Metropolitan State University

Computer Sciences Department

CS 3810 - Principles of Database Systems – Spring 2021

Database Project 03

Deadline: April 22th, 11:59pm

Overview

The goal of this assignment is to finish a course enrollment application that uses Object-relational Mapping (ORM) Hibernate's JPA implementation. The application uses a simple text-based interface with only 3 options (enroll, drop, and list) as illustrated below.

```
code   |title                                |instructor     |max|actual|remain
CS1030 |Computer Science Principles          |Jody Paul      |005| 004  | 001
CS1050 |Computer Science 1                   |David Kramer   |003| 000  | 003
CS2050 |Computer Science 2                   |Steve Geinitz  |003| 001  | 002
CS3810 |Principles of Database Systems       |Thyago Mota    |002| 000  | 002
[1:enroll|2:drop|3:list|4:exit]?
```

The application uses the MVC (Model-View-Controller) design pattern with the following classes:

- Model:
  - `Course`
  - `Enrollment`
  - `EnrollmentPK`
  - `Student`
- View:
  - `Main`
- Controller:
  - `Controller`

Get the initial code for this project from https://github.com/thyagomota/21SCS3810/tree/main/db03_enrollments.

To facilitate future grading, please use the `META-INF/persistence.xml` file available in the assignment's GitHub repository. The `META-INF` folder containing the XML file should be placed in your Maven project under `resources`.

The Model

The model is responsible for defining the objects of the application, which will use a MySQL database for persistency. The `enrollments.sql` script is available for you to create the `enrollments` database, consisting of the following tables: `courses`, `students`, and `enrollments`. The relationship between those tables should be self-explanatory. You are asked to create 2 triggers and 1 stored procedure for the `enrollments` database:

- `enroll_student`: whenever a student enrolls in a course, this trigger should increment the `actual` field (in `course`).
- `drop_student`: whenever a student drops from a course, this trigger should decrement the `actual` field (in `course`).
- `list_students`: given a course code, this stored procedure should return a list of ids and names of all students currently enrolled in the given course.

The `Student` class defines an entity that has a direct map to the `students` table. This class is given to you to help you get started.

Similarly, the `Course` and the `Enrollment` classes define entities that map to the `courses` and the `enrollments` tables, respectively. Because the `enrollments` table has a compound key (based on course `code` and student `id`), you will be required to create a separate class named `EnrollmentPK` to represent the key for the `Enrollment` entity. Lesson 17 has an example that uses an entity with a compound key.

The Controller

The Controller class has the following methods left for you to implement:

`Student getStudent(int id)`	returns a Student entity from the given id (or null if the entity does not exist)
`boolean addStudent(final Student student)`	adds the given student entity, returning true/false depending whether the operation was successful or not
`List<Course> getCourses()`	returns a list of all Course entities
`boolean enrollStudent(String code, int id)`	enrolls a student to a course based on the given parameters, returning true/false depending whether the operation was successful or not
`boolean dropStudent(String code, int id)`	drops a student from a course based on the given parameters, returning true/false depending whether the operation was successful or not
`List<Student> getStudentsEnrolled(String course)`	returns a list of all Student entities enrolled in the given course (hint: use the stored procedure 'list_students')

To implement `getStudentsEnrolled` you will need a reference to the database connection. Below is a snippet of code that shows how to get that reference.

```
SessionImpl sessionImpl = (SessionImpl) session;
Connection conn = sessionImpl.connection();
```

The View

The view is implemented by the `Main` class and the code was given to you. Enjoy!

Suggested Step-by-step Sequence

- create the database
- create the 2 triggers
- test the triggers using the MySQL shell tool
- create the stored procedure
- test the stored procedure using the MySQL shell tool
- create a Maven project
- configure pom.xml with the dependencies
- add META-INF/persistence.xml under resources
- configure persistence.xml appropriately
- implement getStudent in Controller and test it separately
- implement addStudent in Controller and test it separately
- create the Course entity
- implement getCourses in Controller
- run main and check if the application shows the list of courses
- implement getStudentsEnrolled
- run main and check if the application shows the list of students enrolled in a course
- create the EnrollmentPK and Enrollment entities
- implement enrollStudent in Controller
- run main and check if the application is able to enroll a student in a course
- implement dropStudent in Controller
- run main and check if the application is able to drop a student from a course

Deliverables

A zip file containing the following files:

- `enrollments.sql` (with your implementations for the triggers and the stored procedure)
- `Course.java`
- `EnrollmentPK.java`
- `Enrollment.java`
- `Controller.java`

I only accept zip format!!!

Rubric

+10 for the 2 triggers (+5 each)

+10 for the stored procedure

+10 Course entity

+10 EnrollmentPK entity

+10 Enrollment entity

+5 Controller's getStudent

+5 Controller's addStudent

+10 Controller's getCourses

+10 Controller's enrollStudent

+10 Controller's dropStudent

+10 Controller's getStudentsEnrolled

-5 didn't named participants in the team

-5 didn't use the deliverable format (zip and the exact files asked inside the zip)

Total: +100