

SHOEMAZON

Summary:

With the increasing appeal of online shopping, SHOEMAZON delivers the best assortment of footwear through an e-commerce web application, where user can choose a pair of shoes from variety of options. Plenty of shoe options will be available to the user and user via which user can select the desired product and the put those into cart. User can also delete items from the cart as well. After confirming the desired product, user can then proceed with checkout. User needs to put the basic information about himself before the order gets confirm, which he can update it later as well. User can finalize the order and receive the email about the order number of recently confirmed order. Other role system have is of admin who can add new products and view all placed orders.

Summary Of functionality

User will register to system by email confirmation which he will receive after entering desired details. User will activate the account by clicking on the link which he will receive in the email.

After Registration, User will log in to the application. Where he can select products from exclusive range of footwear. Use can put an item into cart and proceed to checkout. User can continue to shopping or proceed to checkout.

If he clicks on the proceed to checkout, he can update the customer details or edit the shopping cart as well. There will be an send button to finalize the order, if user the finalize the order he can send the email notification about the confirmed order number.

The other role in the system is MANAGER who can add the new products. Manager can view all the orders that are placed into the application.

Summary of Technologies Used:

Web application is developed using spring mvc. Maven is used for managing the repository. Hibernate is used to map an object-oriented domain model to a relational database tables. HQL is used for performing the queries. To restrict result of HQL criteria's are used. Ajax is used to refresh part of page with the help of XMLHttpRequest. BotDetech captcha is used for captcha image while registration. Itextpdf dependency is used to download and view pdf report of the order. For email notification, Common.mail extension is used.

Roles Of System

- **EMPLOYEE**
Employee is the user of the system, who will register himself the login and place and view the order in the cart. Receive the confirmation of the placed order number.
- **MANAGER**
Manager can add new footwear, view all the orders placed and view specific order with the total. He can download the pdf of the order details.

ScreenShots

Register

Shoemazon


Hello [nishad](#) | [Logout](#) [Register a new Customer](#)

[Home](#) | [Product List](#) | [My Cart](#)

User Email:

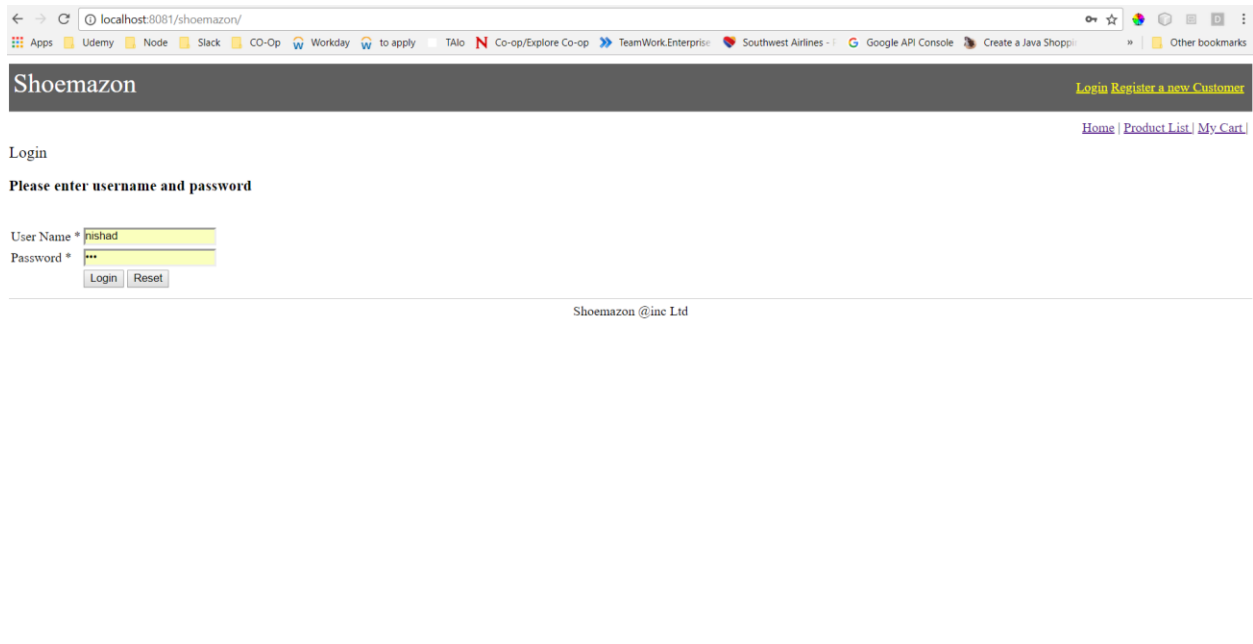
Password:

Retype the characters from the picture:



