**Source Code**

1. **com.ecommerce.controller**
2. **AdminController.java**

**package** com.ecommerce.controller;

**import** java.math.BigDecimal;

**import** java.util.HashMap;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

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

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

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

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

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

**import** com.ecommerce.entity.\*;

**import** com.ecommerce.service.\*;

@Controller

**public** **class** AdminController {

@Autowired

**private** AdminService adminService;

@Autowired

**private** CategoryService categoryService;

@Autowired

**private** ProductService productService;

@Autowired

**private** PurchaseService purchaseService;

@Autowired

**private** PurchaseItemService purchaseItemService;

@Autowired

**private** UserService userService;

@RequestMapping(value = "/adminlogin", method = RequestMethod.***GET***)

**public** String login(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

map.addAttribute("pageTitle", "ADMIN LOGIN");

**return** "admin/login";

}

@RequestMapping(value = "/adminloginaction", method = RequestMethod.***POST***)

**public** String loginAction(ModelMap map, javax.servlet.http.HttpServletRequest request,

@RequestParam(value="admin\_id", required=**true**) String adminId,

@RequestParam(value="admin\_pwd", required=**true**) String adminPwd)

{

Admin admin = adminService.authenticate(adminId, adminPwd);

**if** (admin == **null**) {

map.addAttribute("error", "Admin login failed");

**return** "admin/login";

}

// store admin id in session

HttpSession session = request.getSession();

session.setAttribute("admin\_id", admin.getID());

**return** "admin/dashboard";

}

@RequestMapping(value = "/admindashboard", method = RequestMethod.***GET***)

**public** String dashboard(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

map.addAttribute("pageTitle", "ADMIN DASHBOARD");

**return** "admin/dashboard";

}

@RequestMapping(value = "/adminchangepassword", method = RequestMethod.***GET***)

**public** String changePwd(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

Admin admin = adminService.getAdminById((Long) session.getAttribute("admin\_id"));

map.addAttribute("admin", admin);

map.addAttribute("pageTitle", "ADMIN CHANGE PASSWORD");

**return** "admin/change-password";

}

@RequestMapping(value = "/adminchangepwdaction", method = RequestMethod.***POST***)

**public** String updatePassword(ModelMap map, @RequestParam(value="pwd", required=**true**) String pwd,

@RequestParam(value="pwd2", required=**true**) String pwd2,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

**if** (pwd == **null** || pwd2 == **null** || pwd.equals("") || pwd2.equals("")) {

map.addAttribute("error", "Error , Incomplete passwords submitted.");

**return** "admin/change-password";

}

**if** (!pwd.equals(pwd2)) {

map.addAttribute("error", "Error , Passwords do not match.");

**return** "admin/change-password";

}

Admin admin = adminService.getAdminById((Long) session.getAttribute("admin\_id"));

admin.setAdminPwd(pwd);

adminService.updatePwd(admin);

**return** "admin/dashboard";

}

@RequestMapping(value = "/adminproducts", method = RequestMethod.***GET***)

**public** String products(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

List<Product> list = productService.getAllProducts();

// use a MAP to link category names to each product in list

HashMap<Long, String> mapCats = **new** HashMap<Long, String>();

**for**(Product product: list) {

Category category = categoryService.getCategoryById(product.getCategoryId());

**if** (category != **null**)

mapCats.put(product.getID(), category.getName());

}

map.addAttribute("list", list);

map.addAttribute("mapCats", mapCats);

map.addAttribute("pageTitle", "ADMIN SETUP PRODUCTS");

**return** "admin/products";

}

@RequestMapping(value = "/admincategories", method = RequestMethod.***GET***)

**public** String categories(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

List<Category> list = categoryService.getAllCategories();

map.addAttribute("list", list);

map.addAttribute("pageTitle", "ADMIN SETUP PRODUCT CATEGORIES");

**return** "admin/categories";

}

@RequestMapping(value = "/adminmembers", method = RequestMethod.***GET***)

**public** String members(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

List<User> list = userService.getAllUsers();

map.addAttribute("list", list);

map.addAttribute("pageTitle", "ADMIN BROWSE MEMBERS");

**return** "admin/members";

}

@RequestMapping(value = "/adminpurchases", method = RequestMethod.***GET***)

**public** String purchases(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

List<Purchase> list = purchaseService.getAllItems();

BigDecimal total = **new** BigDecimal(0.0);

**for**(Purchase purchase: list) {

total = total.add(purchase.getTotal());

}

// use MAPs to mape users to each purchase and item names to each purchase item row

HashMap<Long, String> mapItems = **new** HashMap<Long, String>();

HashMap<Long, String> mapUsers = **new** HashMap<Long, String>();

**for**(Purchase purchase: list) {

total = total.add(purchase.getTotal());

User user = userService.getUserById(purchase.getUserId());

**if** (user != **null**)

mapUsers.put(purchase.getID(), user.getFname() + " " + user.getLname());

List<PurchaseItem> itemList = purchaseItemService.getAllItemsByPurchaseId(purchase.getID());

StringBuilder sb = **new** StringBuilder("");

**for**(PurchaseItem item: itemList) {

Product product = productService.getProductById(item.getProductId());

**if** (product != **null**)

sb.append(product.getName() + ", " +

item.getQty() + " units @" + item.getRate() + " = "

+ item.getPrice() + "<br>");

}

mapItems.put(purchase.getID(), sb.toString());

}

map.addAttribute("totalAmount", total);

map.addAttribute("list", list);

map.addAttribute("mapItems", mapItems);

map.addAttribute("mapUsers", mapUsers);

map.addAttribute("pageTitle", "ADMIN PURCHASES REPORT");

**return** "admin/purchases";

}

@RequestMapping(value = "/admindeletecat", method = RequestMethod.***GET***)

