City Library Management System

Project 3

ANDRIANI RACIC, Monica Vita
04/01/2019

Table of Contents

1. Introduction 2
   1) General Context 2
   2) Technologies 2
2. Configuration and Deployment 3
   1) Database MySQL(8.0.13) 3
   2) Resources images, css and js 3
   3) In IDE (Intellij or Eclipse) 3
   4) Batch 3
3. Web Application 4
   1) Class diagram 4
4. Web Service 5
5. Batch 5

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
- JAX-WS
- Quartz 2 Scheduler

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

• library-batch	: containing batch to send automatically email
• library-business	: containing services / business logic
• library-consumer	: containing repository to connect to database
• library-client	: containing all generated classes from web services
• library-model	: containing different entities

- library-webapp :containing a package for the view + controllers (web application) also a package containing webservice. This web application doesn't directly call the business module.

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.

2) Resources images, css and js

- Install and start Apache Server Local
- Download the resources folder to get all 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' . This url is one of an attribute called 'imageUrl' in 'Work' table.

3) In IDE (Intellij or Eclipse)

- Unzip library.zip
- Import the project library in your chosen IDE (Intellij or Eclipse) as a Maven Project
- 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)
  1. in Intellij, Run – Edit Configurations – click + – Tomcat Server Local – Deployment – click + Artifact library-webapp-war – write /library-webapp in Application context – Apply OK – Run
  2. in Eclipse, Run on server - Select Tomcat v9.0 Server- Click next - Add library-webapp - Click finish - Run the server
  3. In http://localhost:8080/library-webapp the application will appear (same url for both IDE)

4) Batch

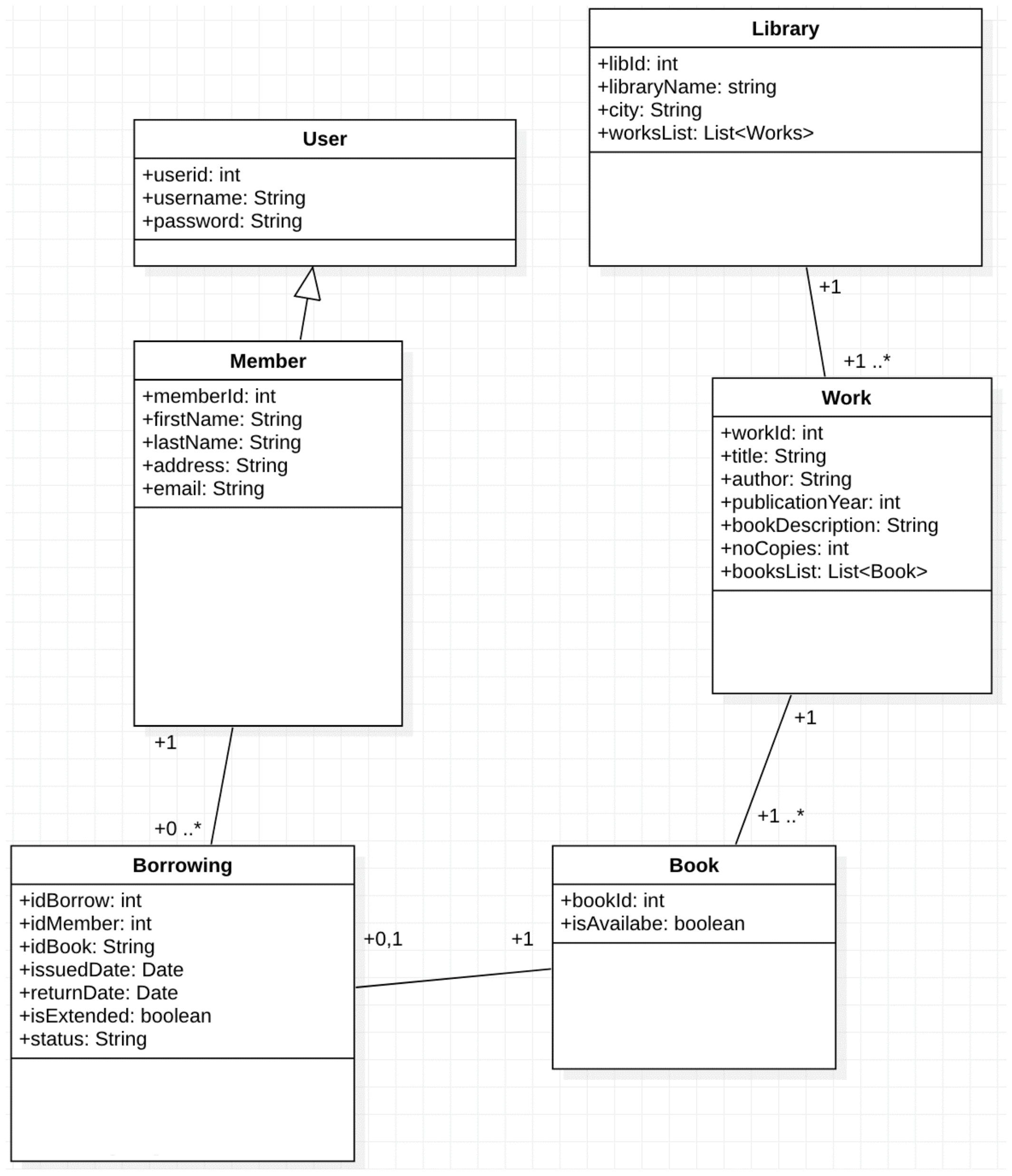
- In Linux and MacOs
  - i. Put the jar file and script shell at the $HOME (example /Users/<currentusername>)
  - ii. Run the send.sh file in terminal ./send.sh
- In Windows

i. Put the jar file at the %homepath% (example \Users\<currentusername>)
ii. Run the send.bat

- In general we can launch the jar by executing
java -jar library-batch-1.0-SNAPSHOT-jar-with-dependencies.jar

3. Web Application

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. Web Service

Inside library-webapp, there is a package corresponding to webservice (SOAP webservice). The webservice is connected to database and constructed with bottom up method. All the web methods found in this package call all methods found in library-business. It means the application web does not directly call the library-business.

WSDL files can be accessed after deploying the library-webapp.war in Tomcat http://localhost:8080/library-webapp/ws/workWs

5. Batch

For this part, the application calls a client of a webservice to get 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 after today, the application gathers members' information and send them a reminding email related to their borrowing period. It is scheduled to be automatically launch every 5 seconds(for test).