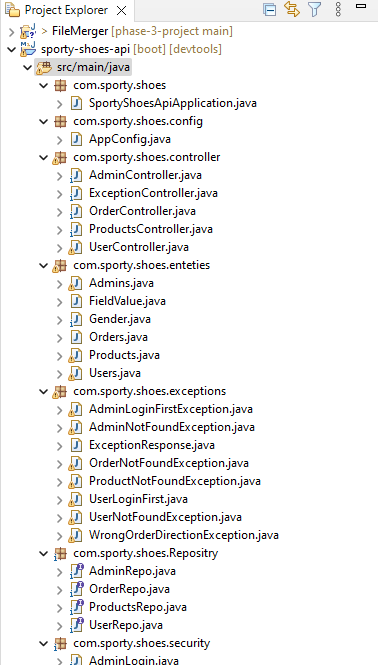
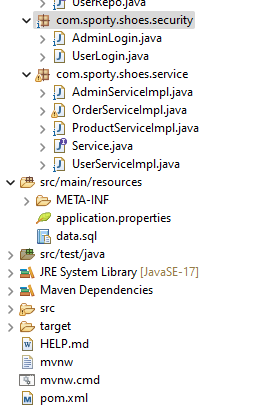
Source Code:

Maintain the following directory Structure:





**SportyShoesApiApplication.java**

package com.sporty.shoes;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SportyShoesApiApplication {

public static void main(String[] args) {

SpringApplication.run(SportyShoesApiApplication.class, args);

}

}

Admins.java

package com.sporty.shoes.enteties;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import com.fasterxml.jackson.annotation.JsonIgnore;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

@EqualsAndHashCode(onlyExplicitlyIncluded = true)

@AllArgsConstructor

@NoArgsConstructor

@Data

@Entity

public class Admins {

@EqualsAndHashCode.Include

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

int id;

@NotBlank

@Size(max=30)

String name;

@NotBlank

@Size(max=120)

String password;

}

FieldValue.java

package com.sporty.shoes.enteties;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

import lombok.ToString;

@EqualsAndHashCode(onlyExplicitlyIncluded = true)

@ToString

@Data

@NoArgsConstructor

@AllArgsConstructor

public class FieldValue {

Object FieldValue;

}

Gender.java

//package com.sporty.shoes.enteties;

//

//import java.util.HashMap;

//import java.util.Map;

//

//import javax.persistence.Entity;

//

//import lombok.Getter;

//

//just experimenting

//@Getter

//public enum Gender {

// MALE(1), FEMALE(2);

//

// private int code;

// private static Map<Integer, Gender> genderMap;

//

// Gender(int code) {

// this.code = code;

// }

//

//

//

// public static Gender getGender(int code) {

// if (genderMap == null) {

// genderMap = new HashMap<Integer, Gender>();

// genderMap.put(1, Gender.MALE);

// genderMap.put(2, Gender.FEMALE);

// }

// return genderMap.getOrDefault(code,null);

// }

//}

Orders.java

package com.sporty.shoes.enteties;

import java.time.LocalDateTime;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import org.hibernate.annotations.CreationTimestamp;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

@Entity

@NoArgsConstructor

@AllArgsConstructor

@Data

@EqualsAndHashCode(onlyExplicitlyIncluded = true)

public class Orders {

@EqualsAndHashCode.Include

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

int id;

@OneToOne

Products product;

@OneToOne

Users users;

@CreationTimestamp

LocalDateTime orderDate;

}

Products.java

package com.sporty.shoes.enteties;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.Pattern;

import javax.validation.constraints.PositiveOrZero;

import javax.validation.constraints.Size;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

@EqualsAndHashCode(onlyExplicitlyIncluded = true)

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Products {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@EqualsAndHashCode.Include

int id;

String name;

Double mrp;

int Size;

String category;

String colour;

//Gender gender;

@Pattern(regexp = "^(?:male|female)$",message = "gender can be male or female for shoes")

String gender;

@PositiveOrZero(message = "quantity can be +ve or 0")

int quantityInStock;

String productVendor;

}

Users.java

package com.sporty.shoes.enteties;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.validation.constraints.Email;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

import lombok.ToString;

@EqualsAndHashCode(onlyExplicitlyIncluded = true)

@AllArgsConstructor

@NoArgsConstructor

@Data

@Entity

@ToString

public class Users {

@EqualsAndHashCode.Include

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

int id;

