Assignment
Access and expose a database.

Data access with JDBC and Thymeleaf

Build a Spring Boot application in Java. Follow the guidelines given below, feel free to expand on the functionality. It must meet the minimum requirements prescribed.

NOTE: This assignment is to be completed in groups of 2. In addition to this, you should complete this assignment using Pair Programming.

1) Set up the development environment.

Make sure you have installed at least the following tools:

- IntelliJ with Java 16.

2) Use plain Java to create a Spring Boot Web API, and use Thymeleaf to create a view with the following minimum requirements (See Appendix A-B for details):

a) Access the Chinook SQL Lite database through JDBC. This should be done according to Appendix A.
b) A Thymeleaf view to show database data according to Appendix B.
c) This should all be in one project.
d) The application must be published as a Docker container on Heroku.

3) Submit

a) Create a GitLab repository containing all your code.
b) Include a well formatted README and appropriate commit messages.
c) The repository must be either public, or I am added as a Maintainer (Ask lecturer for Gitlab handle).
d) Submit only the link to your GitLab repository (not the “clone with SSH”).
e) Only one person from each group needs to submit but add the names of both group members in the submission.

Appendix A: Reading data with JDBC

1) Introduction and overview

Some hotshot media mogul has heard of your newly acquired skills in Java. They have contracted you and a friend to stride on the edge of copyright glory and start re-making iTunes, but under a different name. They have spoken to lawyers and are certain a working prototype should not cause any problems and ensured that you will be safe. The lawyer they use is the same that Epic has been using, so they are familiar with Apple.

You have been provided with a Chinook database, ERDs can be found here.

Chinook models the iTunes database of customers purchasing songs. You are to create a Spring Boot Web API, add the SQL Lite JDBC dependency to your project, and create classes to interact with the database.

2) Endpoint requirements

These endpoints must be on a /api/ sub directory in your applications structure. Meaning, "/" and "/search?term=foo" are for the Thymeleaf pages and "/api/bar" is for the REST endpoints.

The endpoints should be designed with best practices in mind. The endpoints should be named appropriately; remember, nouns not verbs.

Provide a collection of API calls made in Postman to test the endpoints (done by creating a collection and exporting it as JSON).

HINT: Don't be afraid to go deeper with endpoint naming hierarchies, it's perfectly fine to have an endpoint like: */api/customers/:customerId/popular/genre*

3) Customer requirements

For customers in the database, the following functionality should be catered for:

1. Read all the customers in the database, this should display their: Id, first name, last name, country, postal code, phone number and email.
2. Read a specific customer from the database (by Id), should display everything listed in the above point.
3. Read a specific customer by name. HINT: LIKE keyword can help for partial matches.
4. Return a page of customers from the database. This should take in *limit* and *offset* as parameters and make use of the SQL keywords to get a subset of the customer data. The customer model from above should be reused.
5. Add a new customer to the database. You also need to add only the fields listed above (our customer object)
6. Update an existing customer.
7. Return the number of customers in each country, ordered descending (high to low). i.e. USA: 13, …
8. Customers who are the highest spenders (total in invoice table is the largest), ordered descending.
9. For a given customer, their most popular genre (in the case of a tie, display both). Most popular in this context means the genre that corresponds to the most tracks from invoices associated to that customer.

HINT: You should be looking to use some of the following SQL clauses: JOINs, ORDERBY, GROUPBY, and MAX.

NOTE: You should create a class (model) for each data structure you intend to use. Do not return a formatted string. This is a minimum of: Customer, CustomerCountry, CustomerSpender, CustomerGenre. Place these classes in a Models folder.

4) Data access classes

You are encouraged to implement the repository pattern in this assignment. The version of the pattern to implement is up to you. This is not a requirement, but is little extra work that prepares you for structures we will see in the future.

NOTE: If you chose to make a repository, consult the provided material for examples of the Repository pattern. You can place all repository-related classes in a Data Access folder.

Appendix B: Using Thymeleaf to display data.

1) Introduction and overview

You are to add Thymeleaf to your project and create several views:

a) The home page view, showing the 5 random artists, 5 random songs, and 5 random genres. This home page contains a search bar which is used to search for tracks. The search bar should not be empty, meaning you can't have an empty search criterion.
b) The search results page will show the query the user has made, i.e. Search results for "Never gonna give you up". Underneath this, the results will be shown for the search. The search results should show a row where the track name, artist, album, and genre are shown. The search should also be case insensitive.

HINT: Use @Controller for the controllers that are used with Thymeleaf and @RestController for the API endpoints.