Source Code:

KitchenStoryApplication.java

package com.ecommerce;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication

@EnableJpaRepositories(basePackages = "com.ecommerce.repository")

public class KitchenStoryApplication {

public static void main(String[] args) {

SpringApplication.run(KitchenStoryApplication.class, args);

}

}

ProductController.java

package com.ecommerce.controller;

import java.util.List;

import java.util.Optional;

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

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.RequestBody;

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

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

import com.ecommerce.model.Cart;

import com.ecommerce.model.CartItem;

import com.ecommerce.model.CreateProduct;

import com.ecommerce.model.OrderItem;

import com.ecommerce.model.Orders;

import com.ecommerce.model.Product;

import com.ecommerce.model.ProductDetail;

import com.ecommerce.model.User;

import com.ecommerce.model.UserAddProduct;

import com.ecommerce.model.UserBuy;

import com.ecommerce.model.UserRemoveProduct;

import com.ecommerce.repository.CartItemRepository;

import com.ecommerce.repository.CartRepository;

import com.ecommerce.repository.OrderItemRepository;

import com.ecommerce.repository.OrdersRepository;

import com.ecommerce.repository.ProductDetailRepository;

import com.ecommerce.repository.ProductRepository;

import com.ecommerce.repository.UserRepository;

@RestController

@RequestMapping("/home")

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

public class ProductController {

@Autowired

UserRepository userrepo;

@Autowired

ProductRepository productrepo;

@Autowired

ProductDetailRepository productdetailrepo;

@Autowired

CartRepository cartrepo;

@Autowired

CartItemRepository cartitemrepo;

@Autowired

OrdersRepository orderrepo;

@Autowired

OrderItemRepository orderitemrepo;

@GetMapping("products")

public List<Product> getAllProduct(){

return (List<Product>) productrepo.findAll();

}

@PostMapping("add")

public Cart userAddtoCart(@RequestBody UserAddProduct uap) throws Exception {

//Need to find cart of user but cart needs to be active cart not all list of carts

//Search userid in cart

long userid = uap.getUserid();

long productid = uap.getProductid();

User tempuser = userrepo.findById(userid).orElse(null);

Product tempproduct = productrepo.findById(productid).orElse(null);

Cart tempcart = cartrepo.findByUserUserid(userid);

CartItem newItem = cartitemrepo.findCartItemByProductProductid(productid);

int buyquantity = uap.getQuantity();

if (buyquantity <= tempproduct.getProductquantity() && (tempproduct!=null) && (tempuser!=null)) {

int totalnum = tempproduct.getProductquantity() - buyquantity;

tempproduct.setProductquantity(totalnum);

productrepo.save(tempproduct);

if (newItem != null) {//cartitem exists

newItem.setCartitemquantity(newItem.getCartitemquantity()+ buyquantity);

newItem.setTotalprice(newItem.getTotalprice()+(buyquantity \* tempproduct.getPrice()));

cartitemrepo.save(newItem);

tempcart.setTotalprice(tempcart.getTotalprice()+(buyquantity \* (tempproduct.getPrice())));

return cartrepo.save(tempcart);

}

//If cart doesnt exist

newItem = new CartItem();

newItem.setCart(tempcart);

newItem.setProduct(tempproduct);

newItem.setCartitemquantity(buyquantity);

newItem.setItemprice(tempproduct.getPrice());

newItem.setTotalprice(buyquantity \* tempproduct.getPrice());

cartitemrepo.save(newItem);

tempcart.setTotalprice(tempcart.getTotalprice()+(buyquantity \* tempproduct.getPrice()));

tempcart.setStatus("In Progress");

return cartrepo.save(tempcart);

}else {

throw new Exception("Cannot add item - Not enough of the item");

}

}

@PostMapping("remove")

public Cart userRemovefromCart(@RequestBody UserRemoveProduct urp) throws Exception {

long userid = urp.getUserid();

long productid = urp.getProductid();

long cartid = urp.getCartid();

int removequantity = urp.getQuantity();