**public** String deleteCategory(ModelMap map, @RequestParam(value="id", required=**true**) String id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

**long** idValue = Long.*parseLong*(id);

Category category = **new** Category();

**if** (idValue > 0) {

categoryService.deleteCategory(idValue);

}

**return** "redirect:admincategories";

}

@RequestMapping(value = "/admineditcat", method = RequestMethod.***GET***)

**public** String editCategory(ModelMap map, @RequestParam(value="id", required=**true**) String id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

**long** idValue = Long.*parseLong*(id);

Category category = **new** Category();

**if** (idValue > 0) {

category = categoryService.getCategoryById(idValue);

} **else** {

category.setID(0);

}

map.addAttribute("category", category);

map.addAttribute("pageTitle", "ADMIN EDIT PRODUCT CATEGORY");

**return** "admin/edit-category";

}

@RequestMapping(value = "/admineditcataction", method = RequestMethod.***POST***)

**public** String updateCategory(ModelMap map, @RequestParam(value="id", required=**true**) String id,

@RequestParam(value="name", required=**true**) String name,

javax.servlet.http.HttpServletRequest request)

{

**long** idValue = Long.*parseLong*(id);

**if** (name == **null** || name.equals("") ) {

map.addAttribute("error", "Error, A category name must be specified");

**return** "redirect:admineditcat?id="+id;

}

Category category = **new** Category();

category.setID(idValue);

category.setName(name);

categoryService.updateCategory(category);

**return** "redirect:admincategories";

}

@RequestMapping(value = "/admindeleteproduct", method = RequestMethod.***GET***)

**public** String deleteProduct(ModelMap map, @RequestParam(value="id", required=**true**) String id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

**long** idValue = Long.*parseLong*(id);

Product product = **new** Product();

**if** (idValue > 0) {

productService.deleteProduct(idValue);

}

**return** "redirect:adminproducts";

}

@RequestMapping(value = "/admineditproduct", method = RequestMethod.***GET***)

**public** String editProduct(ModelMap map, @RequestParam(value="id", required=**true**) String id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if** (session.getAttribute("admin\_id") == **null**) {

**return** "admin/login";

}

**long** idValue = Long.*parseLong*(id);

Product product = **new** Product();

**if** (idValue > 0) {

product = productService.getProductById(idValue);

} **else** {

product.setID(0);

product.setCategoryId(0);

}

String dropDown = categoryService.getCategoriesDropDown(product.getCategoryId());

map.addAttribute("product", product);

map.addAttribute("catDropdown", dropDown);

map.addAttribute("pageTitle", "ADMIN EDIT PRODUCT");

**return** "admin/edit-product";

}

@RequestMapping(value = "/admineditproductaction", method = RequestMethod.***POST***)

**public** String updateProduct(ModelMap map, javax.servlet.http.HttpServletRequest request,

@RequestParam(value="id", required=**true**) String id,

@RequestParam(value="name", required=**true**) String name,

@RequestParam(value="price", required=**true**) String price,

@RequestParam(value="category", required=**true**) String categoryId)

{

**long** idValue = Long.*parseLong*(id);

**long** categoryIdValue = Long.*parseLong*(categoryId);

BigDecimal priceValue = **new** BigDecimal(0.0);

**if** (name == **null** || name.equals("") ) {

map.addAttribute("error", "Error, A product name must be specified");

**return** "redirect:admineditproduct?id="+id;

}

**try** {

priceValue = **new** BigDecimal(price);

} **catch** (Exception ex) {

map.addAttribute("error", "Error, Price is invalid");

**return** "redirect:admineditproduct?id="+id;

}

Product product = **new** Product();

product.setID(idValue);

product.setCategoryId(Long.*parseLong*(categoryId));

product.setName(name);

product.setPrice(priceValue);

productService.updateProduct(product);

**return** "redirect:adminproducts";

}

@RequestMapping(value = "/adminlogout", method = RequestMethod.***GET***)

**public** String logout(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

session.invalidate();

map.addAttribute("pageTitle", "ADMIN LOGIN");

**return** "admin/login";

}

}

1. **CartController.java**

**package** com.ecommerce.controller;

**import** java.math.BigDecimal;

**import** java.util.ArrayList;

**import** java.util.Calendar;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

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

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

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

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

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

**import** com.ecommerce.service.\*;

**import** com.ecommerce.entity.\*;

@Controller

**public** **class** CartController {

@Autowired

**private** ProductService productService;

@Autowired

**private** PurchaseService purchaseService;

@Autowired

**private** PurchaseItemService purchaseItemService;

@SuppressWarnings("unchecked")

@RequestMapping(value = "/cart", method = RequestMethod.***GET***)

**public** String cart(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before adding items to cart");

} **else** {

// if cart is already in session then retrieve it else create a new cart list and

// save it to session

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

// get total of all cart items

BigDecimal totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

map.addAttribute("pageTitle", "SPORTY SHOES - YOUR CART");

**return** "cart";

}

@SuppressWarnings("unchecked")

@RequestMapping(value = "/cartadditem", method = RequestMethod.***GET***)

**public** String cartAddItem(ModelMap map, javax.servlet.http.HttpServletRequest request,

@RequestParam(value="id", required=**true**) String productId)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before adding items to cart");

} **else** {

**long** idValue = Long.*parseLong*(productId);

// if cart is already in session then retrieve it else create a new cart list and

// save it to session

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

**if** (isItemInCart(cartItems, idValue)) {

map.addAttribute("error", "This item is already in your cart");

} **else** {

Product product = productService.getProductById(idValue);

CartItem item = **new** CartItem();

item.setProductId(idValue);

item.setQty(1);

item.setRate(product.getPrice());

