Music Library v3

Time to stop using a text file for the Music Library instead of a real database.

Now everything's implemented, apart from the database connection, and the repository class.

Your job is to:

- define the datasource in `application.properties` or `application.yml` (whichever you prefer),
- set the server port to `8091` in `application.properties` or `application.yml`
- properly annotate the `LibraryRepository` class as a Spring Repository,
- autowire the `jdbcTemplate` object in the Repository class,
- and fill in the missing pieces in some of its methods.

You should not modify any of the other files. You're free to do so if you want to experiment, but this task is possible to complete by only working on `LibraryRepository` and `application.yml/application.properties`.

Note: the right dependencies are already set in the pom.

Schema

The database contains two tables with the following schema:

Table `artists`:

id (PK)	name
SERIAL	TEXT

Table `tracks`:

id (PK)	name	year	artist_id (FK REFERENCES (artists) id)
SERIAL	TEXT	INT	INT

Dockerized database server

How to create a container serving the database

To create and start a Docker container serving the database on localhost:5431, execute one of the following scripts, depending on your OS: `dockerize_win.sh` (Windows, from Git Bash) or `dockerize_mac.sh` (macOS or Linux, from Terminal).

To run the script, open Git Bash or Terminal from ***this*** folder (the *project root*) and run the following command:

On Windows:

```
./dockerize_win.sh
```

On macOS/Linux:

```
./dockerize_mac.sh
```

Container specs

The Docker container will have the name `postgresPA5`, and the port mapping will be `5432:5432`.

The database name will be `music-library`.

The db username will be `jensenyh`.

The db password will be `losenord`.

That means that the **url of the database** will be:

```
jdbc:postgresql://localhost:5432/music-library
```

(bear this in mind when setting up the datasource in `application.yml/application.properties`.)

Running that script will also initialize the tables and create some artists and tracks. The SQL script that does that is located in `/db/init.sql`, if you're curious.

Note

If you're having trouble with the script and you know how to run a Docker container with the specs above, feel free to do so. However, the script also mounts some volumes that

make sure the db is initialized with tables and some values, and that the data is persisted in `db/data/` even if you delete the container.

Postman collection

This time again I've included my (slightly updated) Postman collection of requests so that you can test your API.