Programming Project
Bookstore Web Application (Part 2)
Due: Tuesday, April 5, 11.59PM (Blackboard)

Rigel Gjomemo

March 29, 2016

1 Description

The purpose of this project is to make you familiar with the way in which web applications are written and supported by a backend database. In this project, you will need to create a web application to implement a basic online book store. The web application will allow registered users to browse through the books and their descriptions, place orders, add reviews, and so on.

The project will be divided into several parts of increasing complexity. This document describes the second part, which is dedicated to adding more functionality to the web application.

Important Note: Since this is a database class, you are not expected to create a fancy client side of the application. However, you will need to become familiar with some basic HTML (e.g., forms). You will not be graded on how good the web site looks, but on the correct implementation of the functionality described in the rest of this document.

2 Setup

We will use the Apache Tomcat web server [1] and JSP [2] for this first project. JSP is a Java-based server side language, similar to PHP and ASP.net. In JSP, one writes Java code to process input from the user and send responses. The Java code can be mixed with HTML content and is executed on the server.

[1] https://tomcat.apache.org/download-70.cgi

[2] http://www.tutorialspoint.com/jsp/

2.1 Setup

To make sure there are no glitches during the submission, you will work on Eclipse and submit the Eclipse project. Create a dynamic web project on Eclipse and put the files from first project inside the WebContent directory of the project (this is the default name). That is, everything that was inside the `bookstore` directory in the first project goes in the folder WebContent in your Eclipse project. When running your project on Eclipse, you can specify an existing Tomcat installation or download and create a new Tomcat server.

3 Project Description

This section describes the high level functionality that you need to add to your web application and the different entities and how they interact with one another. You are free to come up with any design for the database that you deem useful.

3.1 Functionality

In the following paragraphs, the word MUST is in front of the required functionality. You can add more functionality and pages (which you should describe in your writeup, see Deliverables section below). This is not the only possible structure of the web application, however, and it is open for discussion. You may be able to change it if you want, provided the functionality does not change, and only after a discussion with me and/or the TA. In any case, you must provide a clear description of the workflow of your application in your writeup. In addition to logging users in, having the users order books and check them out (done in the first part of the project), in this project you must:

AddUser.jsp.
Function: Add a new user. This is a 'sign up' page for a user. It has a form where a user enters the necessary information (username, password, address, credit card) and clicks on a 'sign up' button. Once the button is clicked, the user's data are inserted in the users table and the user can then log in to the web site.

Logout.
The users must be able to log out (duh!). You must create an html snippet to include in every page of the web site (a header file), after the user is logged in. This html snippet should just display the user's name and a logout button. Once the user clicks on that button, he/she is logged out. In particular, the session must become invalid (check out the documentation on how to do this). The orders that the user made must not be deleted, (think of a shopping cart surviving between different logouts and logins).

Administrator.
The web site has two different user types, normal users (the ones we have been dealing with so far)

and administrators. You should modify the `users` table to keep track of the user type. The first administrator is manually inserted in the database (via an insert query).

The administrator has a user name and password and access to additional administration web pages, described below.

User Management. This page (or set of pages) is reachable by the administrator user only. In case a normal user tries to navigate to the page, the web application should send an HTTP 401 code (unauthorized) to the browser. The web page should give the administrator the ability to modify user's information, delete them, or add them. You are free to implement this as you prefer. The basic idea is to have one form where the administrator user can search for a specific user and have her data displayed and one or more buttons (or checkboxes) that do the modification (e.g., change of address), or the deletion of the user. Also, note that you do not have to use only one page for this functionality. You can have for example a page where you search for a user and one or more pages where you delete the user and so on.

Statistics. This page (or set of pages) is reachable by the administrator user only. In case a normal user tries to navigate to the page, the web application should send an HTTP 401 code (unauthorized) to the browser. This page (or set of pages) gives the administrator the ability to track the following:

1. **Order History for a user**. Given a username and a date, the page must return the orders that that user placed after that date. These orders may include those in `orders_history` and those included in the (current) orders, that is those that have not been check out yet.
2. **Statistics about orders**. The administrator must be able to see: 1) the number of orders and total sales between two dates, 2) given a book's title, the sales of that book between two dates, 3) given an author's name, the sales of books from that author between two dates, 3) a list of the books ranked by sales (amount of money or number of orders), between two dates.

4 Deliverables

1. **Code**. Submit in a zip file (not tar.gz) the Eclipse folder of your project. Before compressing the Eclipse folder, you must dump your database to a SQL dump file named `bookstore.sql`. Then place this file inside the Eclipse folder. Then create a zip file of your Eclipse folder and submit it on Blackboard. Mysqldump instructions can be found at `https://dev.mysql.com/doc/refman/5.7/en/mysqldump-sql-format.html`.
2. **Writeup**. You must include a writeup that gives a detailed description of your application. In particular, it must contain the workflow (that is what each page does, and how the pages are connected to each other). The writeup must also contain a description of how we can actually use the web application, so that we can grade it correctly. Remember, we do not know the usernames and passwords that you are using, so include at least one in the writeup.

   The writeup must also contain feedback to us about the project. What level of difficulty it was, what part did you find the hardest to do, what was the most interesting part, what

suggestions for improvement do you have, and so on.

Grading. Grading will be out of 100%. Roughly one third of the grade will be given for the database design. That is, if the database contains all the information required to implement the functionality described in this document, you will get 33% of the grade. We are not grading on the quality of the design yet. However, a simple design will make your life easier when you write the rest of the application. Roughly 55% of the grade will be given for the correct implementation of the queries, the processing of the results, and the responses to the user, as well as the quality of the HTML. Again, nothing fancy is required on the client side. However, the pages should look somewhat orderly, this is why using tables is a good idea. The final 10% of the grade will be given for clarity of your writeup.

Suggestions and Hints.

1. Start by creating the HTML pages and how they would look like. You can use HTML editors for this. Adobe Dreamweaver is a great one, but expensive. There are free ones that you can get online. Eclipse has a plugin for that too.

2. Identify inside each HTML page, what is the content that comes from the database and what is the static content.

3. Start by adding simple JSP code to your HTML file and by testing it.

4. Implement the different functionalities one at a time.

5. START EARLY!!!!

6. ASK QUESTIONS!!!