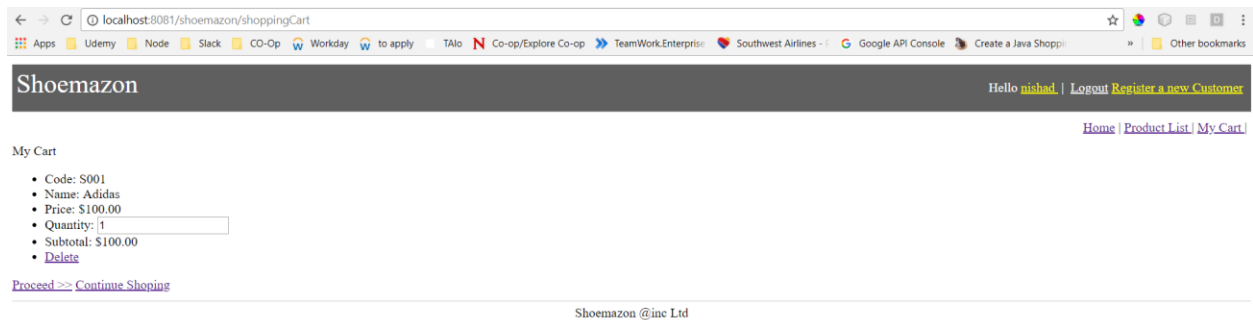
Shoemazon @inc Ltd

Login



The screenshot shows a web browser window with the address bar displaying 'localhost:8081/shoemazon/'. The page has a dark header with the 'Shoemazon' logo on the left and a link to 'Login Register a new Customer' on the right. Below the header, there are navigation links: 'Home', 'Product List', and 'My Cart'. The main content area is titled 'Login' and contains the instruction 'Please enter username and password'. There are two input fields: 'User Name *' with the value 'nishad' and 'Password *' with masked characters '***'. Below these fields are 'Login' and 'Reset' buttons. At the bottom of the page, there is a footer that reads 'Shoemazon @inc Ltd'.

ShoppingCart



The screenshot shows a web browser window with the address bar displaying 'localhost:8081/shoemazon/shoppingCart'. The page has a dark header with the 'Shoemazon' logo on the left and a greeting 'Hello nishad,' followed by links to 'Logout' and 'Register a new Customer' on the right. Below the header, there are navigation links: 'Home', 'Product List', and 'My Cart'. The main content area is titled 'My Cart' and contains a list of items in the cart. The items are: 'Code: S001', 'Name: Adidas', 'Price: \$100.00', 'Quantity: 1' (with an input field), 'Subtotal: \$100.00', and a 'Delete' link. Below the list, there are links to 'Proceed >>' and 'Continue Shopping'. At the bottom of the page, there is a footer that reads 'Shoemazon @inc Ltd'.

shoppingcartCustomer

localhost:8081/shoemazon/shoppingCartCustomer

Apps Udemy Node Slack CO-Op Workday to apply Talo Co-op/Explore Co-op TeamWork.Enterprise Southwest Airlines Google API Console Create a Java Shoppi Other bookmarks

Shoemazon

Hello [nishad](#) | [Logout](#) [Register a new Customer](#)

[Home](#) | [Product List](#) | [My Cart](#)

Enter Customer Information

Name *

Email *

Phone *

Address *

Shoemazon @inc Ltd

Confirmation

localhost:8081/shoemazon/Confirmation

Apps Udemy Node Slack CO-Op Workday to apply Talo Co-op/Explore Co-op TeamWork.Enterprise Southwest Airlines Google API Console Create a Java Shoppi Other bookmarks

Shoemazon

Hello [nishad](#) | [Logout](#) [Register a new Customer](#)

[Home](#) | [Product List](#) | [My Cart](#)

Confirmation

Customer Information:

- Name: Nishad
- Email: ranadive.n@husky.neu.edu
- Phone: 8578005935
- Address: Boston MA

Cart Summary:

- Quantity: 1
- Total: \$100.00

[Edit Cart](#) [Edit Customer Info](#)

- Name: Adidas
- Price: \$100.00
- Quantity: 1
- Subtotal: \$100.00

Shoemazon @inc Ltd

Order Finalize

localhost:8081/shoemazon/shoppingConfirmation

AppsUdemyNodeSlackCO-OpWorkdayto applyTaloCo-op/Explore Co-opTeamWork.EnterpriseSouthwest AirlinesGoogle API ConsoleCreate a Java ShoppiOther bookmarks

Shoemazon

Hello nishad | LogoutRegister a new Customer

Home | Product List | My Cart |

Finalize

Thank you for Order

Your order number is: 9

Send Email

Email Send

Shoemazon @inc Ltd

Manager Order List

localhost:8081/shoemazon/orderList

AppsUdemyNodeSlackCO-OpWorkdayto applyTaloCo-op/Explore Co-opTeamWork.EnterpriseSouthwest AirlinesGoogle API ConsoleCreate a Java ShoppiOther bookmarks

