Faculty of Engineering and Architecture

Department of Computer Science

Course: Advanced Java

Semester: Fall 2022

Assignment 2

Title: Book Review Application

Prepared by: Arjan Dumani & Endi Triço

Instructor: Elton Ballhysa, PhD

Date: 28 February 2023

Table of Contents

1. Problem Statement

The assignment requires the creation of a prototype for a book review web application, and this application includes three types of users, which are

1. Anonymous User → Can search for books and view their details.
2. Standard User → Can add reviews for books.
3. Admin User → Can perform CRUD operations on books and users.

The header of all views will contain the name and surname of the logged-in user, along with links to the user's profile and logout. The index page will display the 5 most recently added books and the 5 top-rated books. Users can search for specific books based on title, author, genre, and average review range. Each book will be displayed with a thumbnail image, title, and average review rating, which will link to the book detail page.

The detail page will display all book details, including the average review rating, and a list of related reviews. If the logged-in user has already reviewed the book, they can remove the review; otherwise, they can add a review. Reviewers can rate a book from 1 to 5 stars and leave a comment up to 255 characters long.

An API web service will also be provided, which will list book content based on parameters such as title, author, and category. Book details will include id, title, author, publishing house, publication year, genre, and date added to the database. Reviewer details will include id, name, surname, email, password, and user type.

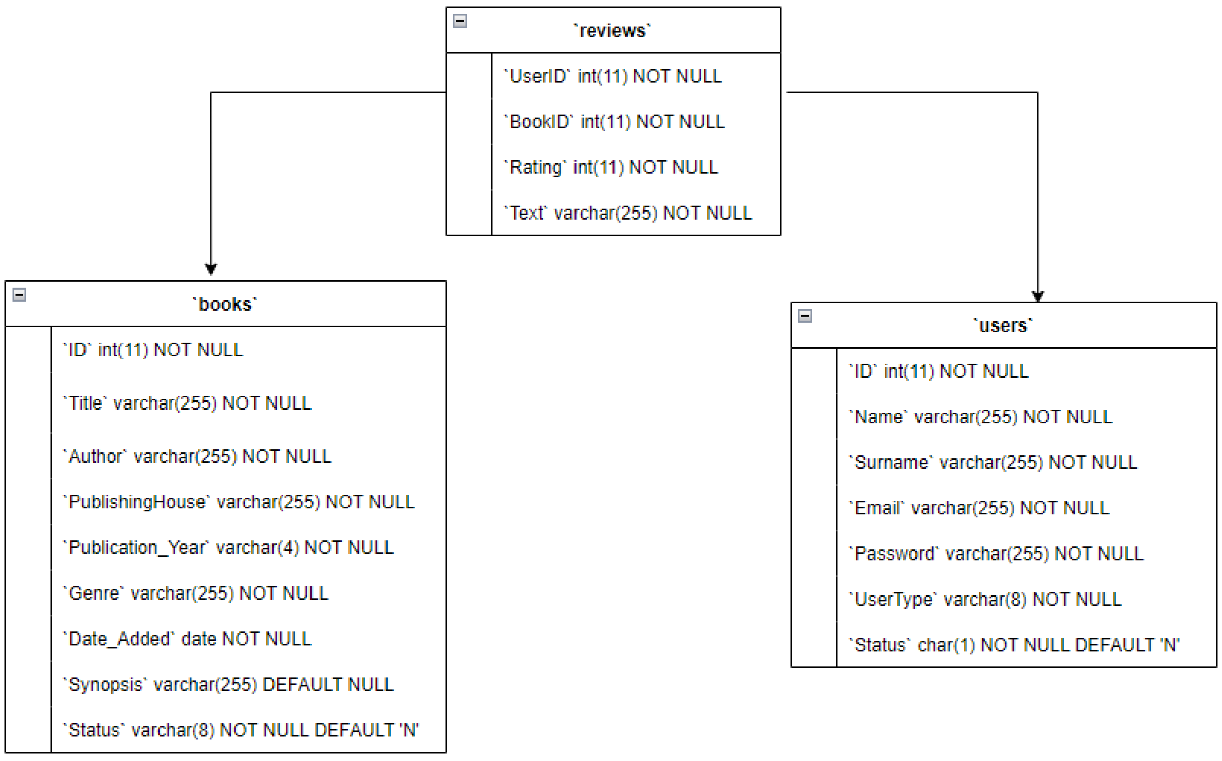
The project must use JSF technology for the web tier, JPA for communication with the information tier, and MySQL Community Edition. As an application server Tomcat is used. The database must be seeded with approximately 5 users, 50 books, and 100 comments.