User tempuser = userrepo.findById(userid).orElse(null);

Product tempproduct = productrepo.findById(productid).orElse(null);

Cart tempcart = cartrepo.findById(cartid).orElse(null);

List<CartItem> tempcartitems = cartitemrepo.findCartItemByCartCartid(cartid);

if(tempuser != null && tempproduct != null && tempcart != null) {

for(int i= 0; i<tempcartitems.size();i++) {

CartItem tempcartitem = tempcartitems.get(i);

if ((tempcartitem.getCart().getCartid() == tempuser.getCart().getCartid()) && (tempcartitem.getProduct().getProductid() == productid)){

int cartquant = tempcartitem.getCartitemquantity();

if(cartquant-removequantity == 0) {

cartitemrepo.deleteById(tempcartitem.getCartitemid());

}else {

tempcartitem.setCartitemquantity(cartquant-removequantity);

tempcartitem.setTotalprice((cartquant-removequantity) \* tempproduct.getPrice());

cartitemrepo.save(tempcartitem);

}

tempproduct.setProductquantity(tempproduct.getProductquantity()+removequantity);//update quantity

productrepo.save(tempproduct);

float itemprice = tempcartitem.getProduct().getPrice();

tempcart.setTotalprice(tempcart.getTotalprice()-(removequantity \* itemprice));//update total price

return cartrepo.save(tempcart); // update cart

}

}

throw new Exception("Remove item error - User does not match cart");

}

else {

throw new Exception("Remove item error");

}

}

@PostMapping("search")

public List<Product> searchProduct(@RequestBody Product product){

return (List<Product>) productrepo.findByProductname(product.getProductname());

}

//Get all cart items in cart

@PostMapping("mycartitems")

public List<CartItem> showUserCart(@RequestBody User user) throws Exception{

User findUser = userrepo.findById(user.getUserid()).orElse(null);

if (findUser != null) {

List<CartItem> cartitems = cartitemrepo.findCartItemByCartCartid(findUser.getCart().getCartid());

return cartitems;

}else {

throw new Exception("Cannot show cart - User not found");

}

}

//Get User cart

@GetMapping("getcart/{id}")

public Cart showCart(@PathVariable long id) throws Exception {

Cart tempcart = cartrepo.findByUserUserid(id);

if(tempcart!=null) {

return tempcart;

}else {

throw new Exception("Cannot get cart - wrong id");

}

}

@PostMapping("buy")

public Optional<Orders> buyProducts(@RequestBody UserBuy ub) throws Exception {

User tempuser = userrepo.findById(ub.getUserid()).orElse(null);

if (tempuser != null) {

List<CartItem> cartitems = tempuser.getCart().getCartitem();

Cart tempcart = tempuser.getCart();

if (cartitems.size()>0) {

Orders newOrder = new Orders();

newOrder.setAddress(ub.getAddress());

newOrder.setTotalprice(tempcart.getTotalprice());

newOrder.setUser(tempuser);

orderrepo.save(newOrder);

for (int i=0; i<cartitems.size();i++) {

OrderItem temporderitem = new OrderItem();

temporderitem.setItemprice(cartitems.get(i).getItemprice());

temporderitem.setOrderitemquantity(cartitems.get(i).getCartitemquantity());

temporderitem.setOrders(newOrder);

temporderitem.setProduct(cartitems.get(i).getProduct());

temporderitem.setItemprice(cartitems.get(i).getItemprice());

temporderitem.setTotalprice(cartitems.get(i).getTotalprice());

orderitemrepo.save(temporderitem);

}

tempcart.setStatus("Empty");

tempcart.setTotalprice(0);

tempcart.setCartitem(null);

cartrepo.save(tempcart);

for (int x=0; x<cartitems.size();x++) {

System.out.println("Cart item: "+x+" , "+cartitems.get(x).getCartitemid());

cartitemrepo.delete(cartitems.get(x));

