    console.log(id);

    this.inventoryDataService.deleteFoodById(this.id).subscribe(

      data => {

        this.router.navigate(['inventory']),

        window.location.reload()

      }

    )

  }

}

export class Inventory{

  constructor(public id : number, public foodName : string, public foodDescription : string, public price : number, public date : Date, public hotelName : string, public hotelAddress : string){

  }

}

**Item component.ts,**

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { Inventory } from '../inventory/inventory.component';

import { InventoryDataService } from '../service/data/inventory-data.service';

@Component({

  selector: 'app-item',

  templateUrl: './item.component.html',

  styleUrls: ['./item.component.css']

})

export class ItemComponent implements OnInit{

  id!: any;

  inventory!: Inventory;

  constructor(private inventoryDataService : InventoryDataService, private activatedRoute : ActivatedRoute, private router : Router){}

  ngOnInit(): void {

    this.id = this.activatedRoute.snapshot.params['id'];

    // we first assign the todo with 1 instance so that it is not undefined

    this.inventory = new Inventory(1,'idli', 'made from rice', 50, new Date, 'Annapoorna', 'Tirupur');

    this.inventoryDataService.retrieveFoodById('idli', this.id).subscribe(data =>

      this.inventory = data

      )

  }

  saveItem(){

    if (this.id == -1){

      this.inventoryDataService.createItem('idli', this.inventory).subscribe(

        data => {

          this.router.navigate(['inventory'])

        }

      )

    } else {

      this.inventoryDataService.updateItem(this.inventory, 1).subscribe(

        data => {

          this.router.navigate(['inventory'])

        }

      )

    }

  }

}

**Inventory.java,**

package simplilearn.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotEmpty;

import javax.validation.constraints.Size;

// convert this class to entity

// we are mapping this class with todos table

@Entity

@Table(name="inventory")

public class Inventory {

// to make the Id act like primary key we use @Id

@Id

// asking hibernate to insert the value using @GeneretedValue

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long id;

@NotEmpty @Size(min = 2, message = "the foodname at least should have 2 characters")

private String foodName;

private String foodDescription;

private String date;

private int price;

private String hotelName;

private String hotelAddress;

public Inventory() {

super(); //default

}

public Inventory(long id,

@NotEmpty @Size(min = 2, message = "the foodname at least should have 2 characters") String foodName,

@NotEmpty @Size(min = 5, message = "the food description at least should have 5 characters") String foodDescription,

int price, String hotelName, String hotelAddress, String date) {

super();

this.id = id;

this.foodName = foodName;

this.foodDescription = foodDescription;

this.date = date;

this.price = price;

this.hotelName = hotelName;

this.hotelAddress = hotelAddress;

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getFoodName() {

return foodName;

}

public void setFoodName(String foodName) {

this.foodName = foodName;

}

public String getFoodDescription() {

return foodDescription;

}

public void setFoodDescription(String foodDescription) {

this.foodDescription = foodDescription;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String getHotelName() {

return hotelName;

}

public void setHotelName(String hotelName) {

this.hotelName = hotelName;

}

public String getHotelAddress() {

return hotelAddress;

}

public void setHotelAddress(String hotelAddress) {

this.hotelAddress = hotelAddress;

}

}

**InventoryResource.java,**

package simplilearn.resources;

import java.net.URI;

import java.util.ArrayList;

import java.util.List;

import javax.validation.Valid;

import org.aspectj.weaver.ast.Instanceof;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import org.springframework.web.util.UriComponentsBuilder;

import simplilearn.errors.TodoNotFoundException;

import simplilearn.model.Inventory;

import simplilearn.services.InventoryService;

// like service annotation and repository annotation

// it converts this class into RESTful resource

@CrossOrigin(origins = "http://localhost:4200")

@RestController

public class InventoryResource {

@Autowired

private InventoryService inventoryService;

@DeleteMapping(path = "/admin/inventory/{id}")

public ResponseEntity delete(@PathVariable long id) {

Inventory theTodo = inventoryService.deleteById(id);

if(theTodo != null) {

// noContent means it would return 204 status code

return ResponseEntity.noContent().build();

}

return ResponseEntity.notFound().build();

}

// it is used to map the POST method

@PostMapping(path = "/admin/inventory/{food\_name}")

//Request Body contains the JSON data it has to be converted to Java object so we use RequestBody

// @Valid means whenever json comes from client in POST it checks weather valid information is sent

// only consider valid annotation

// it goes to handleMethodNotValid method in the centralized which is the HandleMethodArgumentNotValid in CustomizedTodoErrorMessageHandler

public ResponseEntity createFood(@PathVariable String food\_name, @Valid @RequestBody Inventory item) {

System.out.println("inside the post method");

Inventory saveTodo = inventoryService.save(item);

// ServletUriComponentsBuilder is having fromCurrentRequest() which is http://localhost:8080/users/vishnu/todos

// current pequest + path = http://localhost:8080/users/vishnu/todos + /id and buildAndExpand replace the id template with the getID()

// saveTo would create the id value and getID() would get the ID and replace the placeholder using buildAndExpand and convert it to Uri

URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(saveTodo.getId()).toUri();

// created would send 201

return ResponseEntity.created(location).build();

}

@PutMapping(path = "/admin/inventory/{id}")

public ResponseEntity updateFood(@PathVariable long id, @Valid @RequestBody Inventory item) {

System.out.println("inside the put method");

Inventory saveTodo = inventoryService.save(item);

URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(saveTodo.getId()).toUri();

return ResponseEntity.created(location).build();

}

@GetMapping (path = "/admin/inventory/{hotel\_address}")

public List<Inventory> getAllFoods(@PathVariable String hotel\_address){

System.out.println("username: " + hotel\_address);

return inventoryService.findAll();

}

@GetMapping(path="/admin/inventory/search/{food\_name}")

public List<Inventory> getFoodByName(@PathVariable String food\_name) {

Inventory foodname = inventoryService.findByFoodName(food\_name);

if(foodname == null) {

throw new TodoNotFoundException("foodname - " + food\_name);

}

List items = new ArrayList<>();

items.add(foodname);

// System.out.println(items.getClass().getName());

return items;

}

@GetMapping (path = "/admin/inventory/{hotel\_address}/{id}")

public Inventory getFood(@PathVariable String hotel\_address, @PathVariable long id) {

Inventory theTodo = inventoryService.findById(id);

if(theTodo == null) {

throw new TodoNotFoundException("id - " + id);

}

return theTodo;

}

}