2. Database Analysis

This assignment required the usage of only one type of the database, which was MySQL. However, it does not mean that it has negative effects, since it comes with some benefits. MySQL is a popular open-source relational database management system that has several advantages when used in Java EE applications. Some of the basic benefits for the Java developers are:

1. Ease of use: MySQL is a relational database, that it does not contain any difficulties in installing, setting up, and configuring it, and as a result, it makes a good database for Java developers, but not only for them.
2. High Performance: Even though, this is not an application that will be used in the real time, it provides a higher level of performance, by dealing with a high traffic load. This makes it ideal for enterprise applications that require real-time data processing.
3. Scalability: MySQL has the ability to deal with large volume of data and complex data structures, so it will offer scalability, and it will be an ideal choice for Jaca EE application that require efficient data storage.
4. Security: MySQL has several security features that make it a safe choice for Java developers. These features include password authentication, access control, and encryption of data in transit and at rest.
5. Open-source: MySQL is an open-source database management system, which means that it is free to use and can be customized according to specific requirements. This makes it an ideal choice for Java developers who want to build cost-effective applications.

In this case, we have included three tables, which are books, users, reviews. The book table contains these properties, which are ID that corresponds to the ID of the book, which implements auto increment command, and it is the primary key of the table. The other properties are the Title, Author, PublishingHouse, Publication_Year, Genre, Date_Added, Synopsis, and Status. The second table is the users table, which has as primary key the ID of the users, which is auto incremented. The other properties are the Name, Surname, Email, Password, UserType, Status. The last table is the review table that does not have a common primary key but the composition of the UserID, and BookID. This are the foreign key for the books and users' tables. The other properties added to this table are the Rating (from 1 to 5) and the Text.


`reviews`
`UserID` int(11) NOT NULL
`BookID` int(11) NOT NULL
`Rating` int(11) NOT NULL
`Text` varchar(255) NOT NULL
`books`
`ID` int(11) NOT NULL
`Title` varchar(255) NOT NULL
`Author` varchar(255) NOT NULL
`PublishingHouse` varchar(255) NOT NULL
`Publication_Year` varchar(4) NOT NULL
`Genre` varchar(255) NOT NULL
`Date_Added` date NOT NULL
`Synopsis` varchar(255) DEFAULT NULL
`Status` varchar(8) NOT NULL DEFAULT 'N'
`users`
`ID` int(11) NOT NULL
`Name` varchar(255) NOT NULL
`Surname` varchar(255) NOT NULL
`Email` varchar(255) NOT NULL
`Password` varchar(255) NOT NULL
`UserType` varchar(8) NOT NULL
`Status` char(1) NOT NULL DEFAULT 'N'

User of our schema is 'root' and no password is given. In case these are changed, the persistance unit must be changed.

Admin Users

Endi Trico → enditrico@unyt.edu.al → admin

Arjan Dumani → arjandumani@unyt.edu.al → admin

Standard Users

Test Test → test@test.com → user

John Smith → johnsmith@gmail.com → js

Joe Doe → jdoe@gmail.com → jd

Mark Doe → markdoe@gmail.com → user

3. Entities

Entity classes are used in Java EE for object-relational mapping (ORM). ORM is a programming technique that enables the developer to map Java classes to database tables and vice versa. By doing so, the developer can interact with the database in a more object-oriented way, rather than using low-level SQL queries.

Entity classes represent the data model of the application and are used to interact with the database. They define the structure of the data, including the table and column names, and the relationships between them. By using entity classes, the developer can access the database without having to write SQL queries manually, which can be error-prone and time-consuming. So, it brings efficiency, and simplicity of the interaction with database. Moreover, every table corresponds to a class, columns to instance variables, and rows to instances. In this project, these entities are located in the model directory, and every model has its primary key.

So, the entities of this project are Book, User, Review, ReviewPK. These entities are tagged with the appropriate annotation. They represent the tables in the MySQL database, and the embedded time to represent the primary key of the review table.

4. Data Access Object

The DAO makes possible to create an extra layer of abstraction so that the managed bean does not communicate directly with the database. The class has three DAO classes and one interface.

The first interface named BaseDao has four methods. The first one is the insert method, that means adding different objects of type T, and it does not return anything. Then, we have the method delete, that has only one parameter with type T, and it does not return anything. The purpose is to delete an object, such as a review added by the user, and he does not want to show it anymore. Then, we have the getAll method that does not contain any parameter, but it returns a list of type T, based on the object that will be represented, and the final method is the getById method that has as parameter K, and return an object T. This interface is implemented to all the other DAO classes.