BigDecimal dprice = item.getRate().multiply(**new** BigDecimal(item.getQty()));

item.setPrice(dprice);

item.setName(product.getName());

cartItems.add(item);

session.setAttribute("cart\_items", cartItems);

}

}

**return** "redirect:cart";

}

@RequestMapping(value = "/cartdeleteitem", method = RequestMethod.***GET***)

**public** String cartDeleteItem(ModelMap map, javax.servlet.http.HttpServletRequest request,

@RequestParam(value="id", required=**true**) String id)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before deleting items from cart");

} **else** {

**long** idValue = Long.*parseLong*(id);

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

**for**(CartItem item: cartItems) {

**if** (item.getProductId() == idValue) {

cartItems.remove(item);

session.setAttribute("cart\_items", cartItems);

**break**;

}

}

}

**return** "redirect:cart";

}

@RequestMapping(value = "/checkout", method = RequestMethod.***GET***)

**public** String checkout(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before checking out");

} **else** {

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

BigDecimal totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

map.addAttribute("pageTitle", "SPORTY SHOES - CHECKOUT");

**return** "checkout";

}

@RequestMapping(value = "/completepurchase", method = RequestMethod.***GET***)

**public** String completePurchase(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before completing purchase");

} **else** {

// take items from cart and update the databae

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

BigDecimal totalValue = getCartValue(cartItems);

**long** userId = (Long) session.getAttribute("user\_id") ;

Purchase purchase = **new** Purchase();

purchase.setUserId(userId);

purchase.setDate(Calendar.*getInstance*().getTime());

purchase.setTotal(totalValue);

**long** purchaseId = purchaseService.updatePurchase(purchase);

**for**(CartItem item: cartItems) {

PurchaseItem pItem = **new** PurchaseItem();

pItem.setPurchaseId(purchaseId);

pItem.setProductId(item.getProductId());

pItem.setUserId(userId);

pItem.setRate(item.getRate());

pItem.setQty(item.getQty());

pItem.setPrice(item.getPrice());

purchaseItemService.updateItem(pItem);

}

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

**return** "redirect:confirm";

}

@RequestMapping(value = "/gateway", method = RequestMethod.***GET***)

**public** String gateway(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before making payment");

} **else** {

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

BigDecimal totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

map.addAttribute("pageTitle", "SPORTY SHOES - PAYMENT GATEWAY");

**return** "gateway";

}

@RequestMapping(value = "/confirm", method = RequestMethod.***GET***)

**public** String confirm(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

map.addAttribute("error", "Error, You need to login before completing the purchase");

} **else** {

// clear items from cart as order has completed

List<CartItem> cartItems = **new** ArrayList<CartItem>();

**if** (session.getAttribute("cart\_items") != **null**)

cartItems = (List<CartItem>) session.getAttribute("cart\_items");

BigDecimal totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

cartItems.clear();

session.setAttribute("cart\_items", **null**);

}

map.addAttribute("pageTitle", "SPORTY SHOES - PURCHASE CONFIRMATION");

**return** "confirm";

}

/\*\*

\* Check if an item is already in the cart

\* **@param** list

\* **@param** item

\* **@return**

\*/

**private** **boolean** isItemInCart(List<CartItem> list, **long** item) {

**boolean** retVal = **false**;

**for**(CartItem thisItem: list) {

**if** (item == thisItem.getProductId()) {

retVal = **true**;

**break**;

}

}

**return** retVal;

}

/\*\*

\* Get total value of items in cart

\* **@param** list

\* **@return**

\*/

**private** BigDecimal getCartValue(List<CartItem> list) {

BigDecimal total = **new** BigDecimal(0.0);

**for**(CartItem item: list) {

BigDecimal dprice = item.getRate().multiply(**new** BigDecimal(item.getQty()));

total= total.add(dprice);

}

**return** total;

}

}

1. **DashboardController.java**

**package** com.ecommerce.controller;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

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

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

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

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

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

@Controller

**public** **class** DashboardController {

@RequestMapping(value = "/dashboard", method = RequestMethod.***GET***)

**public** String dashboard(ModelMap map)

{

map.addAttribute("pageTitle", "SPORTY SHOES - DASHBOARD");

**return** "dashboard";

}

}

1. **HomeController.java**

**package** com.ecommerce.controller;

**import** java.util.HashMap;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

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

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

**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.servlet.ModelAndView;

**import** com.ecommerce.entity.\*;

**import** com.ecommerce.service.\*;

@Controller

**public** **class** HomeController {

@Autowired

**private** CategoryService categoryService;

@Autowired

**private** ProductService productService;

**static** ModelAndView *modelViewObj*;

@RequestMapping("/greetingsMsg")

**public** ModelAndView getGreetingsMessage() {

System.***out***.println("Inside Greetings Message");

*modelViewObj* = **new** ModelAndView("hellopage","messageObj","Good Morning, Java Code Geek!");

**return** *modelViewObj*;

}

@RequestMapping(value = {"/", "/home"}, method = RequestMethod.***GET***)

**public** String home(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

List<Product> list = productService.getAllProducts();

// use MAP to map the category names to product rows

HashMap<Long, String> mapCats = **new** HashMap<Long, String>();

**for**(Product product: list) {

Category category = categoryService.getCategoryById(product.getCategoryId());

**if** (category != **null**)

mapCats.put(product.getID(), category.getName());

}

map.addAttribute("list", list);

map.addAttribute("mapCats", mapCats);

map.addAttribute("pageTitle", "SPORTY SHOES - HOMEPAGE");

**return** "index";

}

}

1. **MemberController.java**

**package** com.ecommerce.controller;

**import** javax.servlet.http.HttpSession;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

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

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

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

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

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

**import** com.ecommerce.entity.\*;

**import** com.ecommerce.service.\*;

@Controller

**public** **class** MemberController {

@Autowired

**private** UserService userService;

@RequestMapping(value = "/login", method = RequestMethod.***GET***)