String name;

String password;

}

AdminRepo.java

package com.sporty.shoes.Repositry;

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

import org.springframework.stereotype.Repository;

import com.sporty.shoes.enteties.Admins;

@Repository

public interface AdminRepo extends JpaRepository<Admins, Integer> {

}

OrderRepo.java

package com.sporty.shoes.Repositry;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.sporty.shoes.enteties.Orders;

import com.sporty.shoes.enteties.Products;

import com.sporty.shoes.enteties.Users;

@Repository

public interface OrderRepo extends JpaRepository<Orders,Integer> {

List<Orders> findByUsers(Users users );

List<Orders> findByProduct(Products products);

}

ProductsRepo.java

package com.sporty.shoes.Repositry;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.sporty.shoes.enteties.Products;

@Repository

public interface ProductsRepo extends JpaRepository<Products, Integer>{

List<Products> findByNameContaining(String name);

List<Products> findAllByOrderByCategoryAsc();

}

UserRepo.java

package com.sporty.shoes.Repositry;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.sporty.shoes.enteties.Users;

@Repository

public interface UserRepo extends JpaRepository<Users, Integer>{

List<Users> findByNameContaining(String name);

}

AdminServiceImpl.java

package com.sporty.shoes.service;

import java.lang.reflect.Field;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.util.ReflectionUtils;

import com.sporty.shoes.Repositry.AdminRepo;

import com.sporty.shoes.enteties.Admins;

import com.sporty.shoes.exceptions.AdminNotFoundException;

@Service

public class AdminServiceImpl implements com.sporty.shoes.service.Service<Admins> {

@Autowired

AdminRepo adminRepo;

@Override

public Admins save(Admins obj) {

return adminRepo.saveAndFlush(obj);

}

@Override

public List<Admins> getAll() {

return adminRepo.findAll();

}

@Override

public Admins get(int id) {

return adminRepo.findById(id).orElseThrow(() -> new AdminNotFoundException("No Admin with id = "+id));

}

@Override

public void delete(int id) {

adminRepo.deleteById(id);

}

@Override

public Admins update(int id, String fieldname, Object FieldValue) {

Admins admins=get(id);

Field field=ReflectionUtils.findField(admins.getClass(), fieldname);

field.setAccessible(true);

ReflectionUtils.setField(field, admins, FieldValue);

adminRepo.saveAndFlush(admins);

return admins;

}

}

OrderServiceImpl.java

package com.sporty.shoes.service;

import java.lang.reflect.Field;

import java.util.List;

import javax.persistence.criteria.Order;

import javax.transaction.Transactional;

import org.apache.el.util.ReflectionUtil;

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

import org.springframework.data.domain.Sort;

import org.springframework.data.domain.Sort.Direction;

import org.springframework.util.ReflectionUtils;

import com.sporty.shoes.Repositry.OrderRepo;

import com.sporty.shoes.Repositry.ProductsRepo;

import com.sporty.shoes.enteties.Orders;

import com.sporty.shoes.enteties.Products;

import com.sporty.shoes.enteties.Users;

import com.sporty.shoes.exceptions.OrderNotFoundException;

@org.springframework.stereotype.Service

@Transactional

public class OrderServiceImpl implements Service<Orders>{

@Autowired

OrderRepo orderRepo;

@Autowired

ProductServiceImpl productServiceImpl;

public List<Orders> getAllOrderedByFieldWithDirn(Direction direction,String ... fieldsname) {

return orderRepo.findAll(Sort.by(direction, fieldsname));

}

@Override

public List<Orders> getAll() {

return orderRepo.findAll();

}

@Override

public Orders get(int id) {

return orderRepo.findById(id).orElseThrow(() -> new OrderNotFoundException("No order with id = "+id));

}

@Override

public void delete(int id) {

orderRepo.deleteById(id);

}

@Override

public Orders update(int id, String fieldname, Object FieldValue) {

Orders orders=get(id);

Field field=ReflectionUtils.findField(Order.class, fieldname);

field.setAccessible(true);

ReflectionUtils.setField(field, orders, FieldValue);

return orders;

}

@Override

public Orders save(Orders obj) {

orderRepo.saveAndFlush(obj);

return obj;

}

public List<Orders> getAllOrdersByUser(Users users)

{

return orderRepo.findByUsers(users);

}

public List<Orders> orderByProuctid(int productid)