Shoemazon

Hello manager1 | LogoutRegister a new Customer

Home | Product List | My Cart | Order List | Create Product

Order List

Total Order Count: 9

Order Num	Order Date	Customer Name	Customer Address	Customer Email	Amount	View
9	27-04-2018 15:52	Nishad	Boston MA	ranadive.n@husky.neu.edu	\$100.00	View
8	27-04-2018 11:54	Nishad	boston MA	nishadranadive@yahoo.com	\$120.00	View
7	27-04-2018 11:38	Nishad	31 Saint Germain Street, Apartment 6	ranadive.n@husky.neu.edu	\$120.00	View
6	27-04-2018 11:27	kinnari	boston	kinnarisangvi2016@gmail.com	\$100.00	View
5	27-04-2018 01:48	Nishad	31 Saint Germain Street, Apartment 6	ranadive.n@husky.neu.edu	\$100.00	View
4	27-04-2018 01:47	Nishad	31 Saint Germain Street, Apartment 6	ranadive.n@husky.neu.edu	\$100.00	View
3	27-04-2018 01:33	Nishad	31 Saint Germain Street, Apartment 6	ranadive.n@husky.neu.edu	\$100.00	View
2	27-04-2018 01:17	Nishad	31 Saint Germain Street, Apartment 6	ranadive.n@husky.neu.edu	\$100.00	View
1	27-04-2018 01:14	Nishad	31 Saint Germain Street, Apartment 6	ranadive.n@husky.neu.edu	\$100.00	View

Shoemazon @inc Ltd

Particular order details

Shoemazon

Hello [manager1](#) | [Logout](#) [Register a new Customer](#)

[Home](#) | [Product List](#) | [My Cart](#) | [Order List](#) | [Create Product](#)

Order Info

Customer Information:

- Name: Nishad
- Email: ranadive.n@husky.neu.edu
- Phone: 8578005935
- Address: Boston MA

Order Summary:

- Total: \$100.00

[Download PDF](#)

