Assignment 5
JSP, Servlet, and MVC
Order Processing, Secure Administration
Points: 100

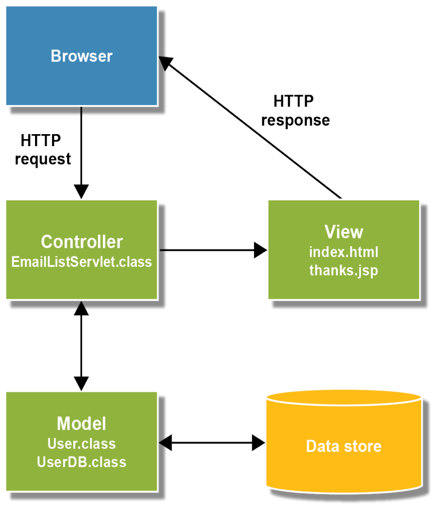
This assignment is to be done by the previously defined teams of two students. If you have not done so, you should establish a sensible way to share your project development work. You **MUST NOT** make your assignment code available publicly, so do not use a public repository service such as GitHub – any such repository must be private.

Assignment Description:

This assignment is intended to apply previous techniques to enable order processing with storage of full order details, as well as to provide experience with declarative security techniques for administrative access.

In this assignment you will update your previous JSP/Servlet MVC web application, according to the following specifications:

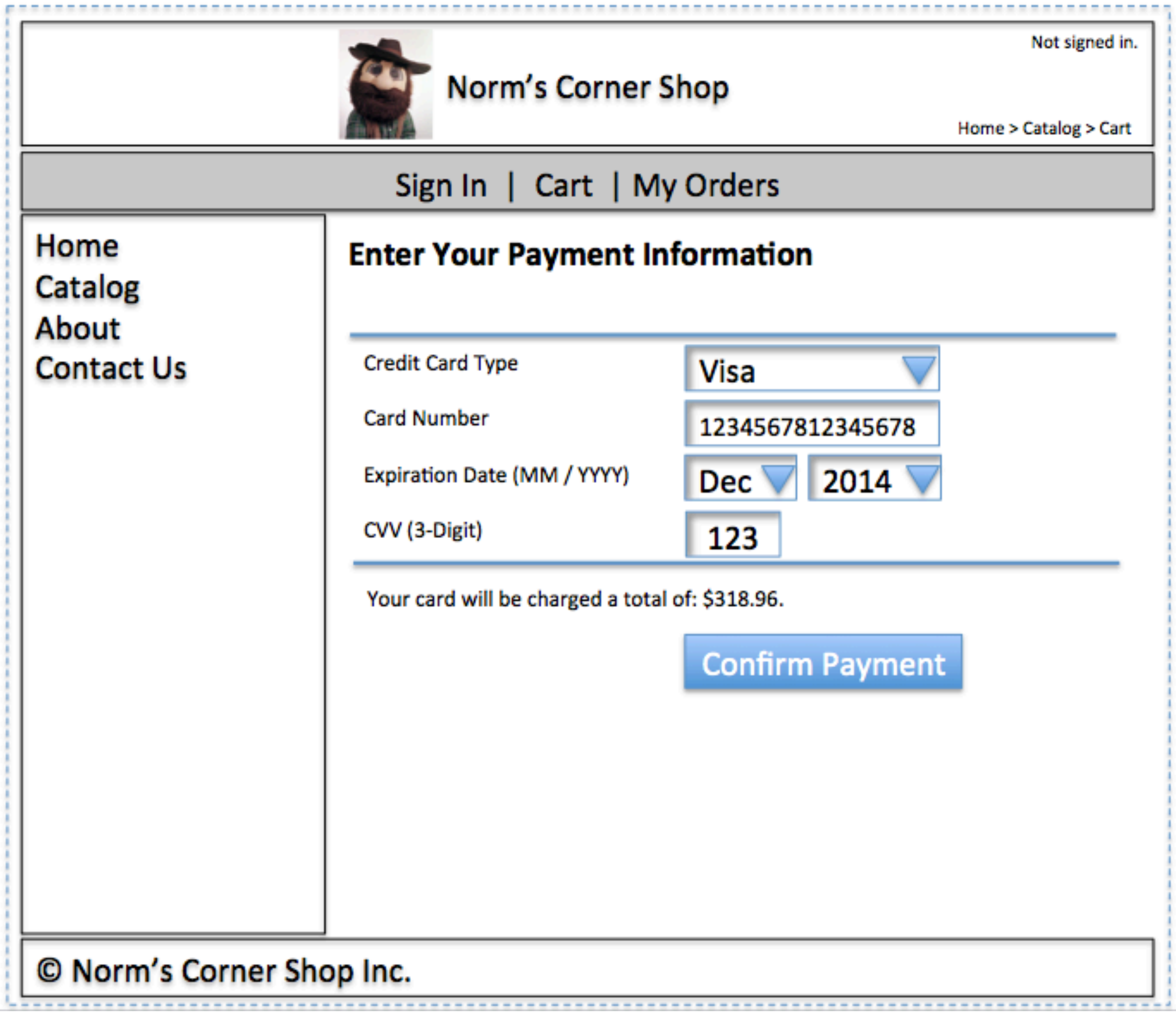
1. Correct any errors identified or partial/missing functionality from the previous assignment.
2. All structure, design, and content requirements from previous assignments are mandatory, unless explicitly updated in this assignment description.
3. Use JavaBeans to implement the business layer of the application (**model**).
4. Use JSP pages to present the **view** to the browser.
5. Use Servlet pages to **control** the flow of the application.
6. Functionality that does not follow the assignment specifications will not receive credit.