{

Products products=productServiceImpl.get(productid);

return orderRepo.findByProduct(products);

}

}

ProductServiceImpl.java

package com.sporty.shoes.service;

import java.lang.reflect.Field;

import java.util.List;

import javax.transaction.Transactional;

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

//import org.springframework.cglib.core.ReflectUtils;

import org.springframework.util.ReflectionUtils;

import com.sporty.shoes.Repositry.ProductsRepo;

import com.sporty.shoes.enteties.Products;

import com.sporty.shoes.exceptions.ProductNotFoundException;

@Transactional

@org.springframework.stereotype.Service

public class ProductServiceImpl implements Service<Products> {

@Autowired

ProductsRepo productsRepo;

@Override

public List<Products> getAll() {

return productsRepo.findAll();

}

@Override

public Products get(int id) {

return productsRepo.findById(id).orElseThrow(() -> new ProductNotFoundException("No Product with id " + id));

}

@Override

public void delete(int id) {

productsRepo.deleteById(id);

}

@Override

public Products update(int id, String fieldname, Object FieldValue) {

Products p = productsRepo.findById(id)

.orElseThrow(() -> new ProductNotFoundException("No Product with id = " + id));

Field field = ReflectionUtils.findField(p.getClass(), fieldname);

field.setAccessible(true);

ReflectionUtils.setField(field, p, FieldValue);

productsRepo.saveAndFlush(p);

return p;

}

public List<Products> getAllByName(String name) {

return productsRepo.findByNameContaining(name);

}

public List<Products> getAllBygroupedByCategory() {

return productsRepo.findAllByOrderByCategoryAsc();

}

@Override

public Products save(Products obj) {

productsRepo.saveAndFlush(obj);

return obj;

}

}

Service.java

package com.sporty.shoes.service;

import java.util.List;

public interface Service<E> {

E save(E obj);

List <E> getAll();

E get(int id);

void delete(int id);

E update(int id,String fieldname,Object FieldValue);

}

UserServiceImpl.java

package com.sporty.shoes.service;

import java.lang.reflect.Field;

import java.util.List;

import javax.transaction.Transactional;

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

import org.springframework.util.ReflectionUtils;

import com.sporty.shoes.Repositry.UserRepo;

import com.sporty.shoes.enteties.Users;

import com.sporty.shoes.exceptions.UserNotFoundException;

@org.springframework.stereotype.Service

@Transactional

public class UserServiceImpl implements Service<Users> {

@Autowired

UserRepo userRepo;

@Override

public List<Users> getAll() {

return userRepo.findAll();

}

@Override

public Users get(int id) {

return userRepo.findById(id).orElseThrow(() -> new UserNotFoundException("no user with id = " + id));

}

@Override

public void delete(int id) {

userRepo.deleteById(id);

}

@Override

public Users update(int id, String fieldname, Object FieldValue) {

Users users = userRepo.findById(id).orElseThrow(() -> new UserNotFoundException("no user with id = " + id));

Field field = ReflectionUtils.findField(users.getClass(), fieldname);

field.setAccessible(true);

ReflectionUtils.setField(field, users, FieldValue);

userRepo.save(users);

return users;

}

@Override

public Users save(Users obj) {

userRepo.saveAndFlush(obj);

return obj;

}

public List<Users> getUsesrByName(String name)

{

return userRepo.findByNameContaining(name);

}

}

AdminController.java

package com.sporty.shoes.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

import com.sporty.shoes.enteties.Admins;

import com.sporty.shoes.enteties.FieldValue;

import com.sporty.shoes.enteties.Users;

import com.sporty.shoes.exceptions.AdminLoginFirstException;

import com.sporty.shoes.security.AdminLogin;

import com.sporty.shoes.service.AdminServiceImpl;

import com.sporty.shoes.service.UserServiceImpl;

@RestController

@RequestMapping("/admins")

public class AdminController {

@Autowired

AdminServiceImpl adminServiceImpl;

@Autowired

UserServiceImpl userServiceImpl;

@Autowired

AdminLogin adminLogin;

@PatchMapping("/update/password")

public ResponseEntity<String> updatePassword(@RequestBody FieldValue fielvalue) {

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

adminServiceImpl.update(adminLogin.getAdminid(), "password", fielvalue.getFieldValue());

return new ResponseEntity<String>("Admin password updated sucessfully", HttpStatus.OK);