**public** String login(ModelMap map)

{

map.addAttribute("pageTitle", "SPORTY SHOES - MEMBER LOGIN");

**return** "login";

}

@RequestMapping(value = "/loginaction", method = RequestMethod.***POST***)

**public** String loginAction(ModelMap map, javax.servlet.http.HttpServletRequest request,

@RequestParam(value="email\_id", required=**true**) String emailId,

@RequestParam(value="pwd", required=**true**) String pwd)

{

User user = userService.authenticate(emailId, pwd);

**if** (user == **null**) {

map.addAttribute("error", "Login failed");

**return** "login";

}

HttpSession session = request.getSession();

session.setAttribute("user\_id", user.getID());

**return** "redirect:dashboard";

}

@RequestMapping(value = "/signup", method = RequestMethod.***GET***)

**public** String signup(ModelMap map)

{

map.addAttribute("pageTitle", "SPORTY SHOES - MEMBER REGISTRATION");

**return** "register";

}

@RequestMapping(value = "/signupaction", method = RequestMethod.***POST***)

**public** String signupAction(ModelMap map, javax.servlet.http.HttpServletRequest request,

@RequestParam(value="email\_id", required=**true**) String emailId,

@RequestParam(value="pwd", required=**true**) String pwd,

@RequestParam(value="pwd2", required=**true**) String pwd2,

@RequestParam(value="fname", required=**true**) String fname,

@RequestParam(value="lname", required=**true**) String lname,

@RequestParam(value="age", required=**true**) String age,

@RequestParam(value="address", required=**true**) String address)

{

**if** (emailId == **null** || emailId.equals("")) {

map.addAttribute("error", "Email id is required.");

**return** "register";

}

**if** (pwd == **null** || pwd2 == **null** || pwd.equals("") || pwd2.equals("")) {

map.addAttribute("error", "Error , Incomplete passwords submitted.");

**return** "register";

}

**if** (!pwd.equals(pwd2)) {

map.addAttribute("error", "Error , Passwords do not match.");

**return** "register";

}

**if** (fname == **null** || fname.equals("")) {

map.addAttribute("error", "First name is required.");

**return** "register";

}

**if** (lname == **null** || lname.equals("")) {

map.addAttribute("error", "Last name is required.");

**return** "register";

}

**if** (age == **null** || age.equals("")) {

age = "0";

}

User user = userService.getUserByEmailId(emailId);

**if** (user != **null**) {

map.addAttribute("error", "This email id already exists.");

**return** "register";

}

user = **new** User();

user.setID(0);

user.setEmail(emailId);

user.setFname(fname);

user.setLname(lname);

user.setAge(Integer.*parseInt*(age));

user.setAddress(address);

user.setPwd(pwd);

userService.updateUser(user);

**return** "redirect:registerconfirm";

}

@RequestMapping(value = "/registerconfirm", method = RequestMethod.***GET***)

**public** String registerConfirm(ModelMap map)

{

**return** "register-confirm";

}

@RequestMapping(value = "/editprofile", method = RequestMethod.***GET***)

**public** String editProfile(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

**return** "login";

}

User user = userService.getUserById((Long) session.getAttribute("user\_id"));

map.addAttribute("pageTitle", "SPORTY SHOES - MEMBER EDIT PROFILE");

map.addAttribute("user", user);

**return** "edit-profile";

}

@RequestMapping(value = "/editprofileaction", method = RequestMethod.***POST***)

**public** String editProfileAction(ModelMap map,

javax.servlet.http.HttpServletRequest request,

@RequestParam(value="user\_id", required=**true**) String userId,

@RequestParam(value="pwd", required=**true**) String pwd,

@RequestParam(value="pwd2", required=**true**) String pwd2,

@RequestParam(value="fname", required=**true**) String fname,

@RequestParam(value="lname", required=**true**) String lname,

@RequestParam(value="age", required=**true**) String age,

@RequestParam(value="address", required=**true**) String address)

{

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

**return** "login";

}

User user = userService.getUserById((Long) session.getAttribute("user\_id"));

map.addAttribute("user", user);

**if** (pwd == **null** || pwd2 == **null** || pwd.equals("") || pwd2.equals("")) {

map.addAttribute("error", "Error , Incomplete passwords submitted.");

**return** "edit-profile";

}

**if** (!pwd.equals(pwd2)) {

map.addAttribute("error", "Error , Passwords do not match.");

**return** "edit-profile";

}

**if** (fname == **null** || fname.equals("")) {

map.addAttribute("error", "First name is required.");

**return** "edit-profile";

}

**if** (lname == **null** || lname.equals("")) {

map.addAttribute("error", "Last name is required.");

**return** "edit-profile";

}

**if** (age == **null** || age.equals("")) {

age = "0";

}

user.setFname(fname);

user.setLname(lname);

user.setAge(Integer.*parseInt*(age));

user.setAddress(address);

user.setPwd(pwd);

userService.updateUser(user);

**return** "redirect:dashboard";

}

@RequestMapping(value = "/logout", method = RequestMethod.***GET***)

**public** String logout(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

session.invalidate();

**return** "redirect:home";

}

}

1. **PurchaseController.java**

**package** com.ecommerce.controller;

**import** java.math.BigDecimal;

**import** java.util.HashMap;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

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

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

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

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

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

**import** com.ecommerce.service.\*;

**import** com.ecommerce.entity.\*;

@Controller

**public** **class** PurchaseController {

@Autowired

**private** ProductService productService;

@Autowired

**private** PurchaseService purchaseService;

@Autowired

**private** PurchaseItemService purchaseItemService;

@RequestMapping(value = "/memberpurchases", method = RequestMethod.***GET***)

