Backend

package com.simplilearn.foodbox.config;

import com.simplilearn.foodbox.entity.User;

import com.simplilearn.foodbox.entity.Cuisine;

import com.simplilearn.foodbox.entity.Product;

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

import org.springframework.context.annotation.Configuration;

import org.springframework.data.rest.core.config.RepositoryRestConfiguration;

import org.springframework.data.rest.webmvc.config.RepositoryRestConfigurer;

import org.springframework.http.HttpMethod;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import javax.persistence.EntityManager;

import javax.persistence.metamodel.EntityType;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

@Configuration

public class MyDataRestConfig implements RepositoryRestConfigurer {

private EntityManager entityManager;

@Autowired

public MyDataRestConfig(EntityManager theEntityManager) {

entityManager = theEntityManager;

}

@Override

public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

HttpMethod[] theUnsupportedActions = {HttpMethod.PUT, HttpMethod.POST, HttpMethod.DELETE};

// disable HTTP methods for Product: PUT, POST and DELETE

config.getExposureConfiguration()

.forDomainType(Product.class)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

// disable HTTP methods for Cuisine: PUT, POST and DELETE

config.getExposureConfiguration()

.forDomainType(Cuisine.class)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

// disable HTTP methods for User: PUT, POST and DELETE

config.getExposureConfiguration()

.forDomainType(User.class)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

// call an internal helper method

exposeIds(config);

}

private void exposeIds(RepositoryRestConfiguration config) {

// expose entity ids

//

// - get a list of all entity classes from the entity manager

Set<EntityType<?>> entities = entityManager.getMetamodel().getEntities();

// - create an array of the entity types

List<Class> entityClasses = new ArrayList<>();

// - get the entity types for the entities

for (EntityType tempEntityType : entities) {

entityClasses.add(tempEntityType.getJavaType());

}

// - expose the entity ids for the array of entity/domain types

Class[] domainTypes = entityClasses.toArray(new Class[0]);

config.exposeIdsFor(domainTypes);

}

}

package com.simplilearn.foodbox.controller;

import com.simplilearn.foodbox.dao.ProductRepository;

import com.simplilearn.foodbox.entity.Product;

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

import org.springframework.data.rest.webmvc.ResourceNotFoundException;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

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

@RestController

@RequestMapping("/api/v1/")

public class ProductController {

@Autowired

private ProductRepository productRepository;

// get all products

@GetMapping("/products")

public List<Product> getAllproducts(){

return productRepository.findAll();

}

// create product rest api

@PostMapping("/products")

public Product createProduct(@RequestBody Product product) {

return productRepository.save(product);

}

// get product by id rest api

@GetMapping("/products/{id}")

public ResponseEntity<Product> getproductById(@PathVariable Long id) {

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("product not exist with id :" + id));

return ResponseEntity.ok(product);

}

// update product rest api

@PutMapping("/products/{id}")

public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product productDetails){

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("product not exist with id :" + id));

product.setName(productDetails.getName());

product.setDescription(productDetails.getDescription());

product.setPrice(productDetails.getPrice());

product.setImageUrl(productDetails.getImageUrl());

Product updatedProduct = productRepository.save(product);

return ResponseEntity.ok(updatedProduct);

}

// delete product rest api

@DeleteMapping("/products/{id}")

public ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id){

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("product not exist with id :" + id));

productRepository.delete(product);

Map<String, Boolean> response = new HashMap<>();

response.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(response);

}

}

package com.simplilearn.foodbox.controller;

import com.simplilearn.foodbox.dao.UserRepository;

import com.simplilearn.foodbox.entity.User;

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

import org.springframework.data.rest.webmvc.ResourceNotFoundException;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

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

@RestController

@RequestMapping("/api/v1/")

public class UserController {

@Autowired

private UserRepository userRepository;

// get all users

@GetMapping("/users")

public List<User> getAllUsers(){

return userRepository.findAll();

}

// create user rest api

@PostMapping("/users")

public User createUser(@RequestBody User user) {

return userRepository.save(user);