}

@GetMapping("/users/all")

public ResponseEntity<List<Users>> getAllUsers() {

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

return new ResponseEntity<List<Users>>(userServiceImpl.getAll(), HttpStatus.OK);

}

@GetMapping("/users/search")

public ResponseEntity<List<Users>> getAllUsersByName(@RequestParam String name) {

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

return new ResponseEntity<List<Users>>(userServiceImpl.getUsesrByName(name), HttpStatus.OK);

}

@PostMapping("/signin")

public ResponseEntity<String> Signin(@RequestBody Admins admins)

{

Admins admins2= adminServiceImpl.get(admins.getId());

if(admins2.getPassword().equals(admins2.getPassword()))

{

adminLogin.setAdminid(admins.getId());

adminLogin.setAdminLoggedin(true);

return new ResponseEntity<String>("Admin logged in sucessfully",HttpStatus.OK);

}

return new ResponseEntity<String>("in correct password",HttpStatus.BAD\_REQUEST);

}

@GetMapping("/signout")

public ResponseEntity<String> Signout()

{

adminLogin.setAdminLoggedin(false);

return new ResponseEntity<String>("Admin signed Out sucsessfully",HttpStatus.OK);

}

}

ExceptionController.java

package com.sporty.shoes.controller;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

import com.sporty.shoes.exceptions.ExceptionResponse;

@RestControllerAdvice

public class ExceptionController {

@ExceptionHandler

public ResponseEntity<?> handleException(Exception E)

{

ExceptionResponse ER = new ExceptionResponse(E.getClass().getSimpleName(),E.getMessage() );

return new ResponseEntity<ExceptionResponse> (ER,HttpStatus.NOT\_FOUND);

}

}

OrderController.java

package com.sporty.shoes.controller;

import java.util.List;

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

import org.springframework.data.domain.Sort.Direction;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

import com.sporty.shoes.enteties.Orders;

import com.sporty.shoes.exceptions.AdminLoginFirstException;

import com.sporty.shoes.exceptions.WrongOrderDirectionException;

import com.sporty.shoes.security.AdminLogin;

import com.sporty.shoes.service.OrderServiceImpl;

@RestController

@RequestMapping("/orders")

public class OrderController {

@Autowired

OrderServiceImpl orderServiceImpl;

@Autowired

AdminLogin adminLogin;

@GetMapping("/all/sorted")

ResponseEntity<List<Orders>> getAllOrdersByfieldWithDirection(@RequestParam String key,@RequestParam String ordering) throws WrongOrderDirectionException{

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

if(ordering.equals("asc"))

return new ResponseEntity<List<Orders>>(

orderServiceImpl.getAllOrderedByFieldWithDirn(Direction.ASC, key)

,HttpStatus.OK);

else if(ordering.equals("desc"))

return new ResponseEntity<List<Orders>>(

orderServiceImpl.getAllOrderedByFieldWithDirn(Direction.DESC, key)

,HttpStatus.OK);

else throw new WrongOrderDirectionException("Direction Order can be only asc or desc");

}

@GetMapping("/{productsid}/order/all")

ResponseEntity<List<Orders>> getAllOrdersByProductId(@PathVariable int productsid)

{

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

return new ResponseEntity<List<Orders>>( orderServiceImpl.orderByProuctid(productsid),HttpStatus.FOUND);

}

}

ProductsController.java

package com.sporty.shoes.controller;

import java.lang.reflect.Method;

import java.util.List;

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

import org.springframework.hateoas.EntityModel;

import org.springframework.hateoas.Link;

import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

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

import com.sporty.shoes.enteties.FieldValue;

import com.sporty.shoes.enteties.Products;

import com.sporty.shoes.exceptions.AdminLoginFirstException;

import com.sporty.shoes.security.AdminLogin;

import com.sporty.shoes.service.ProductServiceImpl;

@RestController

@RequestMapping("/products")

public class ProductsController {

@Autowired

AdminLogin adminLogin;

@Autowired

ProductServiceImpl productServiceImpl;

@GetMapping("/all")

public ResponseEntity<List<Products>> getAllProducts() {

return new ResponseEntity<List<Products>>(productServiceImpl.getAll(), HttpStatus.OK);

}

@GetMapping("/allGroupedByCategories")

public ResponseEntity<List<Products>> getAllProductsgroupedByCategories() {

return new ResponseEntity<List<Products>>(productServiceImpl.getAllBygroupedByCategory(), HttpStatus.OK);