**public** String memberpurchases(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

**if** (session.getAttribute("user\_id") == **null**) {

**return** "login";

}

**long** userId = (Long) session.getAttribute("user\_id");

List<Purchase> list = purchaseService.getAllItemsByUserId(userId);

BigDecimal total = **new** BigDecimal(0.0);

// map purchase items to each purchase for display

HashMap<Long, String> mapItems = **new** HashMap<Long, String>();

**for**(Purchase purchase: list) {

total = total.add(purchase.getTotal());

List<PurchaseItem> itemList = purchaseItemService.getAllItemsByPurchaseId(purchase.getID());

StringBuilder sb = **new** StringBuilder("");

**for**(PurchaseItem item: itemList) {

Product product = productService.getProductById(item.getProductId());

**if** (product != **null**) {

sb.append(product.getName() + ", " +

item.getQty() + " units @" + item.getRate() + " = "

+ item.getPrice() + "<br>");

}

}

mapItems.put(purchase.getID(), sb.toString());

}

map.addAttribute("totalAmount", total);

map.addAttribute("list", list);

map.addAttribute("mapItems", mapItems);

map.addAttribute("pageTitle", "SPORTY SHOES - YOUR ORDERS");

**return** "purchases";

}

}

1. **com.ecommerce.dao**
2. **AdminDao.java**

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Admin;

@Repository

@Component

**public** **class** AdminDAO {

@Autowired

**private** SessionFactory sessionFactory;

@SuppressWarnings("unchecked")

**public** Admin authenticate(String adminId, String pwd) {

List<Admin> list = **this**.sessionFactory.getCurrentSession().createQuery("from Admin where admin\_id=:admin\_id and admin\_pwd=:admin\_pwd")

.setParameter("admin\_id", adminId)

.setParameter("admin\_pwd", pwd)

.list();

**if** (list.size() > 0)

**return** (Admin) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** Admin getAdminById(**long** id) {

List<Admin> list = **this**.sessionFactory.getCurrentSession().createQuery("from Admin where ID=:admin\_id")

.setParameter("admin\_id", id)

.list();

**if** (list.size() > 0)

**return** (Admin) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** **void** updatePwd(Admin admin) {

String sql = "update Admin set admin\_pwd=:admin\_pwd where ID=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("admin\_pwd", admin.getAdminPwd());

query.~~setParameter~~("id", admin.getID());

query.executeUpdate();

}

}

1. **CategoryDao.java**

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.Session;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Category;

@Repository

@Component

**public** **class** CategoryDAO {

@Autowired

**private** SessionFactory sessionFactory;

@SuppressWarnings("unchecked")

**public** Category getCategoryById(**long** id) {

String strId = String.*valueOf*(id);

List<Category> list = **this**.sessionFactory.getCurrentSession().createQuery("from Category where id=" + strId).list();

**if** (list.size() > 0)

**return** (Category) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** **void** updateCategory(Category category) {

String sql = "";

**if** (category.getID() == 0)

**this**.sessionFactory.getCurrentSession().save(category);

**else** **if** (category.getID() > 0) {

sql = "update Category set name=:name where " +

" ID=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("name", category.getName());

**if** (category.getID() > 0)

query.~~setParameter~~("id", category.getID());

query.executeUpdate();

}

}

@SuppressWarnings("unchecked")

**public** **void** deleteCategory(**long** id) {

// mark all category\_id of products with this category to zero before deleting the category row

String sql = "";

sql = "update Product set category\_id=0 where category\_id=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("id", id);

query.executeUpdate();

sql = "delete from Category where ID=:id";

query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("id", id);

query.executeUpdate();

}

@SuppressWarnings("unchecked")

**public** List<Category> getAllCategories() {

List<Category> list = **this**.sessionFactory.getCurrentSession().createQuery("from Category order by name").list();

**return** list;

}

}

1. **ProductDao.java**

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Product;

@Repository

@Component

**public** **class** ProductDAO {

@Autowired

**private** SessionFactory sessionFactory;

@SuppressWarnings("unchecked")

**public** Product getProductById(**long** id) {

String strId = String.*valueOf*(id);

List<Product> list = **this**.sessionFactory.getCurrentSession().createQuery("from Product where id=" + strId).list();

**if** (list.size() > 0)

**return** (Product) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** **void** updateProduct(Product product) {

String sql = "";

**if** (product.getID() == 0)

**this**.sessionFactory.getCurrentSession().save(product);

**else** **if** (product.getID() > 0) {

sql = "update Product set name=:name, price=:price, category\_id=:category\_id where " +

" ID=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("name", product.getName());

query.~~setParameter~~("price", product.getPrice());

query.~~setParameter~~("category\_id", product.getCategoryId());

query.~~setParameter~~("id", product.getID());

query.executeUpdate();

}

}

@SuppressWarnings("unchecked")

**public** **void** deleteProduct(**long** id) {

// delete all purchase items for this product before deleting this product

// Pending:Purchase total is not updated in the purchase total - it shows the old value

String sql = "";

sql = "delete from PurchaseItem where product\_id=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("id", id);

sql = "delete from Product where ID=:id";

query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("id", id);

query.executeUpdate();

}

@SuppressWarnings("unchecked")

**public** List<Product> getAllProducts() {

List<Product> list = **this**.sessionFactory.getCurrentSession().createQuery("from Product order by name").list();

**return** list;

}

@SuppressWarnings("unchecked")

**public** List<Product> getAllProductsWithJoin() {

List<Product> list = **this**.sessionFactory.getCurrentSession().createQuery("from Product order by name").list();

**return** list;

}

}

1. **PurchaseDao.java**

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Purchase;

@Repository

@Component

**public** **class** PurchaseDAO {

@Autowired

**private** SessionFactory sessionFactory;

@SuppressWarnings("unchecked")

