Notes From JDBC Template for Assessment

When using a Rowmapper, be sure that the import is:

Import org.springframework.jdbc.core.RowMapper;

To Use JDBC (Java Database connection), you will need the two following dependencies:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

<version>2.0.5.RELEASE</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.13</version>

</dependency>

The first is for the JDBC itself and the second is to make the connection to the SQL server.

The main APP class needs to be annotated with “@SpringBootApplication”, we do not need to implement the command line runner interface, because it is not needed for the assessment. Therefore, the example from the Complex relationships, we don’t need that overridden run method.

We need to set up the database connection by making a file in the src-main-resources folder called application.properties. In this file, we need the following:

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/todoDB?serverTimezone=America/Chicago&useSSL=false

spring.datasource.username=root

spring.datasource.password=rootroot

logging.pattern.console=

In the second line for datasource URL, we replace the “todoDB” with the name of our database that we are working with.

The “logging.pattern.console=” is being used for this complex relationship project to suppress Spring Boot console output so we only see our own output. We typically wouldn’t use this property, but it helps our small project work a little better. See if this is needed for the REST API. Edit: It is not needed!!!

Now we have to add in the following to our main App class:

@Autowired

private JdbcTemplate jdbc;

I’m not sure if this is needed again in the main app class. It is clear that we need to access the JDBC in the Dao’s where we will actually be talking to the databases, so it seems like it will most likely go there and not in the main app class. Edit: the jdbc definitely gets included in the dao’s they are a part of.

We will need RowMappers as private methods inside each Dao class. Sample Mapper:

private static final class ToDoMapper implements RowMapper<ToDo> {

@Override

public ToDo mapRow(ResultSet rs, int index) throws SQLException {

ToDo td = new ToDo();

td.setId(rs.getInt("id"));

td.setTodo(rs.getString("todo"));

td.setNote(rs.getString("note"));

td.setFinished(rs.getBoolean("finished"));

return td;

}

}

We will need to make a GameDao and a RoundDao. Each needs to have the CRRUD methods.

Game Object in Java

gameId int, Answer String, Status boolean, ListOfRounds List<Round>

Round object in Java

roundId, gameId, guess, timeOfGuess, resultOfGuess

The database needs to be made first.

Don’t forget to make SQL queries final!!!

GameDao

Needs @Repository Annotation

Needs JDBC Autowired in the constructor

@Autowired

public ToDoDatabaseDao(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate = jdbcTemplate;

}

Needs rowMapper, need to determine if it should be public/private

Public static Game getGameById(int gameId); (Use helper method)

Public static List<Game> getAllGames(); (Use helper method??????????)

Public static Game addGame(Game game);

Add to database, then get the last id made, return the object you just added.

Public static Game updateGame(Game game);

Will only be updating the status of the game, I think. Has no effect on Round table.

Update in database, get the object from database, then return that object

Public static Game deleteGameById(int gameId);

Must first delete all rounds from Round table that have this gameId, then delete the game. Note to self – this should not be calling on the RoomDaoDB method to delete. This needs its own delete

Get object from database, delete it, return the object.

HELPER METHODS

Pu/pr static List<Round> getRoundsForGame(Game game); (SHOULD THIS TAKE IN THE FULL GAME OR JUST THE GAMEiD?) **Edit**: made it take in the gameId as an int, added another method to add these to the game object.

RoundDao

Needs @Repository Annotation

Needs JDBC Autowired in the constructor

Needs a rowMapper for Round, determine if it needs to be private/public

Public static Round getRoundById(int id);

Public static List<Round> getAllRounds();

public static List<Round> getAllRoundsForGameId(int gameId);

Public static Round addRound(Round round);

Public static Round updateRound(Round round);

Public static Round deleteRoundById(int id);

ONCE THIS IS DONE, WRITE THE TESTS FOR THEM!!!!

<!--The Spring Boot starter parent. Controls <dependencies> below.-->

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.0.RELEASE</version>

<relativePath />

</parent>

<dependencies>

<!--Spring Boot starter children. No versions needed.-->

<!--Enables and configures Spring MVC. -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!--Database dependencies -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<!-- Spring Boot Dev Tools -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

Application.properties file needs the following:

spring.datasource.url=jdbc:mysql://localhost:3306/todoDB?serverTimezone=UTC&useSSL=false

spring.datasource.username=root

spring.datasource.password=your-root-password

spring.profiles.active=database

CLEAN AND BUILD

DON’T FORGET!!! **PROFILES** FOR TESTING AND EXCEPTION HANDLING!!!!

