Assignment 2
JSP, Servlet, and MVC
Points: 100

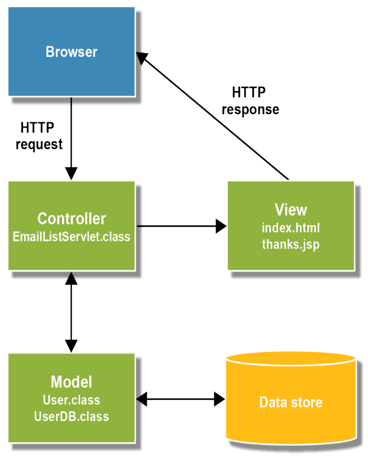
This assignment is to be done by a team of two students. Your **first step of the assignment** is to ensure that you have documented your team membership to course staff in the manner your instructor has specified. There is a fair amount of work to be done, but there are two people working the assignment together. You should establish a sensible way to share your project development work. You **MUST NOT** make your assignment code available publicly, as this would have to be treated as a violation of UNCC academic integrity policy. So do not use a repository service such as GitHub where all projects are publicly viewable – any such repository must be private.

This assignment is intended to familiarize students with Servlet/JSP web development. The goal is to use the MVC design pattern to structure your web application most developers consider MVC pattern as the best practice. The assignment assumes student familiarity with static HTML design and will involve use of JSP pages for the presentation, servlets for the controller and JavaBeans for the business logic.

Assignment Description:

In this assignment you will develop JSP/Servlet pages using the MVC pattern, according to the following specifications:

1. Select one (and only one) team member's e-commerce application from the previous assignment as the basis for your team's homework.
2. Correct any errors identified in the selected HTML5 prototype before proceeding.
3. All structure, design, and content requirements from the previous assignment are mandatory, unless explicitly updated in this assignment description.
4. Use JavaBeans to implement the business layer of the application (**model**).
5. Use JSP pages to present the **view** to the browser.
6. Use Servlet pages to **control** the flow of the application.



Convert Existing Pages to JSP (View Prototypes)

As an initial step, convert all 4 pages from the first assignment from HTML5 (.html) to JSP (.jsp). Remember that all pages generated from JSPs must still meet the HTML5 validation requirement. **All following site structure requirements must use this breakdown.**

- Separate the common design elements for header, footer, and navigation into individual .jsp files.
    - Filenames: header.jsp, user-navigation.jsp, site-navigation.jsp, footer.jsp
- For each of the 4 main pages
    - Remove the common design element content from the page
    - Use a JSP include directive or include action to re-include that content from the individual files. Do not use JSTL imports.

Add Return To Catalog Link / Button on Individual Item/Product Page

Individual item display should include an application "back" link to return to the catalog. Add such a link, sensibly located, in the main content area for the individual item / product page.

Add New JSP Site Pages Linked from General Navigation

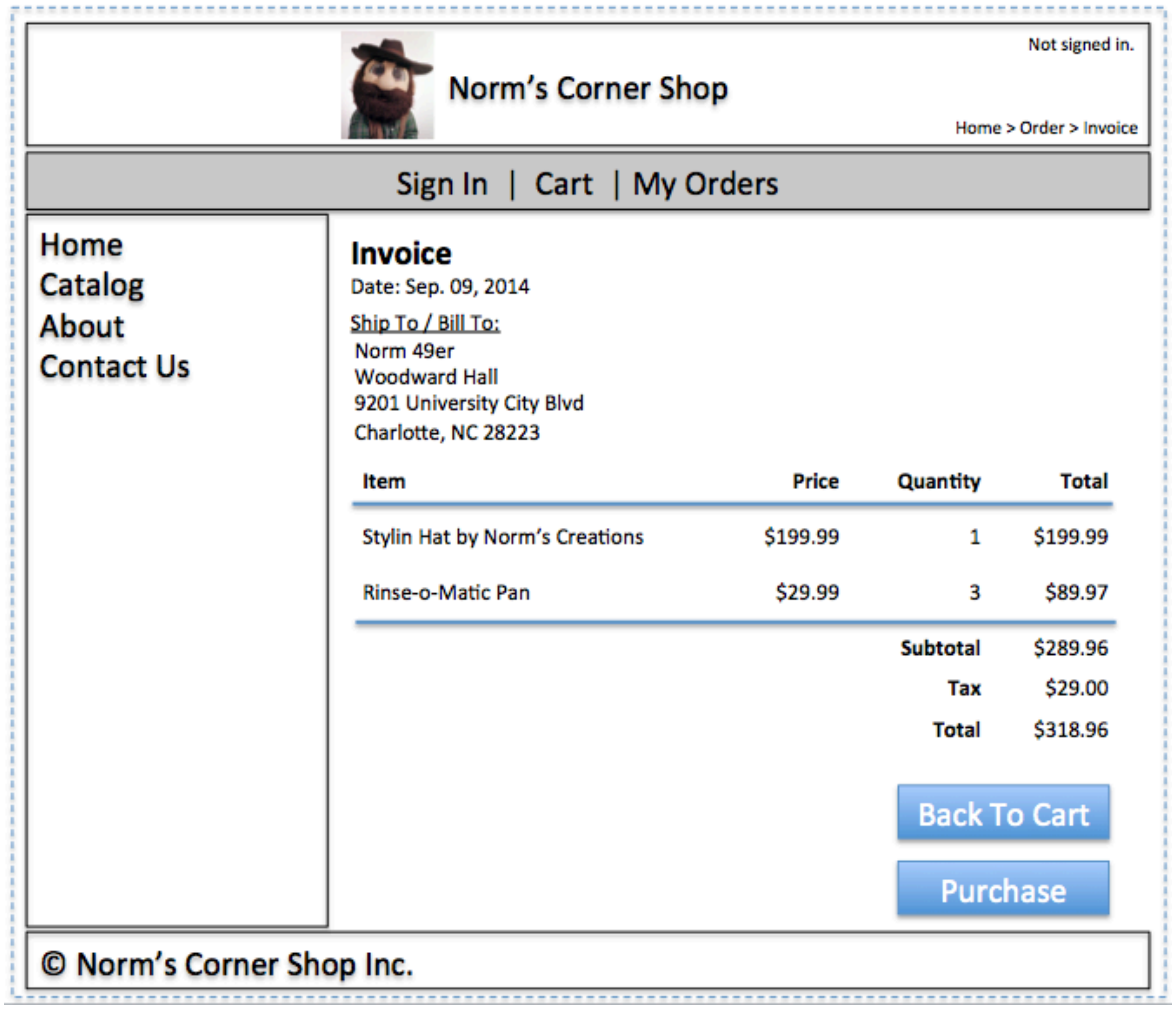
If not already part of the project, add the following pages using the required site structure.