**public** Purchase getPurchaseById(**long** id) {

String strId = String.*valueOf*(id);

List<Purchase> list = **this**.sessionFactory.getCurrentSession().createQuery("from Purchase where id=" + strId).list();

**if** (list.size() > 0)

**return** (Purchase) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** List<Purchase> getAllItems() {

List<Purchase> list = **this**.sessionFactory.getCurrentSession().createQuery("from Purchase order by ID desc").list();

**return** list;

}

@SuppressWarnings("unchecked")

**public** List<Purchase> getAllItemsByUserId(**long** userId) {

String strId = String.*valueOf*(userId);

List<Purchase> list = **this**.sessionFactory.getCurrentSession().createQuery("from Purchase where user\_id=" + strId + " order by ID desc").list();

**return** list;

}

@SuppressWarnings("unchecked")

**public** **long** updatePurchase(Purchase purchase) {

String sql = "";

**long** newId = 0;

**if** (purchase.getID() == 0) {

**this**.sessionFactory.getCurrentSession().save(purchase);

newId = purchase.getID();

} **else** {

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("user\_id", purchase.getUserId());

query.~~setParameter~~("gross\_total", purchase.getTotal());

query.executeUpdate();

}

**return** newId;

}

}

1. **PurchaseItemDao.java**

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.PurchaseItem;

@Repository

@Component

**public** **class** PurchaseItemDAO {

@Autowired

**private** SessionFactory sessionFactory;

@SuppressWarnings("unchecked")

**public** PurchaseItem getItemById(**long** id) {

String strId = String.*valueOf*(id);

List<PurchaseItem> list = **this**.sessionFactory.getCurrentSession().createQuery("from PurchaseItem where id=" + strId).list();

**if** (list.size() > 0)

**return** (PurchaseItem) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** List<PurchaseItem> getAllItemsByPurchaseId(**long** purchaseId) {

String strId = String.*valueOf*(purchaseId);

List<PurchaseItem> list = **this**.sessionFactory.getCurrentSession().createQuery("from PurchaseItem where purchase\_id=" + strId).list();

**return** list;

}

@SuppressWarnings("unchecked")

**public** **void** updateItem(PurchaseItem item) {

**this**.sessionFactory.getCurrentSession().save(item);

}

@SuppressWarnings("unchecked")

**public** **void** deleteItem(**long** id) {

//Pending: Purchase total in purchase table should be updated after this

String sql = "";

sql += "delete from PurchaseItem where ID=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("id", id);

query.executeUpdate();

}

@SuppressWarnings("unchecked")

**public** **void** deleteAllItemsForPurchaseId(**long** purchaseId) {

String sql = "";

sql += "delete from PurchaseItem where purchase\_id=:purchase\_id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("purchase\_id", purchaseId);

query.executeUpdate();

}

}

1. **UserDao.java**

**package** com.ecommerce.dao;

**import** java.util.Calendar;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Admin;

**import** com.ecommerce.entity.User;

@Repository

@Component

**public** **class** UserDAO {

@Autowired

**private** SessionFactory sessionFactory;

@SuppressWarnings("unchecked")

**public** User authenticate(String emailId, String pwd) {

List<User> list = **this**.sessionFactory.getCurrentSession().createQuery("from User where emailid=:emailid and pwd=:pwd")

.setParameter("emailid", emailId)

.setParameter("pwd", pwd)

.list();

**if** (list.size() > 0)

**return** (User) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** User getUserById(**long** id) {

String strId = String.*valueOf*(id);

List<User> list = **this**.sessionFactory.getCurrentSession().createQuery("from User where id=" + strId).list();

**if** (list.size() > 0)

**return** (User) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** User getUserByEmailId(String emailId) {

List<User> list = **this**.sessionFactory.getCurrentSession().createQuery("from User where emailid='" + emailId + "'").list();

**if** (list.size() > 0)

**return** (User) list.get(0);

**else**

**return** **null**;

}

@SuppressWarnings("unchecked")

**public** **void** updateUser(User user) {

String sql = "";

**if** (user.getID() == 0) {

user.setDateAdded(Calendar.*getInstance*().getTime());

**this**.sessionFactory.getCurrentSession().save(user);

} **else** **if** (user.getID() > 0) {

sql = "update User set fname=:fname, lname=:lname, address=:address, age=:age, pwd=:pwd where " +

" ID=:id";

~~Query~~ query = **this**.sessionFactory.getCurrentSession().createQuery(sql);

query.~~setParameter~~("fname", user.getFname());

query.~~setParameter~~("lname", user.getLname());

query.~~setParameter~~("address", user.getAddress());

query.~~setParameter~~("age", user.getAge());

query.~~setParameter~~("pwd", user.getPwd());

**if** (user.getID() > 0)

query.~~setParameter~~("id", user.getID());

query.executeUpdate();

}

}

@SuppressWarnings("unchecked")

**public** List<User> getAllUsers() {

List<User> list = **this**.sessionFactory.getCurrentSession().createQuery("from User order by date\_added").list();

**return** list;

}

}

1. **com.ecommerce.entity**
2. **Admin.java**

**package** com.ecommerce.entity;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name= "admin")

**public** **class** Admin {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "ID")

**private** **long** ID;

@Column(name = "admin\_id")

**private** String adminId;

@Column(name = "admin\_pwd")

**private** String pwd;

**public** **long** getID() {**return** **this**.ID; }

**public** String getAdminId() { **return** **this**.adminId;}

**public** String getAdminPwd() { **return** **this**.pwd;}

**public** **void** setID(**long** id) { **this**.ID = id;}

**public** **void** setAdminId(String value) { **this**.adminId= value;}

**public** **void** setAdminPwd(String value) { **this**.pwd = value;}

}

1. **CartItem.java**

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

/\*\*

\* This is NOT a Hibernate class or a table class. This is a POJO which is used internally within the app

