TU Kaiserslautern
FB Informatik, AG HIS
Prof. Dr.-Ing. Stefan Deßloch
M.Sc. Michael Hohenstein

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Worksheet 4 - DB Schema Design and Programming

"JSON API for Online Shopping"

Introduction

In this worksheet you will develop a web application for the internet book store *Larazon*. Your task is to implement the core engine of a web application for online shopping. The engine consists of a set of `servlets`, which are accessed via HTTP and deliver responses in the JSON format. Therefore, you must not implement the HTML interface, but rather a simple API that allows multiple interfaces to interact with the core application.

In this worksheet, you must design your own database schema, using the concepts you learned in the previous worksheets. Your schema must be concise and extensible, and you are free to use object-relational features, functions, views, triggers, and stored procedures if it makes the development easier and your architecture simpler and more extensible. Investing time in the schema design not only will make your job easier when implementing the servlets later on, but a good design will also earn you bonus points. Your schema must support the following information for each book:

- Book ID
- Title, e.g., 1984
- Price in EUR, e.g., 8.95
- a small cover page picture
- ISBN, e.g., 3453164210
- Title, e.g., 1984
- List of authors, e.g., George Orwell

All fields must be stored entirely in the database, including the cover picture (as a BLOB column).

The application logic must be implemented using Java EE technology, based on the JBoss Application Server. You must implement the tasks with servlets only, using pure Java code. Additional tools like JSP, JSF, or EJB are not necessary.

The exact descriptions of the response formats and request arguments for each Servlet are given in the Wiki. A sample Web project, containing all servlet classes with empty methods is also available in the Wiki.

Requirements

- Use the Jackson library for producing the JSON results.
- Follow the guidelines for handling exceptions and closing connections properly given in the Wiki. Failing to do so will likely cause unexpected errors.
- Use the session and application scopes provided by the application server to maintain any application state. Do not store any information on global variables and keep your servlets stateless.
- Implement all query logic in SQL, using **prepared** statements. Fetching all rows from the DB and performing tasks filtering or aggregation on the Java side, for instance, is extremely bad coding practice.
- Use proper indentation and meaningful comments. This is even more important in the case of Java code than in SQL.

- Use proper bean classes to represent and perform basic operations on application objects like books and shopping carts.
- Pack as much reusable functionality as possible into utility classes. The servlet code should be as small as possible and avoid any redundancy (i.e., the same code in multiple servlets).
- Failing to adhere to any of the requirements will incur in **point deductions**, even if your solutions produce the correct result.

Exercises

Exercise 1: Search

The first functionality to be implemented is the ability to search the book catalog.

1.1. Servlet `SearchBooks`

The servlet receives a list of keywords, each as an argument name. A book qualifies for the result list if all keywords are found in the content of the *title* and *author* fields. If no keywords are provided, the servlet must simply return a list of all books. It is also possible that a search key is only a part of a word. You do not need to distinguish between lower and upper-case characters (case-insensitive search).

1.2. Servlet `LoadCover`

The cover image can be retrieved for each book using a separate servlet. It takes a book ID as argument and returns the cover image directly as binary JPEG data. This would allow the image to be embedded, for example, in an HTML page using the `<img>` tag.

Exercise 2: Shopping

Extend your application to allow customers to place books into a *shopping cart*. The shopping cart must not be stored in the database. It must be managed entirely using *sessions*. A shopping cart is exclusive to a customer and is discarded when leaving the shop (e.g. closing the browser), or when purchase is finalized.

2.1. Servlet `AddToCart`

Using the book ID as argument, this servlet allows the customer to add books to his shopping cart. When it is invoked for the same book ID multiple times, the quantity field of each entry must be incremented.

2.2. Servlet `ShowCart`

This servlet produces a summary of the books currently in the customer's shopping cart, including the book information and their quantity (number of identical books in the cart). Additionally, it shows the total price for the whole shopping cart.

2.3. Servlet `OrderCart`

This servlet allows a customer to purchase all the items on his shopping cart by providing his address and payment information. After the purchase is completed, a new session must be generated (using a new session ID). The details of the order must be stored in the database.

2.4. Servlet `LoadOrder`

This servlet displays the details of a single order stored in the system, i.e., all successfully purchased shopping carts. It should provide, the shopping cart summary (as returned by `ShowCart`) together with the customer information (the arguments of `OrderCart`), the session ID, and the timestamp when the purchase was concluded.

2.5. Servlet `ClearOrders`

This servlet is used by the evaluation system to reset the tables to an empty state, before inserting the sample data used for validation. Your task is to simply remove all orders in the system. Please note that the evaluation will fail if the tables are not properly emptied, even when all the other servlets are correct.

Exercise 3: Logging

Develop a tracking system that records customer behavior within your online shop. The application should track every request like searching, adding books to the shopping cart, or buying books. The collected information must be stored within *log tables* in the database. Remember that the system does not have any log-in functionality, but actions must be assigned to distinct customers using the session ID.

3.1. Servlet `TrackingFilter`

This servlet is not invoked explicitly, but it intercepts every request to allow any pre-processing step to be implemented. You may use it to collect log data for the relevant events.

3.2. Servlet `ShowTrackingData`

This servlet displays all the currently available tracking information. The returned data must be a list of *raw events*, ordered by the timestamp recorded in the log tables. The format of each event record depends on the type of action that was logged (i.e., search, addition to cart, purchase, etc.).

In a real scenario, this functionality should only be available to authenticated system managers, but you do not have to implement this functionality. The same holds for the following exercises.

Exercise 4: Statistics

In order to support business decisions, the "raw" log data must be aggregated and loaded into *summary tables* for the analysis of customer behavior. This load operation is invoked using a servlet. When the operation finishes, the log tables must be emptied. This will be verified by invoking `ShowTrackingData`. The load procedure must update the summary information in an incremental manner, meaning that information from previous loads must not be lost.

4.1. Servlet `CalcStats`

This servlet is invoked to compute statistics and store them in the summary tables, flushing the log tables at the end. Because this is a critical operation, you should try to implement it in a reasonably efficient way.

4.2. Servlet `ShowStats`

The summary table should answer the following questions:

1. How often (in the average) does a customer search for books during a shopping tour?
2. What is the average number of results delivered by a book search?
3. What is the average number of keywords in a search?
4. What is the probability of a purchase?
5. What is the probability of a purchase when a user has added an item to the shopping cart?
6. What is the average total price of an order?
7. Top 5 list of best sellers.
8. Top 5 shopping carts by total price.
9. Top 5 buyers (sum of all purchases) identified by their e-mail.

10. Top 5 keywords used in searches.

For the values that result from averages, there may be slight variations depending on the datatype used for numbers (decimal, float, etc.) and on the order of operations performed. Therefore, our evaluation will consider results within a 1 % error margin.

4.3. Servlet `ClearStats`

This servlet clears all data from the tracking and summary tables, resetting the database to an empty state.

Additional Requirements and Hints

- Use the Eclipse IDE (see Wiki page of worksheet 4 for instructions).
- Make sure that your responses follow the exact specification given in the Servlet descriptions in the Wiki.
- Instead of using the interface `DriverManager`, you have to establish the JDBC connection via a *DataSource* provided by the application server.
- The JBoss instance used for evaluation is not accessible from outside our internal network. For development and testing, you should run an instance on your local machine or in the VM (see Wiki for instructions).

Submission

Submission deadline is Sunday, July 21, 2019 at 23:59:59.
Lots of fun and success!