}

return orderrepo.findById(newOrder.getOrderid());

}else {

throw new Exception("Buy error - Cart is empty");

}

}else {

throw new Exception("Buy error - User does not exist!");

}

}

//Search product by productid

@GetMapping("filtersearch/{id}")

public ProductDetail getProductFilter(@PathVariable long id) {

return productdetailrepo.findByProductProductid(id);

}

//Delete Product

@DeleteMapping("delete/{id}")

public void deleteProduct(@PathVariable long id) {

Product tempproduct = productrepo.findById(id).orElse(null);

ProductDetail tempproductdetail = productdetailrepo.findByProductProductid(id);

productdetailrepo.delete(tempproductdetail);

productrepo.delete(tempproduct);

}

//Create Product

@PostMapping("create")

public Product createProduct(@RequestBody CreateProduct product) {

Product temp= new Product();

temp.setPrice(product.getPrice());

temp.setProductname(product.getProductname());

temp.setProductquantity(product.getProductquantity());

temp.setUser(userrepo.findById(product.getUserid()).orElse(null));

productrepo.save(temp);

ProductDetail pdt = new ProductDetail();

pdt.setProduct(temp);

pdt.setType(product.getType());

productdetailrepo.save(pdt);

return temp;

}

}

UserController.java

package com.ecommerce.controller;

import java.util.List;

import java.util.Optional;

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

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.RestController;

import com.ecommerce.model.Cart;

import com.ecommerce.model.User;

import com.ecommerce.model.UserLogin;

import com.ecommerce.repository.CartItemRepository;

import com.ecommerce.repository.CartRepository;

import com.ecommerce.repository.OrderItemRepository;

import com.ecommerce.repository.OrdersRepository;

import com.ecommerce.repository.ProductRepository;

import com.ecommerce.repository.UserRepository;

@RestController

@RequestMapping("user")

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

public class UserController {

@Autowired

UserRepository userrepo;

@Autowired

ProductRepository productrepo;

@Autowired

CartRepository cartrepo;

@Autowired

CartItemRepository cartitemrepo;

@Autowired

OrdersRepository orderrepo;

@Autowired

OrderItemRepository orderitemrepo;

@PostMapping("allusers")

public List<User> getAllUsers(@RequestBody User user) throws Exception{

User tempuser = userrepo.findById(user.getUserid()).get();

boolean admincheck = tempuser.isAdminrights();

if (admincheck) {

List<User> userslist = (List<User>) userrepo.findAll();

return userslist;

}else {

throw new Exception("Cannot get all users - Not Admin");

}

}

@PostMapping("create")

public Optional<User> createNewUser(@RequestBody User user) {

User newUser = new User();

newUser = user;

Cart newCart = new Cart();

newCart.setUser(newUser);

newCart.setStatus("Empty");

cartrepo.save(newCart);

newUser.setCart(newCart);

userrepo.save(newUser);

return userrepo.findById(newUser.getUserid());

}

@GetMapping("getuser/{id}")

public User getOneUser(@PathVariable long id) {

return userrepo.findById(id).orElse(null);

}

@DeleteMapping("delete")

public void deleteUser(@RequestBody User user) throws Exception {

User tempuser = userrepo.findById(user.getUserid()).orElse(null);

Cart tempcart = cartrepo.findById(tempuser.getCart().getCartid()).orElse(null);

if(tempuser != null && tempcart != null) {

cartrepo.delete(tempcart);

userrepo.delete(tempuser);

}

else {

throw new Exception("Error deleting user");

}

}

@PostMapping("login")

public User loginUser(@RequestBody UserLogin user) throws Exception {

System.out.println("Email:"+user.getEmail());

System.out.println("Password:"+user.getPassword());

User usercheck = userrepo.findByEmail(user.getEmail());

if(usercheck.getPassword().equals(user.getPassword())) {

return usercheck;