- about.jsp – provides background and detail about the e-commerce site.
- contact.jsp – provides contact details for the e-commerce site.

Add New JSP Site Page for Final Order Details

Create a new (static, design prototype) JSP page that lists final invoice details. The page should include placeholder data in keeping with your own site's theme for the content indicated.

Filename: order.jsp



Invoice

Date: Sep. 09, 2014

Ship To / Bill To:

Norm 49er
Woodward Hall
9201 University City Blvd
Charlotte, NC 28223

Item	Price	Quantity	Total
Stylin Hat by Norm's Creations	$199.99	1	$199.99
Rinse-o-Matic Pan	$29.99	3	$89.97
		Subtotal	$289.96
		Tax	$29.00
		Total	$318.96

Create JavaBeans for Business Objects (Model)

This assignment will make use of JavaBeans for the Model, in order to represent the main data elements / business objects being used. Remember that a JavaBean is a Java class that (1) provides a zero-argument constructor; (2) provides get and set methods for all of its private instance variables that follow standard Java naming conventions; and (3) implements the Serializable interface. Create JavaBeans for the following Model elements, with the specified instance variables and additional methods. Bean class names MUST use the specified names.

Product

- Product Code – unique identifier for the product, you should select a sensible format (alphanumeric string, say) that will be standard across your products. Choose a representation that can be used in a straightforward way for form submissions, or form submission parameter names, and can be used as part of URLs for storing data such as images that will be associated with the product.
- Product Name
- Catalog Category – this is the category that will be used to select or arrange products by section or type in the catalog.
- Description
- Price
- String getImageURL() – URL that can be used in your pages, pointing to an image file within the project for your product. Generated from Product Code.

Create Utility Classes for Persistent Data

For initial development, we will use a hard-coded "database" to represent your sets of products. You should hard-code a fixed set of products within the following class. As in the first assignment, you must have at least 2 categories of product, with at least 3 products per category. Note that this means you will need to create the individual data for the additional products, since there was only individual data for one representative product required in the first assignment.

ProductDB

- Hard-Coded set of product details (your choice on how to represent internally, but must be converted into a list/collection of Product beans when required)
- List/Collection<Product> getProducts() – returns a set of all the products in the hardcoded "database"

Create Servlet for Business Logic (Controller)

For this assignment, you will implement a limited part of the dynamic e-commerce functionality for the catalog and items. You must use ONLY the http request object to pass data.

Implement the following Controller servlet to operationalize the business logic. You will need to send parameters as part of the GET or POST http requests from link / button / form submissions as the context information that tells the controller how to proceed.

CatalogController.java

- Loads the catalog / database of products.
- Checks the http request for a parameter called "productCode"
    - If there is a productCode parameter, validate that its value matches your product code format and is a valid product code.
    - If the product code is valid
        - Add a bean for the specified product to the http request object
        - Dispatch to the correct JSP view for **individual item display**
    - If the product code is not valid, display the catalog as if no code had been provided
    - If there is no productCode parameter, dispatch to the correct JSP view for **catalog display**. Display the full catalog listing, separated by category as in the first assignment, but with entries created dynamically from the catalog list

- **Mandatory For Graduate Students (Optional extra credit for Undergraduate students)** – additional catalog controller functionality:
    - Display one category of products
        - Checks the http request for a parameter called "catalogCategory"
        - If there is a catalogCategory parameter, validate that its value matches your catalog category format and is a valid category
        - If the catalog category is valid dispatch to catalog view only displaying that category and items
        - If the catalog category is not valid, display the catalog as if no category had been provided

Update JSP Views to Include Dynamic Content From Bean Data

The catalog and item views will now receive some kind of dynamic data. Update the JSP views to replace all of the static placeholder information with the dynamic data using JSP functionality to access bean data.

Update Form / Button Actions and Links in the Site to Dispatch Correctly

Each of the places in the site where a user can take an action that uses the dynamic data from this assignment should be updated to make the appropriate link or GET or POST request with the necessary parameters or form data. Updates must be made in all appropriate places where action would reasonably be indicated from the first assignment prototypes.

Assignment Submissions

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

1. hw2.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. info.pdf – PDF document with the following assignment information :
    a. Explanation of additional features, if any.
    b. Explanation of status, stopping point, and issues if incomplete.
    c. Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?
    d. Discuss in detail the division of labor, specifying who did what and why this is a fair and equal split.

Finally, set up a time to demo your assignment to your grader using the process described in the course Moodle site. 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.