Product Code	Product Name	Quantity	Price	Amount
S001	Adidas	1	\$100.00	\$100.00

Shoemazon @inc Ltd

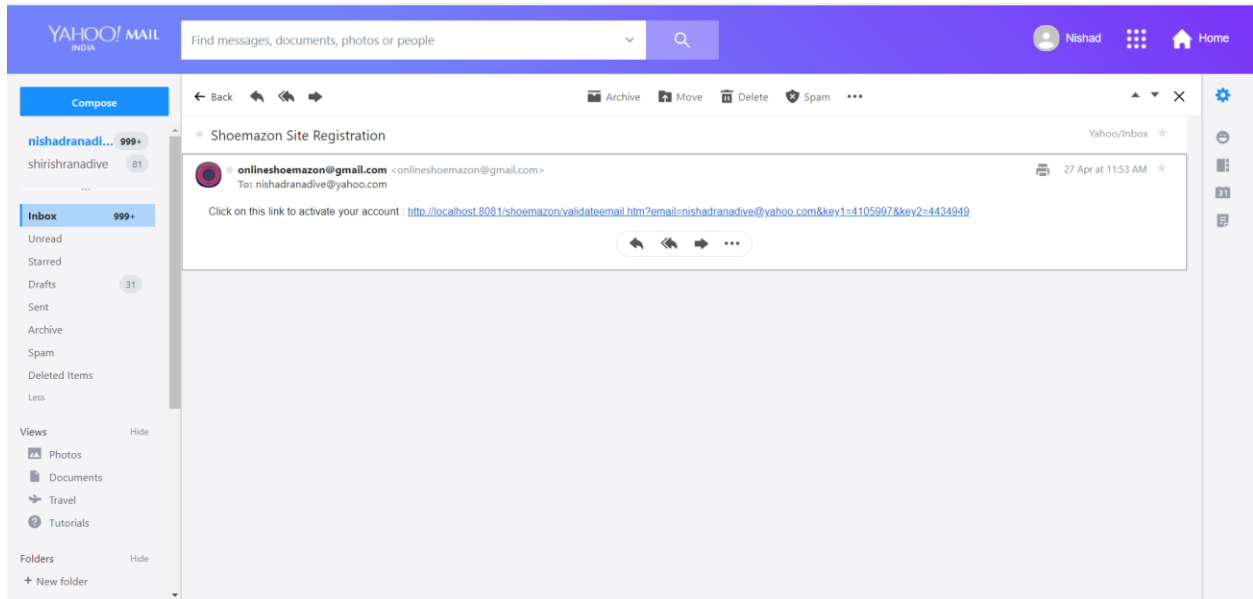
Download pdf for order

printorder.pdf 1 / 1

Order Details

Product Code	Product Name	Product Price	Product Quantity
S001	Adidas	100.0	1

Registration Email



APPENDIX

AdminController.java

```
package com.me.shoemazon;

import java.util.Date;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.ModelAttribute;
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 org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.me.shoemazon.dao.AccountDAO;
import com.me.shoemazon.dao.OrderDAO;
import com.me.shoemazon.dao.ProductDAO;
import com.me.shoemazon.models.OrderDetailInfo;
import com.me.shoemazon.models.OrderInfo;
import com.me.shoemazon.models.ProductMaster;
import com.me.shoemazon.pojos.Order;
import com.me.shoemazon.pojos.Product;

@Controller
public class AdminController {
    @RequestMapping(value = { "/orderList" }, method = RequestMethod.GET)
    public ModelAndView orderList(Model model, //
        @RequestParam(value = "page", defaultValue = "1") String pageStr,
        OrderDAO orderDAO, ModelMap map)
        throws Exception {

        int page = 1;
        try {
            page = Integer.parseInt(pageStr);
```



```
        } catch (Exception e) {
        }
        final int MAX_RESULT = 10;
        final int MAX_NAVIGATION_PAGE = 10;

        List<Order> result = orderDAO.listOrderInfo(page, MAX_RESULT,
MAX_NAVIGATION_PAGE);

        System.out.println(result);
        map.addAttribute("orderList", result);
        model.addAttribute("paginationProducts", null);

        // model.addAttribute("paginationResult", paginationResult);
        // return "orderList";
        return new ModelAndView("orderList", "map", map);
    }

    // POST: Save product
    @RequestMapping(value = { "/product" }, method = RequestMethod.POST)

    public ModelAndView productSave(HttpServletRequest request, AccountDAO
 userDao,ProductDAO productDAO, ModelMap map) throws Exception {
        System.out.println("inside Product save");
        String code= request.getParameter("code");
        String name = request.getParameter("name");
        String price = request.getParameter("price");
        System.out.println(code + name +price);
        Product product = new Product();
        product.setCode(code);
        product.setName(name);
        product.setPrice(Double.parseDouble( price));
        product.setCreateDate(new Date());
        System.out.println(product.getCode() + product.getName() + product.getPrice());
        Boolean saved = productDAO.addProduct(product);
        int count = 1;
        map.put("msgtyp", "add");

        if(saved) {
            map.put("msgfor", "success");
            map.put("rowcount", count);}
        else {
            map.put("msgfor", "error");
            map.put("msg", "Could not save product");
        }

        return new ModelAndView("add-success", "map", map);
    }
}
```

```
}
```

HomeController.java

```
package com.me.shoemazon;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.Random;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.apache.commons.mail.DefaultAuthenticator;
import org.apache.commons.mail.Email;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.SimpleEmail;
import org.omg.CORBA.Request;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import com.captcha.botdetect.web.servlet.Captcha;
import com.me.shoemazon.pojos.Account;

import com.me.shoemazon.dao.AccountDAO;

import com.me.shoemazon.dao.OrderDAO;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
```

```
/**
 * Simply selects the home view to render by returning its name.
 */
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Model model) {
    logger.info("Welcome home! The client locale is { }.", locale);

    Date date = new Date();
    DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG,
    DateFormat.LONG, locale);

    String formattedDate = dateFormat.format(date);

    model.addAttribute("serverTime", formattedDate );

    return "login";
}

@RequestMapping(value = { "/login" }, method = RequestMethod.GET)
public String login(Model model) {

    return "login";
}

@RequestMapping(value = { "/accountInfo" }, method = RequestMethod.POST)
public ModelAndView accountInfo(HttpServletRequest request, Model model, AccountDAO
userDAO, ModelMap map) {

    String username = request.getParameter("userName");
    String password = request.getParameter("password");
    try {
        Account u = userDAO.findAccount(username, password);
        request.getSession().setAttribute("user", u);
        request.getSession().setAttribute("role", u.getUserRole());
        if (u != null && u.isActive() == true) {
            Account vt=(Account)request.getSession().getAttribute("user") ;
//            System.out.println(userDetails.getPassword());
//            System.out.println(userDetails.getUsername());
//            System.out.println(userDetails.isEnabled());
            map.addAttribute("username",vt.getUserName());
            map.addAttribute("role", vt.getUserRole());

            return new ModelAndView("accountInfo", "map", map);

        } else if (u != null && u.isActive() == true) {
```

```
        map.addAttribute("errorMessage", "Please activate your account to login!");
        return new ModelAndView("error");
    } else {
        map.addAttribute("errorMessage", "Invalid username/password!");
        return new ModelAndView("error");
    }
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