The first class of DAO package is BookDAO that implements the BaseDAO interface, and it contains all the method signatures of its interface. As a result, these methods are elaborated based on their needs, which are insert method, to add a book and take as a parameter an object of the Book class found in model package (entity), delete method that has as a parameter an object of the book class and it removes that book. The following method is getById, which requires the id of the book, and it introduce the details of that book that corresponds to that specific id, and the last one is getAll method, which return a list of all Books. The other methods implemented here as follows, the update book, that has as the parameter an object of Book class, which will call the updateBook, in order to change its details to our database. The other methods are the getLasFive, which will give the last 5 added book to the application, and getTopFive method that will give the 5 books with the highest rating. To not explain more in details, there are also some other methods that are required for this assignment.

The second class is the ReviewDAO that implements DAO interface, and except from its method, it contains also the other methods, which are reviewExists, that will check for the existence of a user review based on its id and book id. Then, we have the getUserReviews of the user, and getBookReviews, which will show the reviews of the books.

The third class implements the interface BaseDAO, and as a result, it contains BaseDAO methods. Some other methods are the update method, which will update the user credentials, and profile properties, the second method is getUserByEmail that will retrieve a specific user only by its email that is given as the parameter. Another important method is the filterUsers based on the required properties entered in the application.

5. Managed Bean & Views

In Java EE, a managed bean is a Java object that is instantiated, configured, and managed by the container. Managed beans are used to implement the presentation layer of an application in a way that separates the business logic from the presentation logic.

One advantage of using managed beans is that they simplify the development of web applications by reducing the amount of boilerplate code that needs to be written. They also provide a way to modularize the code, making it easier to maintain and reuse. Additionally, managed beans can be easily integrated with other Java EE technologies, such as dependency injection, to further simplify the development process. They can validate data, handle events fired by a component in a page or perform processing that determines the redirection of the user to another page.

The beans classes that are used here are: BookBean, DetailsOfBooksBean, LoginBean, MainBean, MyReviewsBean, UpdateProfileBean, and UserBean. In this case we have used these beans to communicate directly with the DAO and the views, which are indexSimple, that is the first page introduced to the anonymous users, the indexAdmin.xhtml that is the first page that is introduced to the admin users, and the indexStandard.xhtml that is used for the standard users. The myReview.xhtml deals with the reviews of the users, where they can be different operations, while profile.html deals with the profile credentials/feature of the user. Moreover, as the name says, login.xhtml is the login page that the user needs to have access (standard or admin). The last one, and the first one in the alphabetic order is the bookDetails.xhtml, that deals with a variety of books' operations.

6. Bonus Point (AJAX)

In this assignment it was an easier way to implement AJAX, and one of the cases is when the index page offers the user the chance to search for a specific book by title, author, average review range, genre, or a combination of these.

The f:ajax tag is used in JSF applications to provide asynchronous behavior when a user interacts with a web page. It is used to update the content of a page without requiring a full refresh, making the application more responsive and reducing the load on the server.

When a user interacts with a component on a JSF page the f:ajax tag allows you to specify which parts of the page should be updated when the server responds to the user's action. The execute attribute of the f:ajax tag specifies which components' values should be sent to the server when the user's action is processed, and the render attribute specifies which parts of the page should be updated with the server's response.

In the code the f:ajax tag is used to provide AJAX (Asynchronous JavaScript and XML) functionality to the h:commandButton component. When the button is clicked, the execute attribute specifies the list of components to process on the server side. In this case, it includes the input fields with IDs titleInp, authorInp, maxInp, minInp, and genreInp.

The render attribute specifies the list of components to update in the browser after the AJAX call is completed. In this case, it includes the component with ID allBooksData, the component with ID searchMsg, and the component with ID booksTable. So, when the user clicks the "Search" button, the values of the input fields specified in execute are sent to the server without refreshing the page. The server then performs a search based on the input values and returns the results. The components specified in render are then updated with the new data, allowing the user to see the search results without a full page reload.

7. API

We were required to create a web service that has a single method capable of listing book content based on specified criteria. The method takes five parameters, namely book title, author, category, minimum rating range, and maximum rating range. While none of the parameters are mandatory, at least one parameter must be provided to enable the search. The method should return a list of books that match all the specified criteria. So, the technology used here is SOAP, since the assignment did not specify it. The reasons why we chose SOAP over REST, is that we find it more flexible to use it, and SOAP makes it easier to understand the data being sent and received and ensures that both sides are using the same format. Another advantage is that SOAP includes built-in support for reliable messaging, which means that if a message is lost or corrupted during transmission, it can be retransmitted until it is successfully delivered. It provides a more structured and standardized approach to building web services, which provides a more consistent and structured framework for them to work with. As a result, it is better suited for handling complex operations.