}else {

throw new Exception("Login Error - Email/Password mismatched!");

}

}

@PutMapping("resetpassword")

public User resetPassword(@RequestBody User user) throws Exception {

User tempuser = userrepo.findById(user.getUserid()).orElse(null);

if (tempuser != null) {

tempuser.setPassword(user.getPassword());

return userrepo.save(tempuser);

}else {

throw new Exception("Error resetting password");

}

}

}

Cart.java

package com.ecommerce.model;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="cart")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler","user"})

public class Cart {

@Id @GeneratedValue

@Column(name="cartid")

private long cartid;

@OneToOne(cascade = CascadeType.ALL)

@JoinColumn(name = "userid", referencedColumnName = "userid")

@JsonIgnoreProperties("cart")

private User user;

@Column(name="status",columnDefinition = "varchar(30) default 'Empty'")

private String status;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "cart")

@JsonIgnoreProperties("cart")

private List<CartItem> cartitem;

@Column(name="totalprice",columnDefinition = "DECIMAL(10,2) default 0")

private float totalprice;

public float getTotalprice() {

return totalprice;

}

public void setTotalprice(float totalprice) {

this.totalprice = totalprice;

}

public long getCartid() {

return cartid;

}

public void setCartid(long cartid) {

this.cartid = cartid;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public List<CartItem> getCartitem() {

return cartitem;

}

public void setCartitem(List<CartItem> cartitem) {

this.cartitem = cartitem;

}

}

CartItem.java

package com.ecommerce.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="cartitem")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler","cart"})

public class CartItem {

@Id @GeneratedValue

@Column(name="cartitemid")

private long cartitemid;

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="cartid")

@JsonIgnoreProperties("cartitem")

private Cart cart;

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="productid")

@JsonIgnoreProperties("cartitem")

private Product product;

@Column(name="cartitemquantity")

private int cartitemquantity;

@Column(name="totalprice")

private float totalprice;

@Column(name="itemprice")

private float itemprice;

public float getTotalprice() {

return totalprice;

}

public void setTotalprice(float totalprice) {

this.totalprice = totalprice;

}

public float getItemprice() {

return itemprice;

}

public void setItemprice(float itemprice) {

this.itemprice = itemprice;

}

public long getCartitemid() {

return cartitemid;

}

public void setCartitemid(long cartitemid) {

this.cartitemid = cartitemid;

}

public Cart getCart() {

return cart;

}

public void setCart(Cart cart) {

this.cart = cart;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public int getCartitemquantity() {

return cartitemquantity;

}

public void setCartitemquantity(int cartitemquantity) {

this.cartitemquantity = cartitemquantity;

}

}

CreateProduct.java

package com.ecommerce.model;

public class CreateProduct {

private long userid;

private String productname;

private float price;

private int productquantity;

private String type;

public long getUserid() {

return userid;}

public void setUserid(long userid) {

this.userid = userid;}

public String getProductname() {

return productname;}

public void setProductname(String productname) {

this.productname = productname;}

public float getPrice() {

return price;}

public void setPrice(float price) {

this.price = price;}

public int getProductquantity() {

return productquantity;}

public void setProductquantity(int productquantity) {

this.productquantity = productquantity;}

public String getType() {

return type;}

public void setType(String type) {

this.type = type;}

}

OrderItem.java

package com.ecommerce.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="orderitem")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler","orders"})

public class OrderItem {

@Id @GeneratedValue

@Column(name="orderitem")

private long orderitemid;

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="orderid")

@JsonIgnoreProperties("orderitem")

private Orders orders;

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="productid")

@JsonIgnoreProperties("orderitem")

private Product product;

@Column(name="orderitemquantity")

private int orderitemquantity;

@Column(name="totalprice")

private float totalprice;

@Column(name="itemprice")

private float itemprice;

public float getTotalprice() {

return totalprice;