return null;
//      System.out.println("inside account info");
//      System.out.println(request.getParameter("userName"));
//      UserDetails userDetails = (UserDetails)
userDAO.findAccount(request.getParameter("userName"));
//      System.out.println(userDetails);
//      System.out.println(userDetails.getPassword());
//      System.out.println(userDetails.getUsername());
//      System.out.println(userDetails.isEnabled());
//
//      model.addAttribute("userDetails", userDetails);
//      return "accountInfo";
}

@RequestMapping(value = "/create.htm", method = RequestMethod.GET)
public String showCreateForm() {

    return "user-create-form";
}

public void sendEmail(String useremail, String message) {
    try {
        Email email = new SimpleEmail();
        email.setHostName("smtp.googlemail.com");
        email.setSmtpPort(465);
        email.setAuthenticator(new DefaultAuthenticator("onlineshoemazon@gmail.com",
"Shopping@123"));
        email.setSSLonConnect(true);
        email.setFrom("no-reply@sheomazon.com"); // This user email does not
// exist
        email.setSubject("Shoemazon Site Registration ");
        email.setMsg(message); // Retrieve email from the DAO and send this
        email.addTo(useremail);
        email.send();
    } catch (EmailException e) {
        System.out.println("Email cannot be sent");
    }
}
```

```
    }  
    }  
    @RequestMapping(value = "/create.htm", method = RequestMethod.POST)  
    public String handleCreateForm(HttpServletRequest request, AccountDAO userDao,  
ModelMap map) {  
        Captcha captcha = Captcha.load(request, "CaptchaObject");  
        String captchaCode = request.getParameter("captchaCode");  
        HttpSession session = request.getSession();  
        if (captcha.validate(captchaCode)) {  
            String username= request.getParameter("username");  
            String password = request.getParameter("password");  
            Account user = new Account();  
            user.setUserName(username);  
            user.setPassword(password);  
            user.setUserRole("EMPLOYEE");  
            user.setActive(false);  
  
            try {  
                Account u = userDao.register(user);  
                Random rand = new Random();  
                int randomNum1 = rand.nextInt(5000000);  
                int randomNum2 = rand.nextInt(5000000);  
                try {  
                    String str = "http://localhost:8081/shoemazon/validateemail.htm?email=" +  
username + "&key1=" + randomNum1 + "&key2=" + randomNum2;  
                    session.setAttribute("key1", randomNum1);  
                    session.setAttribute("key2", randomNum2);  
                    sendEmail(username,  
                        "Click on this link to activate your account : "+ str);  
                } catch (Exception e) {  
                    System.out.println("Email cannot be sent");  
                }  
            } catch (Exception e) {  
                // TODO Auto-generated catch block  
                e.printStackTrace();  
            }  
        } else {  
            map.addAttribute("errorMessage", "Invalid Captcha!");  
            return "user-create-form";  
        }  
  
        return "user-created";  
    }  
  
    @RequestMapping(value = "validateemail.htm", method = RequestMethod.GET)
```

```
public String validateEmail(HttpServletRequest request, AccountDAO userDao, ModelMap
map) {

    // The user will be sent the following link when the use registers
    // This is the format of the email
    //
    http://hostname:8080/lab10/user/validateemail.htm?email=useremail&key1=<random_number>
    &key2=<body
    // of the email that when user registers>
    HttpSession session = request.getSession();
    String email = request.getParameter("email");
    int key1 = Integer.parseInt(request.getParameter("key1"));
    int key2 = Integer.parseInt(request.getParameter("key2"));
    System.out.println(session.getAttribute("key1"));
    System.out.println(session.getAttribute("key2"));

    if ((Integer)(session.getAttribute("key1")) == key1 &&
    ((Integer)session.getAttribute("key2")) == key2) {
        try {
            System.out.println("HI_____");
            boolean updateStatus = userDao.updateUser(email);
            if (updateStatus) {
                return "login";
            } else {

                return "error";
            }

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    } else {
        map.addAttribute("errorMessage", "Link expired , generate new link");
        map.addAttribute("resendLink", true);
        return "error";
    }

    return "login";

}

@RequestMapping(value = "/logout", method = RequestMethod.GET)
protected String logoutUser(HttpServletRequest request) throws Exception {
```

```
HttpSession session = (HttpSession) request.getSession();

try {

    session.invalidate();

}
catch(Exception e){
    System.out.println("error");
}
return "login";

}

@RequestMapping(value = { "/accountInfo" }, method = RequestMethod.GET)
public ModelAndView accountInfo(HttpServletRequest request, Model model, ModelMap
map) {

    Account vt=(Account)request.getSession().getAttribute("user") ;
//    System.out.println(userDetails.getPassword());
//    System.out.println(userDetails.getUsername());
//    System.out.println(userDetails.isEnabled());
    map.addAttribute("username",vt.getUserName());
    map.addAttribute("role", vt.getUserRole());

    return new ModelAndView("accountInfo", "map", map);

}
}
```