Add New JSP Site Pages for Payment, Order Listing, and Administration

Create a new JSP page for each of the following screens. While it is not necessary to do so, it may be useful to create them first as static design prototype pages (with placeholder data), in order to get the structuring needed, and then convert them to dynamic pages using bean data.

Filename: payment.jsp – this page is linked into the application flow from the "Purchase" button on the order.jsp page. It should provide for submission of common credit card details for several types of cards. The "Confirm Payment" should be linked to the OrderController Servlet with an "action" request parameter with the value of "confirmOrder".

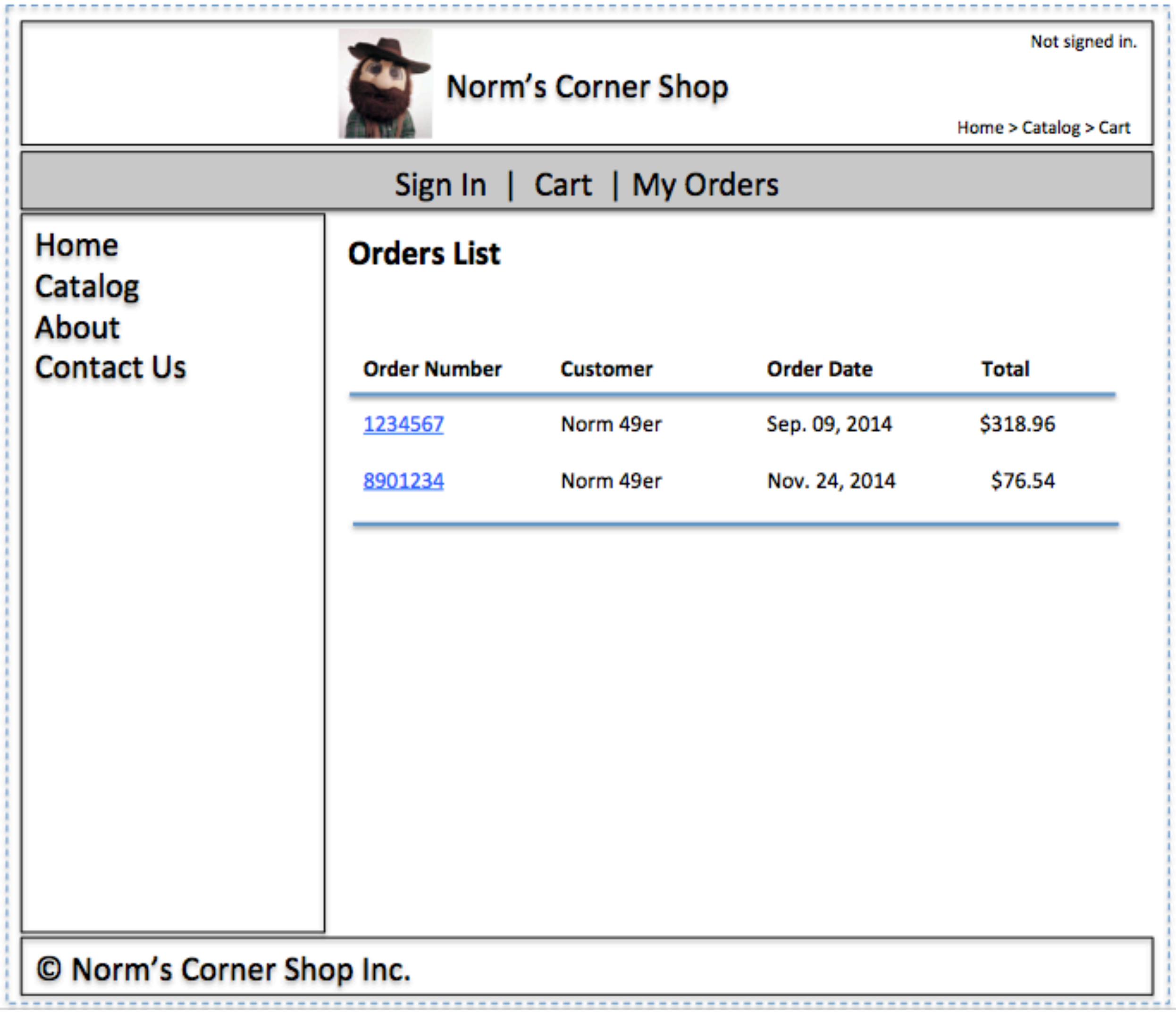


Enter Your Payment Information

Credit Card Type	Visa
Card Number	1234567812345678
Expiration Date (MM / YYYY)	Dec 2014
CVV (3-Digit)	123

Your card will be charged a total of: $318.96.

Filename: orderlist.jsp – this page is linked into the application flow from two separate places. The first place is from the "My Orders" link in the user-specific navigation area. The second place is by accessing an administrative URL. A user should only be able to access his/her orders while the page displays all saved orders for an administrator.



Not signed in.

