**Assignment 6 Documentation**

JDBC Configuration

JDBCTemplate was configured using the application.properties file as shown below:

Text

Description automatically generated

PLEASE NOTE THAT THE DATASOURCE.PASSWORD ATTRIBUTE WILL BE DELETED AND YOU’LL HAVE TO PUT IN YOUR OWN MYSQL DATABASE PASSWORD!

JDBC CRUD

Game and Round objects are created:

Text

Description automatically generated

And updated:

Text

Description automatically generated

Through the used of a JDBCTemplate object that is contained within the DAO class. I didn’t implement a deletion feature as I didn’t believe it was necessary given that the REST endpoints we were asked to implement did not mention deleting a game or round at any point.

DAO and Game Logic Tests

Tests for the DAO and game logic were not implemented in the interest of saving time (we are approaching the end of the training after all).

Errors could arise through the input of invalid data using Postman, such as passing in a number with more than 4 digits when making a guess. This would’ve been dealt with by having the controller validate the input at the input’s point of entry into the program, or possibly through passing the input to the GameLogic class to validate (i.e. a function returning a Boolean value based on whether or not the input is valid).

Whilst the database is (to the best of my knowledge (I do realise saying that without fully testing it is silly)) free of errors when being updated through the JDBCTemplate, an error does occur when rounds are added through MYSQL scripts within MYSQL Workbench. This could possibly be fixed through the use of constraints to have the games automatically update the number of guesses and the win/loss values whenever a new round is introduced (I do believe this is possible but not 100% sure).

To conclude, I am aware of certain areas where errors could arise, and I would’ve implemented tests to check these points, if I had the time.

Spring Boot Usage

Spring Boot is used to run the application:

Text

Description automatically generated

And annotations are used for dependency injection:

Text

Description automatically generatedGraphical user interface, text, application

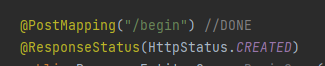
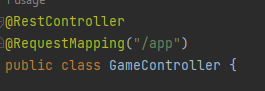
Description automatically generated

Service Components

The service layer functionality was implemented inside the GameLogicImpl class, inheriting from the GameLogic interface. This class was used to evaluate the result of a guess, create new random answers for new games that were started, and finally to hide the answer to the game if the game had not yet been completed. (I’d include screenshots, but I’d just be screenshotting 70-odd lines of code which you are most likely going to review anyway).

Spring MVC Controller & Controller Handler Methods

The controller for the program is a Spring REST controller that handles the HTTP requests sent by Postman.





Gameplay

Through the use of Postman you can:

1. Graphical user interface, text, application, email

   Description automatically generatedBegin a new game

HTTP response is correct (Code 201 CREATED)

Answer is hidden for unfinished games

1. Make a guess for an existing game

Real answer for game 11 was 6785

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generated

1. Get all games

Graphical user interface, text

Description automatically generated etc.

1. Get a specific game using the game id

Graphical user interface, text, application

Description automatically generated

1. Getting all rounds for a game

Graphical user interface, text, email

Description automatically generated