[Nom de la société]

City Library Management System

Project 3

ANDRIANI RACIC, Monica Vita
27/12/2018

Table of Contents

1. Introduction

1. General Context

Project 3 has a goal to create a library management system that can be accessible by all member or users of the e-library. This system includes :

- A web site designed with responsive web design (RWD). The user can perform different things such as :
  - Find different works and their available copies
  - Borrow different books
  - Search books by author's name
  - Check their borrowing status and period
  - Extend their borrowing period
  - Return borrowed books
- A batch that is scheduled to run and send automatically email. This email will be sent to all users who have not returned their borrowed books.

Everybody in general can browse book, find works, use the 'find a book by author' search part and access home page.

To borrow a book and access the profile page, a registered member must log in.

After member login successfully, they can access the profile page, consult their borrowing list, extend and or return their loan, also make a new loan.

A member can only extend once their borrowing period and the borrowing period will be extended for another 4weeks.

2. Technologies

Technologies used in general to develop this application are :

- Apache maven 3.6.0
- Apache Tomcat 9
- MySQL 8.0.13
- Spring Framework 5.1.2 RELEASE
- Spring Data 2.1.2 RELEASE
- Spring MVC 5.1.2 RELEASE
- Hibernate 5.1.0.Final

By using Apache Maven, we decided to divide our Maven project into multiple modules

- library-batch : containing batch to send automatically email (in construction)
- library-business : containing services / business logic

- library-consumer : containing repository to connect to database
- library-model : containing different entities
- library-webapp : containing the view and controllers
- library-webservice : containing webservice (in construction)

2. Configuration and Deployment

1. Database MySQL(8.0.13)

- Install MySQL
- Execute the script library.sql at the server (you can use phpmyadmin or mysql workbench) to create user, database, tables and to insert data into the database.
-
- Create 6 tables (information can be found in library.sql file)
- Enter the data for all the tables using information from library.sql

2. Resources images, css and js

- Install and start Apache Server Local
- Download the resources folder to get all the images, css and js from https://github.com/movitarac/project3_resource
- Put them in Apache server - Document Root, inside a folder called 'resources', this folder contains 2 folders, 'assets' for images and 'style' for css + js. To call an image, for example, we enter the url http://localhost:80/resources/assets/1.jpeg . In database, all the images are already pointed to these urls (attribute called 'imageUrl' in 'Work' table).
- Change the configuration in library-webapp module by going to library-servlet.xml and in bean section 'dataSource, change :
    - Values for property name "username" by your database username and for "password" by your database password
    - ?serverTimezone=UTC can be added after value for property name 'url' "jdbc:mysql://localhost:3306/citylibrary?serverTimezone=UTC" in case of problem with timezone.