}

public void setTotalprice(float totalprice) {

this.totalprice = totalprice;

}

public float getItemprice() {

return itemprice;

}

public void setItemprice(float itemprice) {

this.itemprice = itemprice;

}

public long getOrderitemid() {

return orderitemid;

}

public void setOrderitemid(long orderitemid) {

this.orderitemid = orderitemid;

}

public Orders getOrders() {

return orders;

}

public void setOrders(Orders orders) {

this.orders = orders;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public int getOrderitemquantity() {

return orderitemquantity;

}

public void setOrderitemquantity(int orderitemquantity) {

this.orderitemquantity = orderitemquantity;

}

}

Orders.java

package com.ecommerce.model;

import java.util.List;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="orders")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler","user"})

public class Orders {

@Id @GeneratedValue

@Column(name="orderid")

private long orderid;

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="userid")

@JsonIgnoreProperties("orders")

private User user;

@Column(name="address")

private String address;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "orders")

@JsonIgnoreProperties("orders")

private List<OrderItem> orderitem;

@Column(name="totalprice",columnDefinition = "DECIMAL(10,2) default 0")

private float totalprice;

public float getTotalprice() {

return totalprice;

}

public void setTotalprice(float totalprice) {

this.totalprice = totalprice;

}

public long getOrderid() {

return orderid;

}

public void setOrderid(long orderid) {

this.orderid = orderid;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public List<OrderItem> getOrderitem() {

return orderitem;

}

public void setOrderitem(List<OrderItem> orderitem) {

this.orderitem = orderitem;

}

}

Product.java

package com.ecommerce.model;

import java.util.List;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="product")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler","cartitem"})

public class Product {

@Id @GeneratedValue

@Column(name="productid")

private long productid;

@ManyToOne

@JoinColumn(name="userid")

@JsonIgnoreProperties("product")

private User user;

@Column(name="productname")

private String productname;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "product")

@JsonIgnoreProperties("product")

private List<ProductDetail> productdetail;

@Column(name="price")

private float price;

@Column(name="productquantity")

private int productquantity;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "product")

@JsonIgnoreProperties("product")

private List<CartItem> cartitem;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "product")

@JsonIgnoreProperties("product")

private List<OrderItem> orderitem;

public long getProductid() {

return productid;

}

public void setProductid(long productid) {

this.productid = productid;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public String getProductname() {

return productname;

}

public void setProductname(String productname) {

this.productname = productname;

}

public List<ProductDetail> getProductdetail() {

return productdetail;

}

public void setProductdetail(List<ProductDetail> productdetail) {

this.productdetail = productdetail;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public int getProductquantity() {

return productquantity;

}

public void setProductquantity(int productquantity) {

this.productquantity = productquantity;

}

public List<CartItem> getCartitem() {

return cartitem;

}

public void setCartitem(List<CartItem> cartitem) {

this.cartitem = cartitem;

}

public List<OrderItem> getOrderitem() {

return orderitem;

}

public void setOrderitem(List<OrderItem> orderitem) {

this.orderitem = orderitem;

}

}

ProductDetail.java

package com.ecommerce.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="productdetail")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler","product"})

public class ProductDetail {

@Id @GeneratedValue

@Column(name="productdetailid")

private long productdetailid;

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="productid")

@JsonIgnoreProperties("productdetail")

private Product product;

@Column(name="type")

private String type;

public long getProductdetailid() {

return productdetailid;

}

public void setProductdetailid(long productdetailid) {

this.productdetailid = productdetailid;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

}

User.java

package com.ecommerce.model;

import java.util.List;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name="user1")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})

public class User {

@Id @GeneratedValue

@Column(name="userid")

private long userid;

@Column(name="firstname")

private String firstname;

@Column(name="lastname")

private String lastname;

@Column(name="mobile")

private String mobile;

@Column(name="email")

private String email;

@Column(name="password")

private String password;