}

// get user by id rest api

@GetMapping("/users/{id}")

public ResponseEntity<User> getUserById(@PathVariable Long id) {

User user = userRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("user not exist with id :" + id));

return ResponseEntity.ok(user);

}

//

// // update user rest api

//

// @PutMapping("/users/{id}")

// public ResponseEntity<user> updateuser(@PathVariable Long id, @RequestBody user userDetails){

// user user = userRepository.findById(id)

// .orElseThrow(() -> new ResourceNotFoundException("user not exist with id :" + id));

//

// user.setFirstName(userDetails.getFirstName());

// user.setLastName(userDetails.getLastName());

// user.setEmailId(userDetails.getEmailId());

//

// user updateduser = userRepository.save(user);

// return ResponseEntity.ok(updateduser);

// }

//

// // delete user rest api

// @DeleteMapping("/users/{id}")

// public ResponseEntity<Map<String, Boolean>> deleteuser(@PathVariable Long id){

// user user = userRepository.findById(id)

// .orElseThrow(() -> new ResourceNotFoundException("user not exist with id :" + id));

//

// userRepository.delete(user);

// Map<String, Boolean> response = new HashMap<>();

// response.put("deleted", Boolean.TRUE);

// return ResponseEntity.ok(response);

// }

}

package com.simplilearn.foodbox.dao;

import com.simplilearn.foodbox.entity.Cuisine;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

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

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

public interface CuisineRepository extends JpaRepository<Cuisine, Long> {

}

package com.simplilearn.foodbox.dao;

import com.simplilearn.foodbox.entity.Product;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.data.jpa.repository.JpaRepository;

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

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

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

public interface ProductRepository extends JpaRepository<Product, Long> {

// all products

Page<Product> findAllByOrderByNameAsc(Pageable pageable);

Page<Product> findAllByOrderByDateCreatedDesc(Pageable pageable);

// by Cuisines

Page<Product> findByCuisine(@RequestParam("cuisine") int cuisine, Pageable pageable);

Page<Product> findByCuisineOrderByNameAsc(@RequestParam("cuisine") int cuisine, Pageable pageable);

Page<Product> findByCuisineOrderByDateCreatedDesc(@RequestParam("cuisine") int cuisine, Pageable pageable);

// for searching

Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

Page<Product> findByNameContainingOrderByNameAsc(@RequestParam("name") String name, Pageable pageable);

Page<Product> findByNameContainingOrderByDateCreatedDesc(@RequestParam("name") String name, Pageable pageable);

}

package com.simplilearn.foodbox.dao;

import com.simplilearn.foodbox.entity.User;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.data.jpa.repository.JpaRepository;

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

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

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

public interface UserRepository extends JpaRepository<User, Long> {

Page<User> findByUsername(@RequestParam("username") String username, Pageable pageable);

}

package com.simplilearn.foodbox.entity;

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name="address")

@Data

public class Address {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Long id;

}

package com.simplilearn.foodbox.entity;

import lombok.Getter;

import lombok.Setter;

import javax.persistence.\*;

@Entity

@Table(name="cuisine")

// @Data -- known bug

@Getter

@Setter

public class Cuisine {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Long id;

@Column(name = "cuisine\_name")

private String cuisine\_name;

public String getCuisine\_name() {

return cuisine\_name;

}

public void setCuisine\_name(String cuisine\_name) {

this.cuisine\_name = cuisine\_name;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

package com.simplilearn.foodbox.entity;

import lombok.Data;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.UpdateTimestamp;

import javax.persistence.\*;

import java.math.BigDecimal;

import java.util.Date;

@Entity

@Table(name="product")

@Data

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Long id;

@Column(name = "cuisine\_id")

private int cuisine;

@Column(name = "name")

private String name;

@Column(name = "description")

private String description;

@Column(name = "price")

private BigDecimal price;

@Column(name = "image\_url")

private String imageUrl;

@Column(name = "date\_created")

@CreationTimestamp

private Date dateCreated;