\* **@author** oem

\*

\*/

**public** **class** CartItem {

**private** **long** productId;

**private** String name;

**private** BigDecimal rate;

**private** BigDecimal price;

**private** **int** qty;

**public** **long** getProductId() {**return** **this**.productId; }

**public** String getName() { **return** **this**.name;}

**public** BigDecimal getPrice() { **return** **this**.price;}

**public** BigDecimal getRate() { **return** **this**.rate;}

**public** **int** getQty() { **return** **this**.qty;}

**public** **void** setProductId(**long** id) { **this**.productId= id;}

**public** **void** setName(String value) { **this**.name = value;}

**public** **void** setPrice(BigDecimal value) { **this**.price = value;}

**public** **void** setRate(BigDecimal value) { **this**.rate = value;}

**public** **void** setQty(**int** value) { **this**.qty = value;}

}

1. **Category.java**

**package** com.ecommerce.entity;

**import** java.util.Date;

**import** java.util.List;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

@Entity

@Table(name= "category")

**public** **class** Category {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "ID")

**private** **long** ID;

@Column(name = "name")

**private** String name;

**public** **long** getID() {**return** **this**.ID; }

**public** String getName() { **return** **this**.name;}

**public** **void** setID(**long** id) { **this**.ID = id;}

**public** **void** setName(String value) { **this**.name = value;}

}

1. **Product.java**

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.OneToOne;

**import** javax.persistence.Table;

@Entity

@Table(name= "eproduct")

**public** **class** Product {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "ID")

**private** **long** ID;

@Column(name = "name")

**private** String name;

@Column(name = "price")

**private** BigDecimal price;

@Column(name = "date\_added")

**private** Date dateAdded;

@Column(name = "category\_id")

**private** **long** categoryId;

**public** **long** getID() {**return** **this**.ID; }

**public** String getName() { **return** **this**.name;}

**public** BigDecimal getPrice() { **return** **this**.price;}

**public** **long** getCategoryId() { **return** **this**.categoryId;}

**public** Date getDateAdded() { **return** **this**.dateAdded;}

**public** **void** setID(**long** id) { **this**.ID = id;}

**public** **void** setName(String value) { **this**.name = value;}

**public** **void** setPrice(BigDecimal value) { **this**.price = value;}

**public** **void** setCategoryId(**long** value) { **this**.categoryId = value;}

**public** **void** setDateAdded(Date date) { **this**.dateAdded = date;}

}

1. **Purchase.java**

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name= "purchases")

**public** **class** Purchase {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "ID")

**private** **long** ID;

@Column(name = "user\_id")

**private** **long** userId;

@Column(name = "date")

**private** Date date;

@Column(name = "gross\_total")

**private** BigDecimal total;

**public** **long** getID() {**return** **this**.ID; }

**public** **long** getUserId() { **return** **this**.userId;}

**public** BigDecimal getTotal() { **return** **this**.total;}

**public** Date getDate() { **return** **this**.date;}

**public** **void** setID(**long** id) { **this**.ID = id;}

**public** **void** setUserId(**long** value) { **this**.userId = value;}

**public** **void** setTotal(BigDecimal value) { **this**.total = value;}

**public** **void** setDate(Date date) { **this**.date = date;}

}

1. **PurchaseItem.java**

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name= "purchase\_items")

**public** **class** PurchaseItem {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "ID")

**private** **long** ID;

@Column(name = "purchase\_id")

**private** **long** purchaseId;

@Column(name = "product\_id")

**private** **long** productId;

@Column(name = "user\_id")

**private** **long** userId;

@Column(name = "rate")

**private** BigDecimal rate;

@Column(name = "qty")

**private** **int** qty;

@Column(name = "price")

**private** BigDecimal price;

**public** **long** getID() {**return** **this**.ID; }

**public** **long** getPurchaseId() { **return** **this**.purchaseId;}

**public** **long** getProductId() { **return** **this**.productId;}

**public** **long** getUserId() { **return** **this**.userId;}

**public** BigDecimal getRate() { **return** **this**.rate;}

**public** **int** getQty() { **return** **this**.qty;}

**public** BigDecimal getPrice() { **return** **this**.price;}

**public** **void** setID(**long** id) { **this**.ID = id;}

**public** **void** setPurchaseId(**long** value) { **this**.purchaseId = value;}

**public** **void** setProductId(**long** value) { **this**.productId = value;}

**public** **void** setUserId(**long** value) { **this**.userId = value;}

**public** **void** setRate(BigDecimal value) { **this**.rate = value;}

**public** **void** setQty(**int** value) { **this**.qty= value;}

**public** **void** setPrice(BigDecimal value) { **this**.price= value;}

}

1. **User.java**

**package** com.ecommerce.entity;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name= "users")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "ID")

**private** **long** ID;

@Column(name = "fname")

**private** String fname;

@Column(name = "lname")

**private** String lname;

@Column(name = "address")

**private** String address;

@Column(name = "age")

**private** **int** age;

@Column(name = "date\_added")

**private** Date dateAdded;

@Column(name = "emailid")

**private** String emailId;

@Column(name = "pwd")