3. In IDE (Intellij or Eclipse)

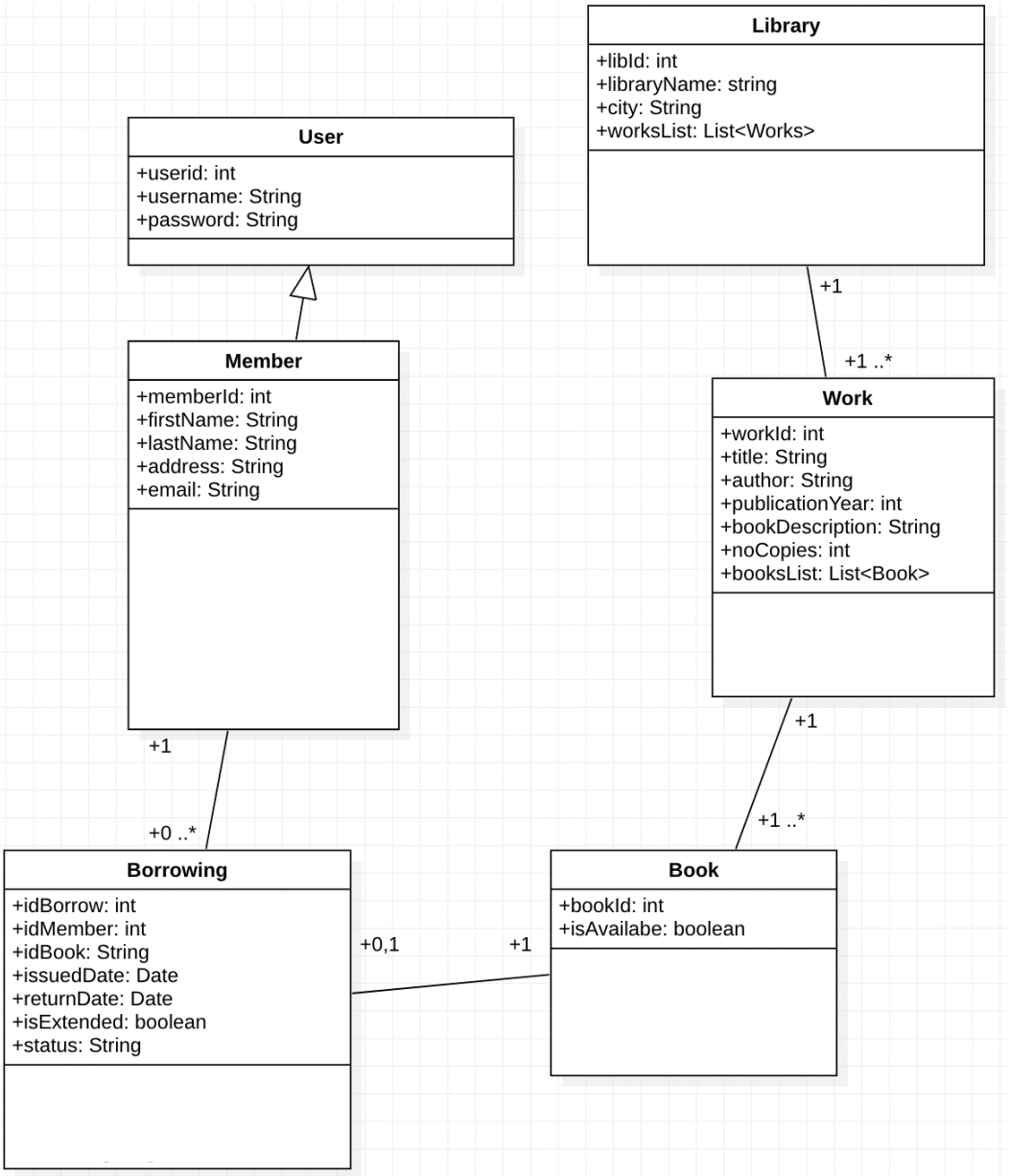
- Unzip library.zip
- Import the project library as a Maven Project in your chosen IDE
- Go to spring configuration file found in ./library/library-webapp/src/main/webapp/WEB-INF/library-servlet.xml, and in bean section 'dataSource, change :
    - Values for property name "username" by your database username and for "password" by your database password
    - ?serverTimezone=UTC can be added after value for property name 'url' "jdbc:mysql://localhost:3306/citylibrary?serverTimezone=UTC" in case of problem with timezone.
- Build the parent project (maven install)
- Deploy the library-webap.war in Tomcat 9

- In Intellij : Run – Edit Configurations – click + – Tomcat Server Local – Deployment – click + Artifact library-webapp-war – delete all text found in Application context – Apply OK – Run. The application will appear in http://localhost:8080/
- In Eclipse :

- For the moment the web service is still on progress. To make the web application part works, the web application is momentarily connected to the database (later version, only the web service will be connected to the database).

3. Application web

1. Class diagram



A member is inherited a user. An e-library has several different works, while for each work, it has different copies that can be borrowed by a member. A borrowing relates between a book (or a copy) and a member. A borrowing contains only a book.

4. Batch

Batch is still on progress.

For this part, the application scans all unreturned books (book's availability = false) or all borrowing with status 'ongoing' and 'extended'. From then, it compares the return date and today. If the return date is before today, the application gathers members' information and send them a reminding email related to their borrowing period.

5. Web service

This part is still on progress and construction.

We are going to create a module for webservice and applied a bottom-up method.

- In pom.xml , add maven plugin to generate web service; using ws-gen ->
- Create java classes corresponding to the existing services located in library-business .
    - Example : WorkWS class -> correspond to WorkService class in library business, etc.
- Add annotation @WebService for the class , @WebMethod for different methods present in service also add @Autowired to do spring IOC to inject service class

```
@WebService(serviceName="WorkService")

public class WorkWS{


        @Autowired
        WorkService workService;

        @WebMethod
        public Work searchById(Integer id) {
        return workService.findById(id);
        }
        ……
}
```

- Continue…