Class Path:-

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

<classpath>

<classpathentry kind="src" output="target/classes" path="src/main/java">

<attributes>

<attribute name="optional" value="true"/>

<attribute name="maven.pomderived" value="true"/>

</attributes>

</classpathentry>

<classpathentry excluding="\*\*" kind="src" output="target/classes" path="src/main/resources">

<attributes>

<attribute name="maven.pomderived" value="true"/>

</attributes>

</classpathentry>

<classpathentry kind="src" output="target/test-classes" path="src/test/java">

<attributes>

<attribute name="test" value="true"/>

<attribute name="optional" value="true"/>

<attribute name="maven.pomderived" value="true"/>

</attributes>

</classpathentry>

<classpathentry exported="true" kind="con" path="org.eclipse.jdt.launching.JRE\_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/J2SE-1.5">

<attributes>

<attribute name="module" value="true"/>

<attribute name="maven.pomderived" value="true"/>

</attributes>

</classpathentry>

<classpathentry exported="true" kind="con" path="org.eclipse.m2e.MAVEN2\_CLASSPATH\_CONTAINER">

<attributes>

<attribute name="maven.pomderived" value="true"/>

<attribute name="org.eclipse.jst.component.dependency" value="/WEB-INF/lib"/>

</attributes>

</classpathentry>

<classpathentry kind="output" path="target/classes"/>

</classpath>

Project:-

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

<projectDescription>

<name>SportyShoes</name>

<comment></comment>

<projects>

</projects>

<buildSpec>

<buildCommand>

<name>org.eclipse.jdt.core.javabuilder</name>

<arguments>

</arguments>

</buildCommand>

<buildCommand>

<name>org.eclipse.wst.common.project.facet.core.builder</name>

<arguments>

</arguments>

</buildCommand>

<buildCommand>

<name>org.eclipse.m2e.core.maven2Builder</name>

<arguments>

</arguments>

</buildCommand>

<buildCommand>

<name>org.eclipse.wst.validation.validationbuilder</name>

<arguments>

</arguments>

</buildCommand>

</buildSpec>

<natures>

<nature>org.eclipse.jem.workbench.JavaEMFNature</nature>

<nature>org.eclipse.wst.common.modulecore.ModuleCoreNature</nature>

<nature>org.eclipse.jdt.core.javanature</nature>

<nature>org.eclipse.m2e.core.maven2Nature</nature>

<nature>org.eclipse.wst.common.project.facet.core.nature</nature>

<nature>org.eclipse.wst.jsdt.core.jsNature</nature>

</natures>

</projectDescription>

Pom:-

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>SportyShoes</groupId>

<artifactId>SportyShoes</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>Section7LEP Maven Webapp</name>

<url>http://maven.apache.org</url>

<!-- JBoss repository for Hibernate -->

<repositories>

<repository>

<id>JBoss repository</id>

<url>http://repository.jboss.org/nexus/content/groups/public/</url>

</repository>

</repositories>

<properties>

<org.springframework.version>5.1.6.RELEASE</org.springframework.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.1-api</artifactId>

<version>1.0.0.Final</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.0.5.Final</version>

</dependency>

<dependency>

<groupId>javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.12.1.GA</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.9</version>

</dependency>

<dependency>

<groupId>javax.transaction</groupId>

<artifactId>jta</artifactId>

<version>1.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>javax.activation</groupId>

<artifactId>activation</artifactId>

<version>1.1.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<build>

<finalName>Section7LEP2</finalName>

<sourceDirectory>src/main</sourceDirectory>

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<configuration>

<webXml>src\main\webapp\WEB-INF\web.xml</webXml>

</configuration>

<version>2.4</version>

<executions>

<execution>

<id>default-war</id>

<phase>package</phase>

<goals>

<goal>war</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

AdminController:-

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();

return "admin/login";

}

}

CartController:-

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;

}

}

DashBoardController:-

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";

}

}

HomeController:-

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 com.ecommerce.entity.\*;

import com.ecommerce.service.\*;

@Controller

public class HomeController {

@Autowired

private CategoryService categoryService;

@Autowired

private ProductService productService;

@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";

}

}

MemberController:-

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";

}

}