OrderController.java

```
package com.me.shoemazon;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.apache.commons.mail.DefaultAuthenticator;
import org.apache.commons.mail.Email;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.SimpleEmail;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.ModelAttribute;
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.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
```

```
import com.me.shoemazon.models.CartDetails;
import com.me.shoemazon.models.CustomerInfo;
import com.me.shoemazon.pojos.Account;
import com.me.shoemazon.pojos.OrderDetail;
import com.me.shoemazon.models.OrderDetailInfo;
import com.me.shoemazon.models.OrderInfo;
import com.me.shoemazon.cartvalidity.CartValidity;
import com.me.shoemazon.dao.OrderDAO;
import com.me.shoemazon.dao.ProductDAO;
```

@Controller

```
public class OrderController {
```

```
    // GET: Show Cart
```

```
    @RequestMapping(value = { "/shoppingCart" }, method = RequestMethod.GET)
    public String shoppingCartHandler(HttpServletRequest request, Model model) {
        CartDetails myCart = CartValidity.getCartInSession(request);

        model.addAttribute("cartForm", myCart);
        return "shoppingCart";
    }
```

```
    // GET: Enter customer information.
```

```
    @RequestMapping(value = { "/shoppingCartCustomer" }, method = RequestMethod.GET)
    public String shoppingCartCustomerForm(HttpServletRequest request, Model model) {
```

```
        CartDetails cartInfo = CartValidity.getCartInSession(request);
```

```
        // Cart is empty.
```

```
        if (cartInfo.isEmpty()) {
```

```
            // Redirect to shoppingCart page.
```

```
            return "redirect:/shoppingCart";
```



```
    }

    CustomerInfo customerInfo = cartInfo.getCustomerInfo();
    if (customerInfo == null) {
        customerInfo = new CustomerInfo();
    }

    model.addAttribute("customerForm", customerInfo);

    return "CartCustomerDetails";
}

// POST: Save customer information.
@RequestMapping(value = { "/shoppingCartCustomer" }, method = RequestMethod.POST)
public String shoppingCartCustomerSave(HttpServletRequest request, //
    Model model, //
    @ModelAttribute("customerForm") @Validated CustomerInfo customerForm, //
    BindingResult result, //
    final RedirectAttributes redirectAttributes) {

    // If has Errors.
    if (result.hasErrors()) {
        customerForm.setValid(false);
        // Forward to reenter customer info.
        return "shoppingCartCustomer";
    }

    customerForm.setValid(true);
    CartDetails cartInfo = CartValidity.getCartInSession(request);

    cartInfo.setCustomerInfo(customerForm);

    // Redirect to Confirmation page.
    return "redirect:/Confirmation";
}

// GET: Review Cart to confirm.
@RequestMapping(value = { "/Confirmation" }, method = RequestMethod.GET)
public String shoppingCartConfirmationReview(HttpServletRequest request, Model model) {
    CartDetails cartInfo = CartValidity.getCartInSession(request);

    // Cart have no products.
    if (cartInfo.isEmpty()) {
        // Redirect to shoppingCart page.
        return "redirect:/shoppingCart";
    } else if (!cartInfo.isValidCustomer()) {
```

```
// Enter customer info.
return "redirect:/shoppingCartCustomer";
}

return "Confirmation";
}

@RequestMapping(value = { "/Confirmation" }, method = RequestMethod.POST)
// Avoid UnexpectedRollbackException (See more explanations)
@Transactional(propagation = Propagation.NEVER)
public String shoppingCartConfirmationSave(HttpServletRequest request, Model
model, OrderDAO orderDAO, ProductDAO productDAO) {
    CartDetails cartInfo = CartValidity.getCartInSession(request);
    System.out.println(productDAO);
    // Cart have no products.
    if (cartInfo.isEmpty()) {
        // Redirect to shoppingCart page.
        return "redirect:/shoppingCart";
    } else if (!cartInfo.isValidCustomer()) {
        // Enter customer info.
        return "redirect:/shoppingCartCustomer";
    }

    try {
        orderDAO.saveOrder(cartInfo, productDAO);
    } catch (Exception e) {
        // Need: Propagation.NEVER?
        System.out.println(e);
        return "Confirmation";
    }
    // Remove Cart In Session.
    CartValidity.removeCartInSession(request);
    System.out.println("storeLastOrderedCartInSession");
    // Store Last ordered cart to Session.
    CartValidity.storeLastOrderedCartInSession(request, cartInfo);

    // Redirect to successful page.
    return "redirect:/shoppingConfirmation";
}

@RequestMapping(value = { "/shoppingConfirmation" }, method = RequestMethod.GET)
public String shoppingCartFinalize(HttpServletRequest request, Model model) {

    CartDetails lastOrderedCart = CartValidity.getLastOrderedCartInSession(request);

    if (lastOrderedCart == null) {
```

```
        return "redirect:/shoppingCart";
    }

    return "shoppingConfirmation";
}

@RequestMapping(value = "/ajaxservice.htm", method = RequestMethod.POST)
@ResponseBody
public String ajaxService(HttpServletRequest request)
{
    // String queryString = request.getParameter("course");
    // System.out.println("inside ajax");
    // @formatter:on

    CartDetails lastOrderedCart = CartValidity.getLastOrderedCartInSession(request);

    if (lastOrderedCart == null) {
        System.out.println("inside ajax false");
        return "Email Failed";
    }

    sendEmail(lastOrderedCart,request);
    System.out.println("inside ajax true");
    return "Email Send";
}

public void sendEmail(CartDetails lastOrderedCart,HttpServletRequest request) {
    try {
        Account u =(Account)request.getSession().getAttribute("user");
        Email email = new SimpleEmail();
        email.setHostName("smtp.googlemail.com");
        email.setSmtpport(465);
        email.setAuthenticator(new DefaultAuthenticator("onlineshoemazon@gmail.com",
"Shopping@123"));
        email.setSSLonConnect(true);
        email.setFrom("no-reply@sheomazon.com"); // This user email does not
        // exist
        email.setSubject("Shoemazon Site Registration ");
        email.setMsg("Your Order with OrderNumber:"+ lastOrderedCart.getOrderNum() + "
Confirmed"); // Retrieve email from the DAO and send this
        email.addTo(lastOrderedCart.getCustomerInfo().getEmail());
        email.send();
    } catch (EmailException e) {
        System.out.println("Email cannot be sent");
    }
}
```

```
@RequestMapping(value = { "/order" }, method = RequestMethod.GET)
public String orderView(Model model, @RequestParam("orderId") String orderId, OrderDAO
orderDAO) {
    OrderInfo orderInfo = null;
    if (orderId != null) {
        orderInfo = orderDAO.getOrderInfo(orderId);
    }
    if (orderInfo == null) {
        return "redirect:/orderList";
    }
    List<OrderDetailInfo> details = orderDAO.listOrderDetailInfos(orderId);
    orderInfo.setDetails(details);

    model.addAttribute("orderInfo", orderInfo);

    return "order";
}
}
```

