City Library Management System

Project 6

ANDRIANI RACIC, Monica Vita
03/04/2019

Table of Contents

1. Introduction

a) 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.

b) Technologies

Technologies used in general to develop this application are:

- Apache maven 3
- Apache Tomcat 9
- Apache Server
- MySQL 8.0.13
- Spring core Framework
- JPA
- Spring Data
- Spring MVC
- SoapUI
- Git - Github

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 manage the database
- library-client : containing all generated classes from web services (WS Client)

- library-model : containing different entities
- library-webapp : containing presentation project: view and controller (Spring MVC)
- library-webservice : containing Web service project: Expose APIs for business logic

2. Configuration and Deployment

a) Database MySQL(8.0.13)

- Install your own MySQL
- Execute the script library-database-and-user.sql at the server (you can use phpmyadmin or mysql workbench) to create user and database

b) 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 the 'resources' folder in Apache server - Document Root, 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.

c) Deployment of presentation project and webservices project from war files

- You need the Tomcat installed in your machine. Let the default port to 8080
- Download from OC library-webapp.war and library-webservice.war
- Put them to the directory <YOUR_TOMCAT_DIRECTORY>/webapps
- Start your tomcat
- Go to your database citylibrary and execute the script library-data-test.sql
- Access the app: http://localhost:8080/library-webapp
- Access the webservice wsdl http://localhost:8080/library-webservice/ws/workWs
- use the following user's information to login: willcam / 1234willow

d) In IDE (Intellij or Eclipse)

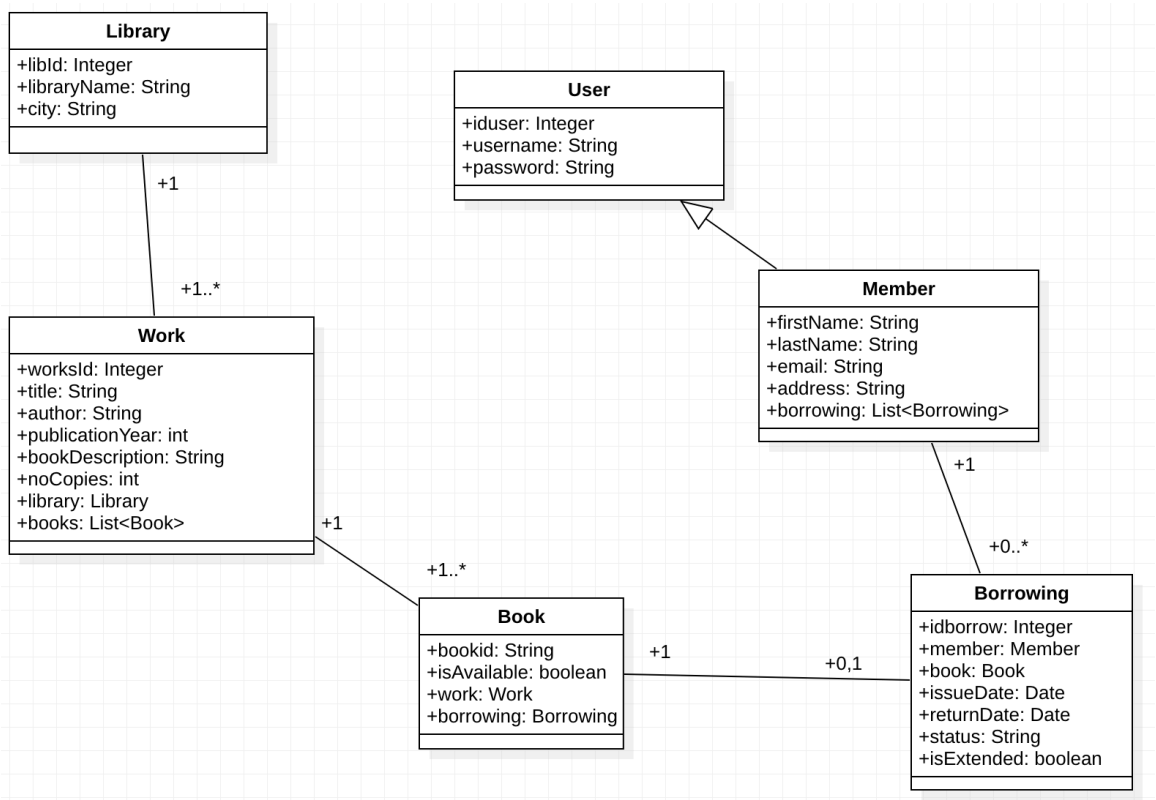
- Download sources from github (zip is better): ***https://github.com/movitarac/project6_monica***
- Unzip your download *project6_monica-master*
- Import the project *project6_monica-master/library* in your chosen IDE (Intellij or Eclipse) as a Maven Project
- Build the parent project (maven install)
    - mvn clean install (on parent project -> library)
    - To run on embedded server in Intellij: Run – Edit Configurations – click + – Tomcat Server Local – Deployment – click + Artifact library-webapp-war – write /library-webapp in Application context – Apply OK – Run
    - To run on embedded server in Eclipse: Run on server - Select Tomcat v9.0 Server- Click next - Add library-webapp - Click finish - Run the server
    - In http://localhost:8080/library-webapp the application will appear (same url for both IDE)

e) Batch

- In linux/MacOs
    - Put the jar file library-batch-1.0-SNAPSHOT-jar-with-dependencies.jar and script shell at the $HOME (example /Users/<currentusername>)
    - Go to your phpmyadmin or workbench - database citylibrary then click Member table, change the email for the first user (George Lukas) with your own email, so you can receive the email during the demonstration (send email each 10seconds)
    - Run the send.sh file in terminal ./send.sh
- In Windows
    - Put the jar file library-batch-1.0-SNAPSHOT-jar-with-dependencies.jar and the bath script at the %homepath% (example \Users\<currentusername>)
    - Go to your phpmyadmin or workbench, click Member table, change the email for the first user (George Lukas) with your own email, so you can receive the email during the demonstration (send email each 10seconds)
    - 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. Application web

a) 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.

As an enhancement, the client asks to add a 'Session' as class. Session will manage connection (duration, session) for each user. The implementation of session is carried out in business (logic) and is considered as a security layer.

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-webservice/ws/workWs

a) SoapUI

For the new enhancement of this library management system, the test can be executed by using SOAP UI. In folder 'SOAP-UI' we can find five SOAP-UI projects and a file containing properties implemented in each Test Case.

- Open SoapUI – Click File – Import project – Choose the project – In 'TestSuite' part, find 'Custom Properties' and 'Load properties from external file'
- Launch the TestSuite

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 send automatically every 10 seconds (for demonstration).