@Column(name = "last\_updated")

@UpdateTimestamp

private Date lastUpdated;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public BigDecimal getPrice() {

return price;

}

public void setPrice(BigDecimal price) {

this.price = price;

}

public String getImageUrl() {

return imageUrl;

}

public void setImageUrl(String imageUrl) {

this.imageUrl = imageUrl;

}

public Date getDateCreated() {

return dateCreated;

}

public void setDateCreated(Date dateCreated) {

this.dateCreated = dateCreated;

}

public Date getLastUpdated() {

return lastUpdated;

}

public void setLastUpdated(Date lastUpdated) {

this.lastUpdated = lastUpdated;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

package com.simplilearn.foodbox.entity;

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name="users")

@Data

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Long id;

@Column(name = "username")

private String username;

@Column(name = "email")

private String email;

@Column(name = "password")

private String password;

@Column(name = "phone")

private int phone;

@Column(name = "street")

private String street;

@Column(name = "city")

private String city;

@Column(name = "zipcode")

private String zipcode;

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public int getPhone() {

return phone;

}

public void setPhone(int phone) {

this.phone = phone;

}

public String getStreet() {

return street;

}

public void setStreet(String street) {

this.street = street;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getZipcode() {

return zipcode;

}

public void setZipcode(String zipcode) {

this.zipcode = zipcode;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

package com.simplilearn.foodbox;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FoodboxApplication {

public static void main(String[] args) {

SpringApplication.run(FoodboxApplication.class, args);

}

}

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3307/foodbox

spring.datasource.username=root

spring.data.rest.base-path=/api

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

package com.simplilearn.foodbox;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class FoodboxApplicationTests {

@Test

void contextLoads() {

}

}

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.5.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.simplilearn</groupId>

<artifactId>foodbox</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>foodbox</name>

<description>Full Stack Java Developer Capstone Project</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

Frontend

{

"$schema": "./node\_modules/@angular/cli/lib/config/schema.json",

"version": 1,

"newProjectRoot": "projects",

"projects": {

"foodbox": {

"projectType": "application",

"schematics": {

"@schematics/angular:application": {

"strict": true

}

},

"root": "",

"sourceRoot": "src",

"prefix": "app",

"architect": {

"build": {

"builder": "@angular-devkit/build-angular:browser",

"options": {

"outputPath": "dist/foodbox",

"index": "src/index.html",

"main": "src/main.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "tsconfig.app.json",

"assets": [

"src/favicon.ico",

"src/assets"

],

"styles": [

"src/styles.css",

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"node\_modules/@fortawesome/fontawesome-free/css/all.min.css"

],

"scripts": []

},

"configurations": {

"production": {

"budgets": [

{

"type": "initial",

"maximumWarning": "500kb",

"maximumError": "1mb"

},

{

"type": "anyComponentStyle",

"maximumWarning": "2kb",

"maximumError": "4kb"

}

],

"fileReplacements": [

{

"replace": "src/environments/environment.ts",

"with": "src/environments/environment.prod.ts"

}

],

"outputHashing": "all"

},

"development": {

"buildOptimizer": false,

"optimization": false,

"vendorChunk": true,

"extractLicenses": false,

"sourceMap": true,

"namedChunks": true

}

},

"defaultConfiguration": "production"

},

"serve": {

"builder": "@angular-devkit/build-angular:dev-server",

"configurations": {

"production": {

"browserTarget": "foodbox:build:production"

},

"development": {

"browserTarget": "foodbox:build:development"

}

},

"defaultConfiguration": "development"

},

"extract-i18n": {

"builder": "@angular-devkit/build-angular:extract-i18n",

"options": {

"browserTarget": "foodbox:build"

}

},

"test": {

"builder": "@angular-devkit/build-angular:karma",

"options": {

"main": "src/test.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "tsconfig.spec.json",

"karmaConfig": "karma.conf.js",

"assets": [

"src/favicon.ico",

"src/assets"

],

"styles": [

"src/styles.css"

],

"scripts": []

