☐ Mark as done

In this milestone, you have learned how to integrate Java code with relational databases using Spring JdbcTemplates. With this in mind, we would like you to try your hand at designing a database and JdbcTemplate-based DAO for a superhero sightings application.

Requirements

With the rising popularity of superhero movies, there has been a heightened awareness of superheroes in our midst. The frequency of superhero (and supervillain) sightings is increasing at an alarming rate. Given this development, the Hero Education and Relationship Organization (HERO) has asked our company to develop a database and data layer for their new superhero sightings web application.

The system has the following requirements:

- It must keep track of all superhero/supervillain information.
  - Heroes have names, descriptions, and a superpower.
  - Heroes are affiliated with one or more superhero/supervillain organizations.
- It must keep track of all location information:
  - Locations have names, descriptions, address information, and latitude/longitude coordinates.
- It must keep track of all superhero/supervillain organization information:
  - Organizations have names, descriptions, and address/contact information.
  - Organizations have members.
- A user must be able to record a superhero/supervillain sighting for a particular location and date.

- The system must be able to report all of the superheroes sighted at a particular location.
- The system must be able to report all of the locations where a particular superhero has been seen.
- The system must be able to report all sightings (hero and location) for a particular date.
- The system must be able to report all of the members of a particular organization.
- The system must be able to report all of the organizations a particular superhero/villain belongs to.

Deliverables

To complete this assignment, you must deliver the following items:

- An Entity-Relationship-Diagram
  - You may use MySQL Workbench to create a diagram or your choice of alternative tools such as Pencil, Draw.IO, or LucidChart.
  - The database must achieve 2nd normal form at minimum.
  - Proper naming conventions should be used.
  - The ERD should be very easy to read, with all components clearly labeled.
  - Use a common and appropriate file format, such as a png or jpg image or a PDF document.
- A database creation script
  - The script should create the database with all tables, columns, and relationships.
  - Make reasonable assumptions about column data types; be prepared to justify your decisions.
  - The script should be re-runnable. This means it should drop the database and all objects if they exist and recreate them. You should be able to execute the script many times in a row without error. See the scripts provided for the databases used in the Relational Database unit for examples.
- DAO Implementation and Unit Tests
  - DAO should have an interface and an implementation.
  - DAO and DTOs must fully represent all data and relationships contained in the database design.
  - Implementation must make proper use of transactions.
  - Unit tests must fully test all create, read, update, and delete functionality for all entities and test all many-to-many and one-to-many relationships in the database.

Slack and Forum Rules

You are free to discuss at a high level tables, columns, and relational assumptions with your cohort in Slack.

If a portion of your script does not work, feel free to post snippets of the specific portion that does not work.

Submitting Your Assessment

When you are satisfied that your project meets all requirements take the following actions:

1. Submit your files following instructions provided in the course.
2. If you are attending the Guild online, schedule a time with a staff member to review your code. If you are attending the Guild in person, your code will be reviewed during the weekly code review.
3. Be prepared to answer questions about your code and thought processes.

Add submission

Submission status

Submission status	No submissions have been made yet
Grading status	Not graded

Grading criteria

ERD - Second Normal Form: The database design meets AT LEAST the requirements for second normal form.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
ERD - Primary Keys: Each table has an appropriate primary key named either id or entityId or an appropriate composite key with all fields named correctly.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
ERD - Relationships: The relationships between the tables are expressed correctly with foreign keys on entity tables for one-to-many relationships and bridge tables for many-to-many relationships.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
ERD - Data Columns: Each field has reasonable types and scales.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***

Script Re-runnable: The script used to create the database can be re-run as needed, including dropping the database, rebuilding it, and USE the database.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
Script - Fits ERD Specification: The script creates tables named as presented in the ERD, with columns specified according to the ERD requirements, all foreign key relationships present, and auto-incrementing where appropriate.	Meets Expectations ***15 points***	Needs Improvement ***9 points***	No Credit ***0 points***
DTOs - Full Entity Modeling: All database entities have equivalent class representations with appropriate fields, getters, and setters.	Meets Expectations ***10 points***	Needs Improvement ***6 points***	No Credit ***0 points***
DAOs - Separation of Concerns: Each entity has its own DAO.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***

DAOs - Interface Extraction: Each DAO implements a method that defines all of its public methods.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
DAOs - Full CRUD Operations: Each DAO implements add, getById, getAll, edit, and delete methods.	Meets Expectations ***10 points***	Needs Improvement ***6 points***	No Credit ***0 points***
Unit Testing: Golden Path Testing Unit tests verify that all public dao methods perform their operations as expected given valid inputs, including validation persistence checks that do not trust the return value of the method being tested.	Meets Expectations ***10 points***	Needs Improvement ***6 points***	No Credit ***0 points***
Unit Testing -Bad Input Testing: Unit tests verify that all public dao methods produce the correct exception types on invalid inputs, including invalid ids and nulls.	Meets Expectations ***10 points***	Needs Improvement ***6 points***	No Credit ***0 points***

	Code Style: All code is written in a readable, conventions-compliant manner.	Click to edit level ***10 points***	Click to edit level ***6 points***	Click to edit level ***0 points***
Last modified	-			
Submission comments	▸ Comments (0)			

◄ Previous activity

Jump to...

Next activity ►