Project Documentation

Project Overview

This project is designed to enhance familiarity with backend development in Java, particularly focusing on CRUD (Create, Read, Update, Delete) operations using Spring Data and a MySQL database. The project does not aim to solve any real-life problems but serves as a learning tool to understand the basic functionalities of a backend application.

Key Features

- **MySQL Database**: The application utilizes a MySQL database to manage reports.
- **Entity Classes**: The core entities in the application are `Report` and `Laborant`, which represent the main data structures.

Report Class

The Report class represents a medical report and includes the following fields:

public class Report:

- public Long id: The id value of a report
- public String patientFirstName
- public String patientLastName
- public String patientId
- public Date date
- public String diagnosisHeader
- public String diagnosis
- public Laborant laborant
- public byte[] image

Each report is linked to a Laborant, which means that the application must also support CRUD operations for laborants.

Laborant Class

The Laborant class represents the medical staff associated with each report and includes the following fields:

public class Laborant:

- public Long id
- public String firstName
- public String laborantId
- public String lastName
- private List<Report> reports = new ArrayList<>()

Note: The Project does not have a Service layer due to its simplicity. Instead controller layer handles both of them.

Development Environment Setup

To set up the development environment for this project, follow the steps below:

1. Create a MySQL JDBC Data Connection

First, ensure you have a MySQL server running. Create a new database for this project. Use a MySQL client or command line to execute the following command:

SQL: *CREATE DATABASE your_database_name;*

2. Specify Database Connection in Spring Boot:

You link your Java project to the database through the `application.properties` (or `application.yml`) file in your Spring Boot project. In this file, you'll provide the MySQL database URL, username, and password. For example:

```
spring.datasource.url=jdbc:mysql://localhost:3306/your_database_name
spring.datasource.username=your_mysql_username
spring.datasource.password=your_mysql_password
```

- **`spring.datasource.url`** specifies the database URL (which includes the `your_database_name` you created).
- **`spring.datasource.username`** and **`spring.datasource.password`** provide your MySQL credentials to authenticate and connect.
- **`spring.jpa.hibernate.ddl-auto=update`** tells Hibernate to automatically generate or update the database schema based on your entity classes.

After that run the code once to create datatables

3. Update the Image Column Type

In your database schema, update the image column in the Report table from BLOB (or Lob) to LONGBLOB to accommodate larger image files. You can execute a command similar to the following:

Note: Your table name will probably be {Your_Database_Name}.Report

SQL: *ALTER TABLE {your_table_name} MODIFY image LONGBLOB;*

4. Configure MySQL Connection

When running the application, you must provide the following connection parameters:

- MySQL URL
- MySQL username
- MySQL password

You can pass these parameters via the command line when starting the application.

5. Open the Project Using Command Line

1. **Open Command Prompt**: Press Win + R, type cmd, and hit Enter.
2. **Navigate to Project Directory**: Change to the directory where your project is located using cd command
3. **Run the Application**: Use the following command to run the project, replacing the placeholders with your actual MySQL connection details

Bash: java -jar target/user-0.0.1-SNAPSHOT.jar --
spring.datasource.url=jdbc:mysql://localhost:3306/{your_database_name} --
spring.datasource.username={your_username} --
spring.datasource.password={your_password}

Note

This project follows standard Maven project conventions. Ensure that you have Maven installed on your system to build and run the application.

Usage Instructions

Once the application is up and running, you can perform various operations on Laborants and Reports using *curl* commands. These operations can also be tested in a web browser for GET requests. {} indicates variables and will be changed depending on the user.

Laborant Operations

1. **To see all Laborants**:

   ```
   curl -X GET http://localhost:8080/Laborants
   ```

2. **To add a Laborant**:

   ```
   curl -X POST http://localhost:8080/Laborants ^
   -H "Content-Type: application/json" ^
   -d "{\"firstName\":\"{name}\", \"lastName\":\"{lastname} \",
   \"laborantId\":\"{id}\"}"
   ```

3. **To delete a Laborant**:

   ```
   curl -X DELETE http://localhost:8080/Laborants/delete/{id}
   ```

Report Operations

1. **To see all Reports**:

   ```
   curl -X GET http://localhost:8080/Reports
   ```

2. **To see a particular Report**:

```
curl -X GET http://localhost:8080/Reports/{id}
```

3. **To delete a Report**:

```
curl -X DELETE http://localhost:8080/Reports/delete/{id}
```

4. **To add a Report**:

```
curl -X POST http://localhost:8080/Reports ^
-H "Content-Type: multipart/form-data" ^
-F "patientFirstName={name}" ^
-F "patientLastName={surname}" ^
-F "patientId={patientId}" ^
-F "date={date}" ^
-F "diagnosisHeader={diagnosis}" ^
-F "diagnosis={diagnosis}" ^
-F "laborantId={id value of the laborant}" ^
-F "image=@C:/Users/DELL/Desktop/Java/User/Images/1.jpg"
```

Note: The id value of the laborant is not the value in laborantId but the automatic value assigned to it. Also the path to the image will be changed depending on the image you want to put in report

5. **To update a Report**:

```
curl -X PUT http://localhost:8080/Reports/update/103 ^
-H "Content-Type: multipart/form-data" ^
-F "patientFirstName=Jack" ^
-F "patientLastName=Sparrow" ^
-F "patientId=12345" ^
-F "date=01-09-2024" ^
-F "diagnosisHeader=Updated Diagnosis Header" ^
-F "diagnosis=Updated Diagnosis" ^
-F "laborantId=1" ^
-F "image=@C:/Users/DELL/Desktop/Java/User/Images/1.jpg"
```

Note: The path to the image will be changed depending on the image you want to put in report

6. **To see Reports sorted based on date**:

```
curl -X GET http://localhost:8080/Reports/sorted
```

7. **To see Reports in reverse sorted order**:

```
curl -X GET http://localhost:8080/Reports/reverseSorted
```

8. **To retrieve images**:

```
curl -X GET http://localhost:8080/Reports/Image/{id} --output
image.jpg
```

MySQL Command

If you did not modify the image column in the database while setting the environment, use the following command:

```
mysql -u root –p
{password}
ALTER TABLE abc.report MODIFY COLUMN image LONGBLOB;
```