}

}

}

}

},

"defaultProject": "foodbox"

}

// Karma configuration file, see link for more information

// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {

config.set({

basePath: '',

frameworks: ['jasmine', '@angular-devkit/build-angular'],

plugins: [

require('karma-jasmine'),

require('karma-chrome-launcher'),

require('karma-jasmine-html-reporter'),

require('karma-coverage'),

require('@angular-devkit/build-angular/plugins/karma')

],

client: {

jasmine: {

// you can add configuration options for Jasmine here

// the possible options are listed at https://jasmine.github.io/api/edge/Configuration.html

// for example, you can disable the random execution with `random: false`

// or set a specific seed with `seed: 4321`

},

clearContext: false // leave Jasmine Spec Runner output visible in browser

},

jasmineHtmlReporter: {

suppressAll: true // removes the duplicated traces

},

coverageReporter: {

dir: require('path').join(\_\_dirname, './coverage/foodbox'),

subdir: '.',

reporters: [

{ type: 'html' },

{ type: 'text-summary' }

]

},

reporters: ['progress', 'kjhtml'],

port: 9876,

colors: true,

logLevel: config.LOG\_INFO,

autoWatch: true,

browsers: ['Chrome'],

singleRun: false,

restartOnFileChange: true

});

};

// Karma configuration file, see link for more information

// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {

config.set({

basePath: '',

frameworks: ['jasmine', '@angular-devkit/build-angular'],

plugins: [

require('karma-jasmine'),

require('karma-chrome-launcher'),

require('karma-jasmine-html-reporter'),

require('karma-coverage'),

require('@angular-devkit/build-angular/plugins/karma')

],

client: {

jasmine: {

// you can add configuration options for Jasmine here

// the possible options are listed at https://jasmine.github.io/api/edge/Configuration.html

// for example, you can disable the random execution with `random: false`

// or set a specific seed with `seed: 4321`

},

clearContext: false // leave Jasmine Spec Runner output visible in browser

},

jasmineHtmlReporter: {

suppressAll: true // removes the duplicated traces

},

coverageReporter: {

dir: require('path').join(\_\_dirname, './coverage/foodbox'),

subdir: '.',

reporters: [

{ type: 'html' },

{ type: 'text-summary' }

]

},

reporters: ['progress', 'kjhtml'],

port: 9876,

colors: true,

logLevel: config.LOG\_INFO,

autoWatch: true,

browsers: ['Chrome'],

singleRun: false,

restartOnFileChange: true

});

};

{

"name": "foodbox",

"version": "0.0.0",

"scripts": {

"ng": "ng",

"start": "ng serve",

"build": "ng build",

"watch": "ng build --watch --configuration development",

"test": "ng test"

},

"private": true,

"dependencies": {

"@angular/animations": "~12.1.1",

"@angular/common": "~12.1.1",

"@angular/compiler": "~12.1.1",

"@angular/core": "~12.1.1",

"@angular/forms": "~12.1.1",

"@angular/platform-browser": "~12.1.1",

"@angular/platform-browser-dynamic": "~12.1.1",

"@angular/router": "~12.1.1",

"@fortawesome/fontawesome-free": "^5.15.4",

"@ng-bootstrap/ng-bootstrap": "^10.0.0",

"bootstrap": "^5.1.0",

"rxjs": "~6.6.0",

"tslib": "^2.2.0",

"zone.js": "~0.11.4"

},

"devDependencies": {

"@angular-devkit/build-angular": "~12.1.1",

"@angular/cli": "~12.1.1",

"@angular/compiler-cli": "~12.1.1",

"@angular/localize": "^12.1.5",

"@types/jasmine": "~3.6.0",

"@types/node": "^12.11.1",

"jasmine-core": "~3.7.0",

"karma": "~6.3.0",

"karma-chrome-launcher": "~3.1.0",

"karma-coverage": "~2.0.3",

"karma-jasmine": "~4.0.0",

"karma-jasmine-html-reporter": "^1.5.0",

"typescript": "~4.3.2"

}

}