SOEN 487: Web Services and Applications

Assignment 1 on RESTful Services

Winter 2021, sections NN

January 14, 2022

Contents

1 General Information

Date posted: Thursday, January 14th, 2022.

Date due: Tuesday, February 8th, 2022, by 23:59.[1].

Weight: 6% of the overall grade.

2 Introduction

This assignment targets creating and consuming RESTful web service using java.
In this assignment you implement a small repository system using RESTful API. The repository hosts two collections: 1) Albums and 2) Artists. See section 4 for the details.

3 Ground rules

You are allowed to work on a team of 3 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. See Submission Notes for the details.
ONLY one copy of the assignment is to be submitted by the team leader. Upon submission, you must book an appointment with the marker team and demo the assignment. All members of the team must be present during the demo to receive the credit. Failure to do so may result in zero credit.
This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

[1] see submission notes

4 Overview

The repository hosts two collections:

1. Albums: each album contains a unique International Standard Recording Code (ISRC) code, a title, an optional content description, the released year, and the artist (see below).

2. Artists: An artist is uniquely identified by a nickname (a unique string identifying the artist). The artist record contains the first name and last name of the artist, as well as an optional short bio.

Notes:

- An album may only have one artist.
- The link between the album and the artist is made by the artist nickname.
- It may be possible that an artist record of an album is not stored in the artists repository. No referential integrity constraint is required.

5 Your Assignment

The assignment consists of the following components, to be implemented in separate projects:

1. The Repository Core

   A java class library containing common interfaces / data structures that later on may be shared between the client and the service (The Interface)

2. The Repository Implementation

   A java class library that implements the repository system (The Business)

3. The Repository Service

   A java web application providing the web service (The Service)

4. The Client

   A java console application that enables user to invoke the web service methods and report displays the information returned by the user.

5.1 The Repository Core

Create a thing java class library project and include common data structures (i.e. entity classes) in it. In this assignment, this core project is only referenced by the implementation (and consequently used by the service). This design will be extended in future assignment to be used on the client side. Note that no business classes are implemented in here. The project contains common interfaces and POJO data only.

5.2 The Repository Implementation

This project is the actual repository implementation. It implements standard CRUD operations on both Albums and Artists. See 5.3 for required business functionalities.
The implementation is purely in java. No references or assumptions on web technology must be made. You may design and organize the business layer, as you wish. You may use a single class to maintain the albums and artists data or use completely isolated classes.
In this assignment, using a database is not recommended. However, you are allowed to use databases. You can simply store the data collections in memory. As a result, once the service is shutdown (see 5.3), the data will be lost. Beware of concurrency.
In case you plan to use an internal [database] id, make sure this internal id is not exposed by the business layer. The business functions only expose the pieces of information as listed in section 4.

5.3 The Repository Service

Using the class library created in 5.2, implement a RESTful service to provide the following functionalities:

Artists:

- list artists (nickname + full name)
- get artist details (returns the artist full info including bio)
- add artist
- update artist

- delete artist (delete artist's record, if there are albums under the artist's name, the album records remain unchanged)

Albums:

- list albums (shows the list of albums by ISRC and title)
- get album details (returns the album info as listed in 4)
- add album (adds a new album to the collection; no artist details)
- update album info
- delete album (deletes the album only, artist details will be ignored)

Implementation Notes

1. The Album API is to be implemented using JAX-RS.
2. The Artist API is to be implemented using Servlet technology.
3. The REST API strictly returns the data in text format (`MediaType.TEXT_PLAIN`). To do so, you need to implement the `toString()` method in corresponding POJO objects in the core library (section 5.1.
4. API methods that return a collection, may return each collection item in a separate line.
5. The service layer must catch all errors, and in case existence, the error text must be returned as the result of the API call.
6. methods that do not return data (i.e. add or update methods) must return a short text indicating that the operation succeeded.

5.4 The Client

Create a Console Application that implements the REST client and lets the user to perform all four verbs on the service. The console application displays a menu to the user to receive the user's selection and repeats the process until the user wishes to quit. The menu options include all operations listed in 5.3 plus a "quit" option.

Implementation Notes

- You may use any web client library to perform API calls to the Artists service.
- Use may automatically create the client code for your calling the Albums API using the client tool generator or add a web reference in your project, depending the IDE you use.
- The console class must be implemented in a separate class and use the above two clients to perform calls. During each call, the console receives the required parameters to the API method from the user, performs the call, and reports the result as is, without any interpretation.
- Your client application must also handle client errors and in case of any, they must be reported to the console.

5.5 Other Tasks

5.5.1 Source Control

You may use GitHub or an alternative source control during development phase. Using a source control software is mandatory.

5.5.2 UML Diagram

Upon completion of your project, generate and include the "UML class / package diagram" of all implemented classes in your project.

5.5.3 Using curl

During the demo, you will run the commands in conjunction with using your client application to demonstrate the API calls.

Hint: use `curl -v` for full response view.

You may prepare a list of curl commands ahead of time prior to your demo session.

6 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. It has to be completed by ALL members of the team in one submission file.

Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`. Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above: `https://moodle.concordia.ca`
Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep a record of your submission confirmation. This is your proof of submission, which you may

need should a submission problem arises.

7 Grading Scheme

The Core Layer	5 marks
The Repository Business Layer	15 marks
The Artists Servlet	15 marks
The Albums Service	15 marks
The Client Code	10 marks
The Console Class	10 marks
Error Handling (client and server)	10 marks
Handling Concurrency	5 marks
Correct Architecture and Design	10 marks
Using curl	5 marks

Total: 100 marks.

References

1. REST with Java (JAX-RS) using Jersey:

   `https://www.vogella.com/tutorials/REST/article.html`

2. REST client tool:

   `https://www.jetbrains.com/help/go/rest-client-tool-window.html`