Programming Project 1
Bookstore Web Application (Part 1)
Due: Thursday, March 3, 11.59PM (Calculus and Blackboard)

Rigel Gjomemo

February 25, 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 three parts of increasing complexity. This document describes the first part, which is dedicated to the setup and to getting familiar with the JSP technology, as well as to the creation of the first basic functionality of 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 Apache Tomcat Setup

After you download the zip file from the Apache Tomcat web site, unzip it. The path of the unzipped folder is referred to by the variable `CATALINA_HOME` in the documentation and in this document. To start Tomcat, run from a terminal the file `CATALINA_HOME/bin/startup.sh` (or `CATALINA_HOME/bin/startup.bat` if on Windows). You may need to make these files executable first. To test that Tomcat is running, point your browser to `localhost:8080`. You should see the apache Tomcat welcome page.

The files of your application must go inside `CATALINA_HOME/webapps`. Create a directory called `bookstore` inside that directory and place your JSP files there. Your web site will be accessible from the browser at `localhost:8080/bookstore/...`. Libraries needed by your application (e.g., the JDBC jar file) should go inside the folder `CATALINA_HOME/webapps/bookstore/WEB-INF/lib`.

3 Project Description

The web application is an online book store application. This section describes the high level functionality 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 Domain Description

Let us start with the domain knowledge that your database must model.

- **Books.** A book is written by one or more authors and, in addition, it has the following characteristics:
  1. **ISBN number**. This number uniquely identifies that book and is a 13 digit number. Note that it serves just as a unique ID for the book, you will not need to use it in arithmetic operations.
  2. **Title.** The title of the book. This can be a string composed of many words (e.g., 'The Hitchhiker's Guide to the Galaxy')
  3. **Description.** A two-three sentences long description of the book.
  4. **Image.** This is an image of the book's front cover, which must be shown to the users together with the other information about the book. Note that you do not need to store the actual image inside the database. You can store a JPEG file with the image on the disk and just store the image's URL in the database.
- **Authors**. An author can write one or more books and, in addition, has the following characteristics:
  1. **ID**. This is a 5 digit number that uniquely identifies an author. This number just serves as an identifier.

2. **Name**. The name of the author. It includes both first and last name (e.g., Douglas Adams).

- **Users**. These are the users of your website.
  1. **<u>user_id</u>**. This is a number that uniquely identifies a user.
  2. **username**. This is the username that is used to log in to the website.
  3. **password**. This is the password that is used to log in to the website.
  4. **Name**. This is the name of the user. It includes first and last name.
  5. **address**. This is the address of the user (assume billing and shipping are the same) .
  6. **Credit Card**. This is a 16 digit credit card number associated with the user.
- **Orders**. These are the orders that users make to buy books. An order has the following characteristics.
  1. **<u>order_id</u>**. This is a unique identifier of the order.
  2. **ISBN**. This identifies the book ordered by the user.
  3. **user_id**. This identifies the user who made the order.
  4. **quantity**. This is the number of books ordered by a user.
- **Order History**. The database must maintain the history of all the orders placed by all users. For every order, it should record the date and time of the order as well as the user who made the order.

3.2 Functionality

This subsection contains a description of the main pages and functionality of your web application. 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.

Login.jsp.
Function: Log users in. A user MUST log in using the page `Login.jsp` before submitting orders. This page shows a form to the user with two text input fields, `User Name` and `Password` and a submit button.

After the user clicks on the submit button, an HTTP request is sent to `Login.jsp`. The java code in `Login.jsp` MUST retrieve the username and password from the request and send a query to the database to authenticate the user. In the event of a successful login, the user MUST be redirected to the page `ShowBooks.jsp`. In case of a failed log in, the user MUST remain in the page `Login.jsp`.

ShowBooks.jsp.
Function: Show users list of books to choose from.
This page is responsible for showing a list of books to the user. The information about each book (title, author, front cover image) MUST be retrieved from the database and shown to the user in an HTML table. This HTML table MUST contain one row per book. In addition, for every book, you MUST display a text input field where the user can insert the quantity, and a submit button to submit the order. When the user inserts a quantity and clicks on the submit button, an order should be inserted in the `orders` table. After a successful insertion, the user MUST be shown a message of a successfully placed order (e.g., by redirecting to the ShoppingCart.jsp page, or by redirecting to a different page.). The user MUST be able to order more than one books during the same session. This implies that if you redirect the user to a different page, you must place a link back to the `ShowBooks.jsp` page.

ShoppingCart.jsp.
Function: Show overview of the orders to the user. This page is responsible for showing a list of the orders made by the user in this session only. It MUST show the list of books that the user has ordered, the quantity, together with their prices and a total price. The page MUST contain a `checkout` button. When the user clicks on this button, the order is finalized.

Other Issues. To get started, you can insert manually some users, books, and authors to the database. You are free to choose usernames, passwords, and books. I did not include this as part of the functionality to keep things relatively simple for this first part.

An Apache Tomcat installation will be made available on `calculus` under `/usr/apache`. You can test your application there before submitting the final version. In order to avoid confusion: 1) create a folder named after your netid inside the `webapps` folder, 2) make that folder readable and writable and executable only by yourself, and 3) place your bookstore code there. For instance, since my netid is *rgjome1*, my application would be under `webapps/rgjome1/bookstore` and the folder `webapps/rgjome1` would be readable, writable, and executable only by me.

4 Deliverables

1. **Code**. Your code (including libraries, images of the front covers, and so on) must be placed in a folder named `bookstore` in your home directory on `calculus`. In addition, you must dump your database to a SQL dump file named `bookstore.sql` and place that file inside your home folder on `calculus`. See instructions on how to dump a database at `https://dev.mysql.com/doc/refman/5.7/en/mysqldump-sql-format.html`.

   We will use a script to deploy each application and the script will expect to find a folder named `bookstore` inside your home folder, as well as a file named `bookstore.sql` inside your home folder on `calculus`. So make sure to follow these instructions to the letter. You must upload the folder as a compressed file, and the database dump file on Blackboard as well.

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!!!