ProductController.java

```
package com.me.shoemazon;

import java.io.IOException;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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.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.me.shoemazon.cartvalidity.CartValidity;
import com.me.shoemazon.dao.AccountDAO;
import com.me.shoemazon.dao.OrderDAO;
import com.me.shoemazon.dao.ProductDAO;
import com.me.shoemazon.models.CartDetails;
```

```
import com.me.shoemazon.models.PaginationResult;
import com.me.shoemazon.models.ProductMaster;
import com.me.shoemazon.pojos.Product;
```

```
@Controller
```

```
public class ProductController {
```

```
    @RequestMapping({ "/productList" })
    public ModelAndView listProductHandler(Model model, ProductDAO productDAO,
        @RequestParam(value = "name", defaultValue = "") String likeName,
        @RequestParam(value = "page", defaultValue = "1") int page, ModelMap
map) throws Exception {
        final int maxResult = 5;
        final int maxNavigationPage = 10;
        System.out.println("inside product list");
        List<Product> result = productDAO.queryProducts(page, maxResult,
maxNavigationPage, likeName);
        System.out.println(result);
        map.addAttribute("productlist", result);
        model.addAttribute("paginationProducts", null);
        return new ModelAndView("productList", "map", map);
    }
```

```
//    @RequestMapping(value = { "/productImage" }, method = RequestMethod.GET)
//    public void productImage(HttpServletRequest request, HttpServletResponse response, Model
model,
//    @RequestParam("code") String code, ProductDAO productDAO) throws Exception {
//    Product product = null;
//    if (code != null) {
//        System.out.println("inside image");
//        product = productDAO.findProduct(code);
//    }
//    if (product != null && product.getImage() != null) {
//        response.setContentType("image/jpeg, image/jpg, image/png, image/gif");
//        response.getOutputStream().write(product.getImage());
//    }
//    response.getOutputStream().close();
// }
```

```
    @RequestMapping(value = { "/product" }, method = RequestMethod.GET)
    public String product(Model model, @RequestParam(value = "code", defaultValue = "")
String code,
        ProductDAO productDAO) throws Exception {
        ProductMaster ProductMaster = null;
```

```
        if (code != null && code.length() > 0) {
            ProductMaster = productDAO.findProductInfo(code);
        }
        if (ProductMaster == null) {
            ProductMaster = new ProductMaster();
            ProductMaster.setNewProduct(true);
        }
        model.addAttribute("productForm", ProductMaster);
        return "product";
    }

    @RequestMapping({ "/buyProduct" })
    public String listProductHandler(HttpServletRequest request, Model model, //
        @RequestParam(value = "code", defaultValue = "") String code, ProductDAO
        productDAO) throws Exception {

        Product product = null;
        if (code != null && code.length() > 0) {
            product = productDAO.findProduct(code);
        }
        if (product != null) {

            // Cart info stored in Session.
            CartDetails cartInfo = CartValidity.getCartInSession(request);

            ProductMaster productInfo = new ProductMaster(product);

            cartInfo.addProduct(productInfo, 1);
        }
        // Redirect to shoppingCart page.
        return "redirect:/shoppingCart";
    }

    @RequestMapping({ "/shoppingCartRemoveProduct" })
    public String removeProductHandler(HttpServletRequest request, Model model, //
        @RequestParam(value = "code", defaultValue = "") String code, ProductDAO
        productDAO) throws Exception {

        Product product = null;
        if (code != null && code.length() > 0) {
            product = productDAO.findProduct(code);
        }
        if (product != null) {

            // Cart Info stored in Session.
            CartDetails cartInfo = CartValidity.getCartInSession(request);
```

```
        ProductMaster productInfo = new ProductMaster(product);

        cartInfo.removeProduct(productInfo);

    }
    // Redirect to shoppingCart page.
    return "redirect:/shoppingCart";
}
}
```

ReportController.java

```
package com.me.shoemazon;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
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.me.shoemazon.common.PdfReportView;
import com.me.shoemazon.dao.OrderDAO;
import com.me.shoemazon.models.OrderDetailInfo;
import com.me.shoemazon.models.OrderInfo;
import com.me.shoemazon.pojos.Order;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

@Controller
public class ReportController {

    @RequestMapping(value = "/printorder.pdf", method = RequestMethod.GET)
    public ModelAndView showPrintForm(HttpServletRequest request,
    @RequestParam("orderId") String orderId,
        OrderDAO orderDAO, ModelMap map) {
        OrderInfo orderInfo = null;

        List<OrderDetailInfo> details = null;
        try {
            if (orderId != null) {
                orderInfo = orderDAO.getOrderInfo(orderId);
                details = orderDAO.listOrderDetailInfos(orderId);
            }
        }
    }
}
```

```
        map.addAttribute("orderdetails", details);
    }
} catch (Exception e) {
    System.out.println("Inside All Applications List Exception" + e);
}

return new ModelAndView("pdfView", "order", details);
}
}
```