}

@GetMapping("/searchByName")

public ResponseEntity<List<Products>> getAllProductsByName(@RequestParam("name") String name) {

return new ResponseEntity<List<Products>>(productServiceImpl.getAllByName(name), HttpStatus.OK);

}

@GetMapping("/{id}")

public ResponseEntity<Products> getProductById(@PathVariable int id) {

return new ResponseEntity<Products>(productServiceImpl.get(id), HttpStatus.OK);

}

@PostMapping("/add")

public ResponseEntity<EntityModel<Products>> addProduct(@RequestBody Products product) throws NoSuchMethodException, SecurityException

{

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

productServiceImpl.save(product);

EntityModel<Products> entityModel= EntityModel.of(product);

Method method=this.getClass().getMethod("getAllProducts");

method.setAccessible(true);

Link linktoOrders =WebMvcLinkBuilder.linkTo(method).withSelfRel();;

entityModel.add(linktoOrders);

return new ResponseEntity<EntityModel<Products>>( entityModel,HttpStatus.CREATED);

}

@PatchMapping("/{productid}/update/{fieldname}")

public ResponseEntity<Products> updateProductname(@PathVariable int productid,@RequestBody FieldValue fieldValue,@PathVariable String fieldname)

{

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

Products products= productServiceImpl.update(productid,fieldname,fieldValue.getFieldValue());

return new ResponseEntity<Products>(products,HttpStatus.OK);

}

@DeleteMapping("/{productid}/delete")

public ResponseEntity<String> deleteProductById(@PathVariable int productid)

{

if (adminLogin.isAdminLoggedin() == false)

throw new AdminLoginFirstException("Login As Admin First");

productServiceImpl.delete(productid);

return new ResponseEntity<String>("Product deleted sucessfully",HttpStatus.OK);

}

}

UserController.java

package com.sporty.shoes.controller;

import java.lang.reflect.Method;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

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

import org.springframework.hateoas.EntityModel;

import org.springframework.hateoas.Link;

import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

import com.sporty.shoes.enteties.Admins;

import com.sporty.shoes.enteties.FieldValue;

import com.sporty.shoes.enteties.Orders;

import com.sporty.shoes.enteties.Products;

import com.sporty.shoes.enteties.Users;

import com.sporty.shoes.exceptions.UserLoginFirst;

import com.sporty.shoes.security.UserLogin;

import com.sporty.shoes.service.OrderServiceImpl;

import com.sporty.shoes.service.ProductServiceImpl;

import com.sporty.shoes.service.UserServiceImpl;

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

UserServiceImpl userServiceImpl;

@Autowired

ProductServiceImpl productServiceImpl;

@Autowired

OrderServiceImpl orderServiceImpl;

@Autowired

UserLogin userLogin;

@GetMapping("/")

ResponseEntity<Users> getUser()

{

if(userLogin.isUserLoggedin()==false)

throw new UserLoginFirst("please Login First As User");

return new ResponseEntity<Users>( userServiceImpl.get(userLogin.getUserid()),HttpStatus.OK);

}

@PostMapping("/product/{productid}/order")

public ResponseEntity<EntityModel<Orders>> createOrder( @PathVariable int productid) throws NoSuchMethodException, SecurityException

{

if(userLogin.isUserLoggedin()==false)

throw new UserLoginFirst("please Login First As User");

Users user=userServiceImpl.get(userLogin.getUserid());

Products product=productServiceImpl.get(productid);

productServiceImpl.update(productid,"quantityInStock", product.getQuantityInStock()-1);

Orders order=new Orders();

order.setProduct(product);

order.setUsers(user);

orderServiceImpl.save(order);

EntityModel<Orders> entityModel= EntityModel.of(order);

Method method=this.getClass().getMethod("getAllOrders",Integer.class);

method.setAccessible(true);

Link linktoOrders =WebMvcLinkBuilder.linkTo(method,userLogin.getUserid()).withSelfRel();;

entityModel.add(linktoOrders);

return new ResponseEntity<EntityModel<Orders>>(entityModel,HttpStatus.CREATED);

}

@GetMapping("/order/all")

public ResponseEntity <List<Orders> >getAllOrders()

{

if(userLogin.isUserLoggedin()==false)

throw new UserLoginFirst("please Login First As User");

