Assignment 1

Note: **this assignment is worth 70%** - this differs from the module description, which suggests 30% initial assignment followed by a 70% assignment. The final assignment will be worth 30%.

Due date: 18th November

Overview

Following on from the tutorial entitled "Parsing and Mapping JSON to Java Objects", you complete the exercise that requires you to add additional properties to the Artist class and add an Artwork class. You will then create Artist and Artwork JDBC Repository classes to add the artists and artworks to a MySQL database. Finally, you will develop a web page to display details for a single artist.

You will complete parts 1 to 5 as a group. Part 6 is an individual effort.

Part 1 – Parsing the JSON to Java Objects

The UML diagram on page 3 shows that an Artist has 0 or more Movements, 1 Birth and 0 or 1 Death records. Add the Movement and Birth, matching properties to the fields in the JSON files. (Ignore Death for this assignment and no need to create classes for Places, Eras, etc.).

Add a "movements" property as a List<Movement> in Artist.

Add a "birth" property as a Birth in Artist.

Add some other simple properties (i.e. non-sub-document fields in the JSON file) to the Artist class.

Examine the Artwork JSON files. Create an Artwork class that contains Movements, much like an Artist contains a list of movements. Add some other properties (no need to add all, just the most interesting / simple / non-sub-document ones).

Now add a list of Artworks as a property of the Artist class.

Part 2 – Adding artists and artworks to the database

Create tables for artists, artworks, movements and births. For the purposes of this assignment, it is assumed that 1 artist can have many artworks and an artwork is created by 1 and only 1 artist. However, the relationships between artists and movements and between artworks and movements is many-to-many, so link tables to connect artists and movements (e.g. artist_movements) and artworks and movements (e.g. artwork_movements) will be required. A birth will contain the artist id as a foreign key.

Create a JdbcArtistRepository class and a JdbcArtworkRepository class. As each artist is read in from a JSON file, call a method in JdbcArtistRepository called *save*, passing the Artist object as a parameter. Use the save method to:

- Add the artist (either use SimpleJdbcInsert to retrieve a generated id, or rely on the "id" field in the JSON file to use as a foreign key value in the artist_movements and birth tables).

- If there are any movements in the movements list of the artist, check first if the movement exists in the database and if not, add the movement. Link the artist to any movements via the artist_movements link table.
- Add the birth record, with the artist id as a foreign key.

Do something similar for the artworks, i.e. call save in JdbcArtworkRepository.

Part 3 – Artist list and detail page

Create an artist list page (you can limit this to 40 artists on a single page, e.g. use LIMIT 40 in your query or some other way of reducing the numbers). List clickable artist names on the page. The URL for this page will be http://localhost:8080/artist/.

When the user types the following URL into their browser (or clicks on an artist name), they should see an artist detail page:

http://localhost:8080/artist/{id}

(replace {id} with the artist id number)

This should present all of the details from the artist table, the birth record of the artist, and output a list of movements that the artist participated in, and output a list of artworks to include artwork title, classification, medium, and any date information.

You should add a get method to your repository class. It would simple take the id as a parameter and retrieve all the necessary data for the artist, including birth, movements and artworks.

Part 4 – Artwork list and detail page

Develop an artwork list page with the clickable names of artworks and the artist name in brackets. Develop another web page specifically for an artwork. The url would be http://localhost:8080/artwork/{id}

This will output artwork details (including displaying the thumbnail – see thumbnailUrl in the JSON file). It will list the movements for the artwork. It will display some details of the Artist that created the artwork.

You should add a get method to your repository class. It would simple take the id as a parameter and retrieve all the necessary data for the artwork, including movements and artist details.

You should add Artist as a property of the Artwork class.

Part 5 – Allow a logged in user to leave a comment on an artwork

Create a user table (as per instructions in lectures and tutorials). Create a login page (http://localhost:8080/login).

Create a comments table for artworks. This should include userid, artwork id, the comment and a timestamp.

On the artist detail page, list all comments for the artwork. Add a text box and submit button above or below the comments to allow a logged in user to add a comment for the artwork. This mini-form should be invisible to non-logged in users.

Part 6 – Reflective document and peer assessment

You will submit a document of up to 1000 words detailing your experience of this project and what you contributed, any obstacles overcome, new technologies learned, etc. There will also be a peer assessment form to fill in that will affect the distribution of some of the marks.

Notes

- For parts 3 and 4, the get method will return either an Artist or Artwork. You could consider the ResultSetExtractor callback method as a way to do this, i.e. the get method in the repository calls JdbcTemplate's query method with a complex join query (e.g. joining artist to movements via the link table, and the birth record, etc.) and then constructs the Artist or Artwork object with all properties, including lists of movements and other objects.
- Marking scheme:
    - Part 1 – 12%
    - Part 2 – 18%
    - Part 3 – 20%
    - Part 4 – 15%
    - Part 5 – 15%
    - Part 6 – 20%

Instructions

- You must use Spring Boot, JdbcTemplate and Thymeleaf.
- Parts 1, 2 and 3 to be developed as a group using a shared public repository in BitBucket.
- Part 4 to be developed in a private repository that you have granted me access to.
- There will be two submissions using Blackboard. 1 – zip file and bitbucket url for the group work (one team member submits); 2 – zip file and bitbucket url for the individual project (includes all of the group work plus your additional work). Include the database schema as an SQL file in src/main/resources.
- Usual penalties for late submission apply (10 mark penalty up to 1 week late, 20 mark penalty up to 2 weeks late, 0 marks thereafter). There can be no extensions.