POJOS

Account.java

```
package com.me.shoemazon.pojos;
import java.io.Serializable;
import java.util.Collection;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

@Entity
@Table(name = "Accounts")
public class Account implements Serializable {

    public static final String ROLE_MANAGER = "MANAGER";
    public static final String ROLE_EMPLOYEE = "EMPLOYEE";

    private String userName;
    private String password;
    private boolean active;
    private String userRole;

    @Id
    @Column(name = "User_Name", length = 50, nullable = false)
    public String getUserName() {
```



```
        return userName;
    }

    public void setUsername(String userName) {
        this.userName = userName;
    }

    @Column(name = "Password", length = 20, nullable = false)
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Column(name = "Active", length = 1, nullable = false)
    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    @Column(name = "User_Role", length = 20, nullable = false)
    public String getUserRole() {
        return userRole;
    }

    public void setUserRole(String userRole) {
        this.userRole = userRole;
    }

    @Override
    public String toString() {
        return "["+ this.userName+", "+ this.password+", "+ this.userRole+"]";
    }

}
```

Order.java

```
package com.me.shoemazon.pojos;
import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

@Entity
@Table(name = "Orders", //
uniqueConstraints = { @UniqueConstraint(columnNames = "Order_Num") })
public class Order implements Serializable {

    private static final long serialVersionUID = -2576670215015463100L;

    private String id;
    private Date orderDate;
    private int orderNum;
    private double amount;

    private String customerName;
    private String customerAddress;
    private String customerEmail;
    private String customerPhone;

    @Id
    @Column(name = "ID", length = 50)
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    @Column(name = "Order_Date", nullable = false)
    public Date getOrderDate() {
        return orderDate;
    }

    public void setOrderDate(Date orderDate) {
        this.orderDate = orderDate;
    }
}
```

```
}

@Column(name = "Order_Num", nullable = false)
public int getOrderNum() {
    return orderNum;
}

public void setOrderNum(int orderNum) {
    this.orderNum = orderNum;
}

@Column(name = "Amount", nullable = false)
public double getAmount() {
    return amount;
}

public void setAmount(double amount) {
    this.amount = amount;
}

@Column(name = "Customer_Name", length = 255, nullable = false)
public String getCustomerName() {
    return customerName;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

@Column(name = "Customer_Address", length = 255, nullable = false)
public String getCustomerAddress() {
    return customerAddress;
}

public void setCustomerAddress(String customerAddress) {
    this.customerAddress = customerAddress;
}

@Column(name = "Customer_Email", length = 128, nullable = false)
public String getCustomerEmail() {
    return customerEmail;
}

public void setCustomerEmail(String customerEmail) {
    this.customerEmail = customerEmail;
}
```

```
@Column(name = "Customer_Phone", length = 128, nullable = false)
public String getCustomerPhone() {
    return customerPhone;
}

public void setCustomerPhone(String customerPhone) {
    this.customerPhone = customerPhone;
}
}
```

OrderDetails.java

```
package com.me.shoemazon.pojos;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.ForeignKey;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "Order_Details")
public class OrderDetail implements Serializable {

    private String id;
    private Order order;

    private Product product;
    private int quantity;
    private double price;
    private double amount;

    @Id
    @Column(name = "ID", length = 50, nullable = false)
    public String getId() {
        return id;
    }

    public void setId(String id) {
```

```
        this.id = id;
    }

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "ORDER_ID", nullable = false )
    public Order getOrder() {
        return order;
    }

    public void setOrder(Order order) {
        this.order = order;
    }

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "PRODUCT_ID", nullable = false )
    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
        this.product = product;
    }

    @Column(name = "Quantity", nullable = false)
    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    @Column(name = "Price", nullable = false)
    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @Column(name = "Amount", nullable = false)
    public double getAmount() {
        return amount;
    }
}
```

```
        public void setAmount(double amount) {  
            this.amount = amount;  
        }  
    }  
}
```

Product.java

```
package com.me.shoemazon.pojos;  
  
import java.io.Serializable;  
import java.util.Date;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Lob;  
import javax.persistence.Table;  
import javax.persistence.Temporal;  
import javax.persistence.TemporalType;  
  
@Entity  
@Table(name = "Products")  
public class Product implements Serializable {  
  
    private String code;  
    private String name;  
    private double price;  
  
    // For sort.  
    private Date createDate;  
  
    public Product() {  
    }  
  
    @Id  
    @Column(name = "Code", length = 20, nullable = false)  
    public String getCode() {  
        return code;  
    }  
  
    public void setCode(String code) {  
        this.code = code;  
    }  
}
```

```
@Column(name = "Name", length = 255, nullable = false)
public String getName() {
    return name;
}

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

@Column(name = "Price", nullable = false)
public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

@Temporal(TemporalType.TIMESTAMP)
@Column(name = "Create_Date", nullable = false)
public Date getCreateDate() {
    return createDate;
}

public void setCreateDate(Date createDate) {
    this.createDate = createDate;
}

public void setImage(byte[] image) {
    this.image = image;
}
}
```