Norm's Corner Shop

Home > Catalog > Cart

Sign In | Cart | My Orders

Home
Catalog
About
Contact Us

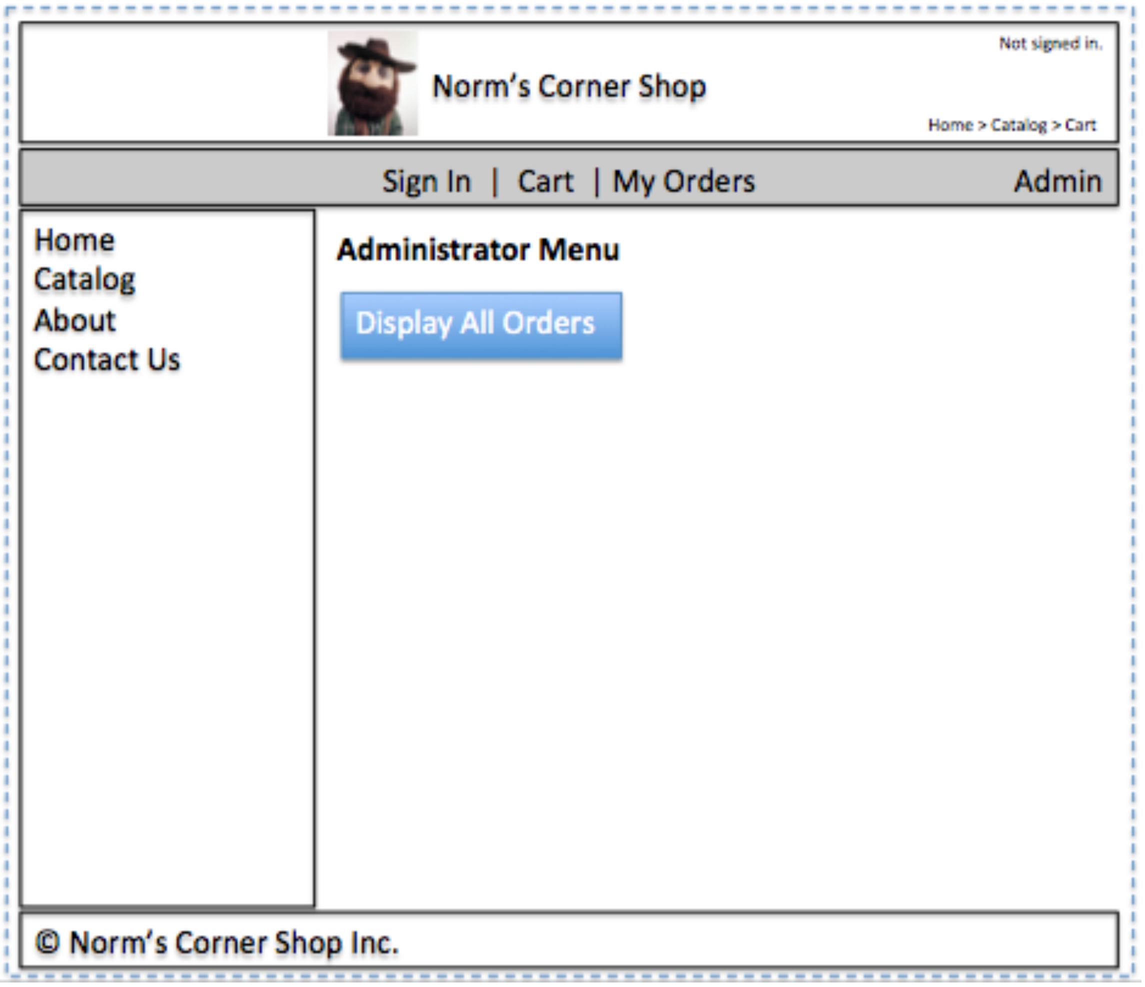
Orders List

Order Number	Customer	Order Date	Total
1234567	Norm 49er	Sep. 09, 2014	$318.96
8901234	Norm 49er	Nov. 24, 2014	$76.54

© Norm's Corner Shop Inc.

Filename: admin.jsp – this page is linked into the application from the home page.

Add a link on your homepage for the administration access. This link should be linked to the AdminController servlet. The “Display All Orders” button should be linked to the AdminController servlet with an “action” request parameter with the value of “viewOrders”.



<u>Add OrderItem and Order Tables to the Database</u>

Add two tables to your application database to store orders. You will need an Order table and an OrderItem table. For the OrderItem table the Order Number and the Product Code are the primary keys. The Order Number is the primary key for the Order table.

The following is a **<u>template</u>** table creation script that you can use to start out. All of the columns listed here should be in your table definitions, but you may need to modify them to reflect the naming scheme and data types you used in your OrderItem and Order JavaBeans from the previous assignments.

```
CREATE TABLE `OrderItem` (
        `OrderNumber` int(50) NOT NULL,
        `ProductCode` int(11) NOT NULL,
        `Quantity` int(11) NOT NULL,
         PRIMARY KEY (`OrderNumber`,`ProductCode`)
);
CREATE TABLE `Order` (
        `OrderNumber` int(50) NOT NULL AUTO_INCREMENT,
        `Date` date NOT NULL,
        `UserID` int(11) NOT NULL,
        `TaxRate` float(11,0) NOT NULL,
        `TotalCost` double(50,0) NOT NULL,
        `Paid` bit(1) NOT NULL,
        PRIMARY KEY (`OrderNumber`)
);
```

Update the link for "My Orders"

- When a user clicks the "My Orders" link this should send a request to the OrderController servlet with an "action" request parameter with the value of "viewOrders".