Check back video for commentary on the generated Keyholder and the channel on how to do the exceptions

Guess the Number

Write a REST server to facilitate playing the number guessing game "Bulls and Cows"

For each GAME,

generate a 4-digit number

every digit must be different.

(Can we allow zeroes to be a part of the number? At any position?)

In each ROUND,

a user guesses a number

they are told how many exact and partial matches they made.

The GAME ends when

the number is guessed (exact matches for all digits).

Game Operation

START - Create the four digit number.

EACH ROUND - User inputs a four digit number.

We take that number and compare it to our number.

We test for exact matches (if digit and position matches, then true)

We test for partial matches (fore, if answer contains digit from user, true)

We show the user the result of that round.

Database Storage

Each game should have an answer and a status (in progress or finished).

While the game is in progress, user should not be able to see the answer.

Each round will have a guess, the time of the guess, the result of the guess in the format "e:0:p:0" where "e" stands for exact matches and "p" stands for partial matches.

Game Table (ONE)

GameId int ai

Answer char(4)

Status bit (0 is in progress, 1 is finished)

Round Table (MANY)

RoundId int ai (This means that a user can enter the same value for two rounds, should this be allowed?)

Guess char(4)

TimeOfGuess (DATETIME or TIME)

ResultOfGuess VARCHAR(11)

GameId (FK)

GAME has a ONE-TO-MANY relationship with Round because each game has many rounds and many rounds belong to each game.

===========================================================================

List of required REST Endpoints/ (controller methods?) (@RequestBody!!!!???!?!?!)

/begin (POST)

This will start a game, generate an answer, and set the set the status to incomplete.

MAKE SURE YOU DO NOT SEND THE ANSWER OF THE GAME TO THE USER!!!!!

It will return an HTTP Status of 201 CREATED and the created GameId.

/guess (POST)

This will be how the user enters a guess. The user will enter a guess and a gameId as JSON. (These are some of the entries of the Round Object, use setters to set them.)

The program will take this and calculate the results of the guess (Job of the service)

It will also need to mark the game as finished IF the guess is fully correct! (Job of service).

It will return the round object with the results filled in.

/game (GET)

Returns a list of all Games. Service - Be sure that in-progress games do now display their answer!!!! (Should this be returning a list of all the rounds with the finished game?)

/game/{gameId} (GET)

Returns a specific game based on Id. Be sure in-progress games do not display their answer!!!!!! - Service!!!

/rounds/{gameId} (GET)

Returns a list of rounds for the specified game sorted by time.

===========================================================================

Service Layer methods

beginGame - This will generate a number for the game. Then enter the new game in the GameDaoDB by using addGame. It needs to return the newly generated gameId

public static int beginGame();

validateGuess - This will take in the round object from the user. It will contain the guess and the gameId. It will then take that gameId and get the game from the gameDaoDB, so it can see the answer. Then run the validation to see how it compares to the answer of the game. It will then set the result in the round object, as well as the timeOfGuess. Then it will add that to the roundDaoDB. Then if the resultOfGuess is "e:4:p:0", it will update the GameDaoDB to have a status of finished. Then it will return the round object with the results filled in.

CHECK TO SEE HOW THIS WILL BE RETRIEVED FROM USER... ?!?!

public static Round validateGuess(Round round);

(DO WE NEED ANYTHING TO INDICATE THAT A GAME IS FINISHED? - How would this even be done?)

getAllGames - this will call on the GameDao to getAllGames. Then run through each game and if the game status is still in progress, it should change the answer to "hidden" or something else cute. Keep playing to find out.

(DOES THIS MEAN THAT THE GETALLGAMES IN THE GAMEDAODB DOES NOT NEED TO PULL IN A LIST OF ROUNDS FOR EACH GAME? BUT LATER ON, THE GETGAMEBYID NEEDS TO? OR SHOULD IT BE GETTING THE LIST OF ROUNDS ANYWAY? I think it should, but what about this method that doesn't need the rounds with the games? Should it be pulling it in and then the service stripping it of some details?)

public static List<Games> getAllGames;

HOW DOES THE JSON SHOW A LIST OF GAMES????? It just does. Don’t worry about this.

getGameById - takes in a gameId and gets the game from the GameDaoDB. Then it checks to see what the status of this game is. IF the status is not finished, it will remove the answer from it.

public static Game getGameById(int gameId);

SHOULD THERE BE A METHOD THAT CLEARS THE ANSWER FROM A GAME? BECAUSE RN, TWO METHODS NEED THIS FUNCTIONALITY.

getAllRoundsByGameId - calls on GameDaoDB to get a game object which will have a list of rounds in it. Then it will extract this list and sort the list by the time and then return this list.

public static List<Round> getAllRoundsByGameId(int gameId);