Users users=userServiceImpl.get(userLogin.getUserid());

return new ResponseEntity<List<Orders>>(orderServiceImpl.getAllOrdersByUser(users),HttpStatus.OK);

}

@PatchMapping("/update/password")

public ResponseEntity <String >updatePassword(@RequestBody FieldValue fielvalue)

{

if(userLogin.isUserLoggedin()==false)

throw new UserLoginFirst("please Login First As User");

Users users=userServiceImpl.update(userLogin.getUserid(), "password",fielvalue.getFieldValue());

return new ResponseEntity<String>("User's password updated",HttpStatus.OK);

}

@PatchMapping("/update/name")

public ResponseEntity <String >updateName(@RequestBody FieldValue fielvalue)

{

Users users=userServiceImpl.update(userLogin.getUserid(), "name",fielvalue.getFieldValue());

return new ResponseEntity<String>("Username updated updated",HttpStatus.OK);

}

@PostMapping("/signup")

public ResponseEntity<Users> signupUp(@RequestBody Users users)

{

if(userLogin.isUserLoggedin()==true)

throw new UserLoginFirst("please logout first as a userss");

System.out.println(users);

users=userServiceImpl.save(users);

return new ResponseEntity<Users>(users,HttpStatus.CREATED);

}

@GetMapping("/signout")

public ResponseEntity<String> Signout()

{

if(userLogin.isUserLoggedin()==false)

throw new UserLoginFirst("please Login First As User");

userLogin.setUserLoggedin(false);

return new ResponseEntity<String>("User signed Out sucsessfully",HttpStatus.OK);

}

@PostMapping("/signin")

public ResponseEntity<String> Signin(@RequestBody Users user)

{

Users user2= userServiceImpl.get(user.getId());

if(user2.getPassword().equals(user.getPassword()))

{

userLogin.setUserid(user.getId());

userLogin.setUserLoggedin(true);

return new ResponseEntity<String>("User logged in sucessfully",HttpStatus.OK);

}

return new ResponseEntity<String>("in correct password",HttpStatus.BAD\_REQUEST);

}

}

AdminLoginFirstException.java

package com.sporty.shoes.exceptions;

public class AdminLoginFirstException extends RuntimeException {

public AdminLoginFirstException(String msg) {

super(msg);

}

}

AdminNotFoundException.java

package com.sporty.shoes.exceptions;

public class AdminNotFoundException extends RuntimeException{

public AdminNotFoundException(String msg) {

super(msg);

}

}

ExceptionResponse.java

package com.sporty.shoes.exceptions;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class ExceptionResponse {

String ExceptionClass;

String ExceptionMsg;

}

OrderNotFoundException.java

package com.sporty.shoes.exceptions;

public class OrderNotFoundException extends RuntimeException {

public OrderNotFoundException(String msg) {

super(msg);

}

}

ProductNotFoundException.java

package com.sporty.shoes.exceptions;

public class ProductNotFoundException extends RuntimeException{

public ProductNotFoundException(String msg) {

super(msg);

}

}

UserLoginFirst.java

package com.sporty.shoes.exceptions;

public class UserLoginFirst extends RuntimeException {

public UserLoginFirst(String msg)

{super(msg);}

}

UserNotFoundException.java

package com.sporty.shoes.exceptions;

public class UserNotFoundException extends RuntimeException{

public UserNotFoundException(String msg) {

super(msg);

}

}

WrongOrderDirectionException.java

package com.sporty.shoes.exceptions;

public class WrongOrderDirectionException extends RuntimeException {

public WrongOrderDirectionException(String msg) {

super(msg);

}

}

AppConfig.java

package com.sporty.shoes.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.sporty.shoes.security.AdminLogin;

import com.sporty.shoes.security.UserLogin;

@Configuration

public class AppConfig {

@Bean

AdminLogin adminLogin()

{

return new AdminLogin(false,0);

}

@Bean

UserLogin userLogin()

{

return new UserLogin(false,0);

}

}

AdminLogin.java

package com.sporty.shoes.security;

import lombok.Data;

@Data

public class AdminLogin {

boolean adminLoggedin;

int Adminid;

public AdminLogin(boolean b,int Adminid) {

this.adminLoggedin=b;

this.Adminid=Adminid;

}

}

UserLogin.java

package com.sporty.shoes.security;

import lombok.Data;

@Data

public class UserLogin {

boolean userLoggedin;

int userid;

public UserLogin(boolean b,int userid)