**private** String pwd;

**public** **long** getID() {**return** **this**.ID; }

**public** String getEmail() { **return** **this**.emailId;}

**public** String getFname() { **return** **this**.fname;}

**public** String getLname() { **return** **this**.lname;}

**public** String getAddress() { **return** **this**.address;}

**public** String getPwd() { **return** **this**.pwd;}

**public** **int** getAge() { **return** **this**.age;}

**public** Date getDateAdded() { **return** **this**.dateAdded;}

**public** **void** setID(**long** id) { **this**.ID = id;}

**public** **void** setEmail(String value) { **this**.emailId = value;}

**public** **void** setFname(String value) { **this**.fname = value;}

**public** **void** setLname(String value) { **this**.lname = value;}

**public** **void** setAddress(String value) { **this**.address= value;}

**public** **void** setPwd(String value) { **this**.pwd= value;}

**public** **void** setAge(**int** value) { **this**.age= value;}

**public** **void** setDateAdded(Date date) { **this**.dateAdded = date;}

}

1. **com.ecommercer.service**
2. **AdminService.java**

**package** com.ecommerce.service;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.AdminDAO;

**import** com.ecommerce.entity.Admin;

@Component

**public** **class** AdminService {

@Autowired

**private** AdminDAO adminDAO;

@Transactional

**public** Admin authenticate(String adminId, String pwd) {

**return** adminDAO.authenticate(adminId, pwd);

}

@Transactional

**public** Admin getAdminById(**long** id) {

**return** adminDAO.getAdminById(id);

}

@Transactional

**public** **void** updatePwd(Admin admin) {

adminDAO.updatePwd(admin);

}

}

1. **CategoryService.java**

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.CategoryDAO;

**import** com.ecommerce.entity.Category;

@Component

**public** **class** CategoryService {

@Autowired

**private** CategoryDAO categoryDAO;

@Transactional

**public** Category getCategoryById(**long** id) {

**return** categoryDAO.getCategoryById(id);

}

@Transactional

**public** **void** updateCategory(Category category) {

categoryDAO.updateCategory(category);

}

@Transactional

**public** **void** deleteCategory(**long** id) {

categoryDAO.deleteCategory(id);

}

@Transactional

**public** List<Category> getAllCategories() {

**return** categoryDAO.getAllCategories();

}

@Transactional

**public** String getCategoriesDropDown(**long** id) {

StringBuilder sb = **new** StringBuilder("");

List<Category> list = categoryDAO.getAllCategories();

**for**(Category cat: list) {

**if** (cat.getID() == id)

sb.append("<option value=" + String.*valueOf*(cat.getID()) + " selected>" + cat.getName() + "</option>");

**else**

sb.append("<option value=" + String.*valueOf*(cat.getID()) + ">" + cat.getName() + "</option>");

}

**return** sb.toString();

}

}

1. **ProductService.java**

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.ProductDAO;

**import** com.ecommerce.entity.Product;

@Component

**public** **class** ProductService {

@Autowired

**private** ProductDAO productDAO;

@Transactional

**public** Product getProductById(**long** id) {

**return** productDAO.getProductById(id);

}

@Transactional

**public** **void** updateProduct(Product product) {

productDAO.updateProduct(product);

}

@Transactional

**public** **void** deleteProduct(**long** id) {

productDAO.deleteProduct(id);

}

@Transactional

**public** List<Product> getAllProducts() {

**return** productDAO.getAllProducts();

}

@Transactional

**public** List<Product> getAllProductsWithJoin() {

**return** productDAO.getAllProductsWithJoin();

}

}

1. **PurchaseItem.java**

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.PurchaseItemDAO;

**import** com.ecommerce.entity.PurchaseItem;

@Component

**public** **class** PurchaseItemService {

@Autowired

**private** PurchaseItemDAO purchaseItemDAO;

@Transactional

**public** PurchaseItem getItemById(**long** id) {

**return** purchaseItemDAO.getItemById(id);

}

@Transactional

**public** List<PurchaseItem> getAllItemsByPurchaseId(**long** purchaseId) {

**return** purchaseItemDAO.getAllItemsByPurchaseId(purchaseId);

}

@Transactional

**public** **void** updateItem(PurchaseItem item) {

purchaseItemDAO.updateItem(item);

}

@Transactional

**public** **void** deleteItem(**long** id) {

purchaseItemDAO.deleteAllItemsForPurchaseId(id);

}

@Transactional

**public** **void** deleteAllItemsForPurchaseId(**long** purchaseId) {

purchaseItemDAO.deleteAllItemsForPurchaseId(purchaseId);

}

}

1. **PurchaseService.java**

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.PurchaseDAO;

**import** com.ecommerce.entity.Purchase;

@Component

**public** **class** PurchaseService {

@Autowired

**private** PurchaseDAO purchaseDAO;

@Transactional

**public** Purchase getPurchaseById(**long** id) {

**return** purchaseDAO.getPurchaseById(id);

}

@Transactional

**public** List<Purchase> getAllItems() {

**return** purchaseDAO.getAllItems();

}

@Transactional

**public** List<Purchase> getAllItemsByUserId(**long** userId) {

**return** purchaseDAO.getAllItemsByUserId(userId);

}

@Transactional

**public** **long** updatePurchase(Purchase purchase) {

**return** purchaseDAO.updatePurchase(purchase);

}

//@Transactional

//public void deletePurchase(long id) {

// purchaseDAO.deletePurchase(id);

//}

}

1. **UserService.java**

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.~~Query~~;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.UserDAO;

**import** com.ecommerce.entity.Admin;

**import** com.ecommerce.entity.User;

@Component

**public** **class** UserService {

@Autowired

**private** UserDAO userDAO;

@Transactional

**public** User authenticate(String userId, String pwd) {

**return** userDAO.authenticate(userId, pwd);

}

@Transactional

**public** User getUserById(**long** id) {

**return** userDAO.getUserById(id);

}

@Transactional

**public** User getUserByEmailId(String emailId) {

**return** userDAO.getUserByEmailId(emailId);

}

@Transactional

**public** **void** updateUser(User user) {

userDAO.updateUser(user);

}

@Transactional

**public** List<User> getAllUsers() {

**return** userDAO.getAllUsers();

}

}

End