@Column(name="adminrights",columnDefinition = "bit(1) default 0")

private boolean adminrights;

@OneToMany(fetch = FetchType.LAZY,mappedBy = "user")

@JsonIgnoreProperties("user")

private List<Product> product;

@OneToOne(mappedBy = "user")

@JsonIgnoreProperties("user")

private Cart cart;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "user")

@JsonIgnoreProperties("user")

private List<Orders> orders;

public long getUserid() {

return userid;

}

public void setUserid(long userid) {

this.userid = userid;

}

public String getFirstname() {

return firstname;

}

public void setFirstname(String firstname) {

this.firstname = firstname;

}

public String getLastname() {

return lastname;

}

public void setLastname(String lastname) {

this.lastname = lastname;

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

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 boolean isAdminrights() {

return adminrights;

}

public void setAdminrights(boolean adminrights) {

this.adminrights = adminrights;

}

public List<Product> getProduct() {

return product;

}

public void setProduct(List<Product> product) {

this.product = product;

}

public Cart getCart() {

return cart;

}

public void setCart(Cart cart) {

this.cart = cart;

}

public List<Orders> getOrders() {

return orders;

}

public void setOrders(List<Orders> orders) {

this.orders = orders;

}

}

UserAddProduct.java

package com.ecommerce.model;

public class UserAddProduct {

private long userid;

private long productid;

private int quantity;

public long getUserid() {

return userid;

}

public void setUserid(long userid) {

this.userid = userid;

}

public long getProductid() {

return productid;

}

public void setProductid(long productid) {

this.productid = productid;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

UserBuy.java

package com.ecommerce.model;

public class UserBuy {

private long userid;

private String address;

public long getUserid() {

return userid;

}

public void setUserid(long userid) {

this.userid = userid;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

UserLogin.java

package com.ecommerce.model;

public class UserLogin {

private String email;

private String password;

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;}

}

UserRemoveProduct.java

package com.ecommerce.model;

public class UserRemoveProduct {

private long userid;

private long productid;

private long cartid;

private int quantity;

public long getUserid() {

return userid;

}

public void setUserid(long userid) {

this.userid = userid;

}

public long getProductid() {

return productid;

}

public void setProductid(long productid) {

this.productid = productid;

}

public long getCartid() {

return cartid;

}

public void setCartid(long cartid) {

this.cartid = cartid;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

CartItemRepository.java

package com.ecommerce.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.ecommerce.model.CartItem;

@Repository

public interface CartItemRepository extends CrudRepository<CartItem, Long> {

public List<CartItem> findCartItemByCartCartid(long cartid);

public CartItem findCartItemByProductProductid(long productid);

}

CartRepository.java

package com.ecommerce.repository;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.ecommerce.model.Cart;

@Repository

public interface CartRepository extends CrudRepository<Cart, Long> {

public Cart findByUserUserid(long userid);

}

OrderItemRepository.java

package com.ecommerce.repository;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.ecommerce.model.OrderItem;

@Repository

public interface OrderItemRepository extends CrudRepository<OrderItem, Long> {

}

OrdersRepository.java

package com.ecommerce.repository;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.ecommerce.model.Orders;

@Repository

public interface OrdersRepository extends CrudRepository<Orders, Long> {

}

ProductDetailRepository.java

package com.ecommerce.repository;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.ecommerce.model.ProductDetail;

@Repository

public interface ProductDetailRepository extends CrudRepository<ProductDetail, Long> {

public ProductDetail findByProductProductid(long productid);

}

ProductRepository.java

package com.ecommerce.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.ecommerce.model.Product;

@Repository

public interface ProductRepository extends CrudRepository<Product, Long> {

public List<Product> findByProductname(String name);

}

UserRepository.java

package com.ecommerce.repository;

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

import org.springframework.stereotype.Repository;

import com.ecommerce.model.User;

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

public User findByEmail(String email);

}