Add Logic to OrderController to view "My Orders"

- If the action parameter validates to a value of "viewOrders":
    - Check the session for "theUser" attribute
    - If no "theUser" attribute is set dispatch to catalog JSP view.
    - If there is a valid User bean stored in the session
        - Retrieve the list of orders for that user from the database. If a user doesn't have any saved orders this should be empty.
        - Create a new Order bean for each order and add that to a list/collection of orders. Again, if the user does not have any saved orders this should be an empty list.
        - Add the list of orders to the session object as "theOrders".
        - Dispatch to the **orderlist.jsp** view

Add Logic to OrderController to Save Order Items and Orders to the Database

When a user clicks the "Confirm Payment" button from the payment.jsp page a new entry should be added to the Order table for that user as well as the corresponding entries in the OrderItem table. Fields must contain the correct information associated with that order.

- If the action parameter validates to a value of "confirmOrder":
    - Save order details to Order and OrderItem tables.
        - Add a new order in the Order table

        - Add a new OrderItem for each item in the order with the corresponding order number.
    - Dispatch to the invoice page with “Paid In Full” message and no “Back To Cart” or “Purchase” links.

Add an AdminController Servlet

- Checks the http request for a parameter called “action”
    - If there is no action parameter, or it has an unknown value dispatch to the admin JSP view.
    - If there is an action parameter, validate that its value is “viewOrders”
        - Retrieve the list of orders from the database
        - Create a new Order bean for each order and add that to a list/collection of orders
        - Add the list of orders to the session object as “theOrders”
        - Dispatch to the **orderlist.jsp** view

Setup Declarative Security Constraint

Accessing an administrative URL (AdminController) should prompt the user to authenticate him/her self. Restrict access to this URL by setting up a Form-based authentication.
Users with an “admin” role should be able to access this URL. Setup at least one admin entry in your application security realm. You can choose any type of security realm implementation (UserDatabaseReal, JDBCRealm or DataSourceRealm).

Database Creation

In addition to the standard WAR file for this assignment, you will submit a text file (plain text, NOT Word or RTF or PDF or any other fancy document format). This file will contain all of the SQL statements that you use to (1) create, and (2) populate your database. Your script will look very similar to the script used in assignment 4. Your database script file must be called:

hw5_create_db.sql

Extra Credit (choose ONLY one) 20 points

1. Add the functionality to allow users to register/login to your application
2. Add the functionality to allow an admin to edit/add items to your application product catalog.

Moodle Assignment Submissions:

What to submit to Moodle (Email submissions will NOT be accepted):

1. **Hw5.war** - An archive of the entire web application (project) stored in a standard WAR file. You must ensure that the java source files are included as part of the archive. The WAR file will be imported into Netbeans for grading and may be required to be deployed as part of the submission.
2. **Hw5_create_db.sql** – SQL script file to create and populate the application's database.
3. **info.pdf** – PDF document with the following assignment information :
    a) Detail your security constraint implementation and provide the username and password for the admin access.
    b) Explanation of status and stopping point, if incomplete.
    c) Explanation of additional features, if any.
    d) Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?
    e) Discuss division of labor specifying who did what and why this is a fair and equal split (team)

Finally, set up a time to demo your assignment to your grader using the process described in the course Moodle site. All team members should be present at the demo. Students should be prepared to answer questions posed by the grader about their work. **Failing to demonstrate the assignment to the grader will result in no credit for all team members**.