**PROG32758 Midterm Exam /100**

***(Open Book, Practical Exam)***

*Part 1: Project Setup (5 marks)*

Create a Spring Boot project. You will need Lombok, Spring Web, Thymeleaf, H2 database, and Spring Database JDBC as dependencies. Develop your midterm project using all the JAVA, HTML and configuration files needed to access the Spring embedded database called H2 as shown during lectures. Troubleshoot as needed. At a minimum the program should perform the following CRUD tasks and represent the different fields and datatypes.

***You have a 2 HOUR TIME LIMIT to complete this Midterm exam, that means that you MUST UPLOAD your ZIP or compressed source folder to the Midterm Dropbox with corresponding screen shots and answer to the 5 short answer questions by 2:30PM.***

***Projects that DO NOT COMPILE and deploy successfully will NOT CONSIDERED A PASS (or 50%).***

Make sure that the project is named ***Midterm\_FirstName\_LastNam***e where FirstName is replaced with your first name, and LastName is replaced with your last name.

You should have a root page which is mapped to *http://localhost:8080/*. Your home page should have a link to each part. Each part should have a hyperlink back to your root page.

*Part 2: Bean (15 marks)*

Create a Bean or POJO in a ***ca.sheridancollege.<yourUserName>.beans*** package that contains the following fields :

* ID: Auto generated when added into the database
* A private String Airplane
* A private String Manufacturer
* A private String Propulsion
* A private LocalDate deliveryDate
* A private LocalTime deliveryTime

*Part 3: Schema.sql and H2 Table (5 marks)*

In an SQL file, create a table for the bean where columns match the fields. Create at least 3 sample records to be generated when the program starts. Consider the following statements:

**CREATE TABLE** airplane(

id LONG **PRIMARY KEY** AUTO\_INCREMENT,

airplane **VARCHAR**(255),

manufacturer **VARCHAR**(255),

propulsion **VARCHAR**(255),

deliveryDate **DATE**,

deliveryTime **TIME**

);

**INSERT INTO** airplane(airplane, manufacturer, propultion, deliveryDate, deliveryTime) **VALUES**

('737', 'Boeing', 'Diesel', '2020-01-01', '12:00:00'),

('A320', 'Airbus', 'Diesel', '2020-02-02', '08:15:00'),

('C-Series', 'Bombardier', 'Solar-Electric', '2020-03-03', '14:30:00'),

('767', 'Boeing', 'Gas', '2020-04-04', '10:15:00');

*Part 4: addItem EDIT/UPDATE (5 marks)*

Create an add item HTML page that will add an Airplane item to the database. After adding an Airplane item your program should return to the add item page.

*Part 5: Controller (5 marks)*

Display all Airplane items in an HTML page. This should be done in a well formatted HTML table with column headings for each field.

Create a Controller called AirplaneController in a ***ca.sheridancollege.<yourUserName>.controllers*** package. Inside, create methods mapped to “/”, insertAirplane, editAirplane, and deleteAirplane for the different CRUD operations.

*Part 6: index.html (6 marks edit, 4 marks delete)*

Create an *index.html*.

Your *index.html* page should automatically display all entered Airplanes at launch (i.e. loading of “/”), along with unique Edit and Delete buttons for each, and then provide your users a simple form where they can enter additional Airplanes info following the POJO.

*Part 7: Database Access and Config (5 marks)*

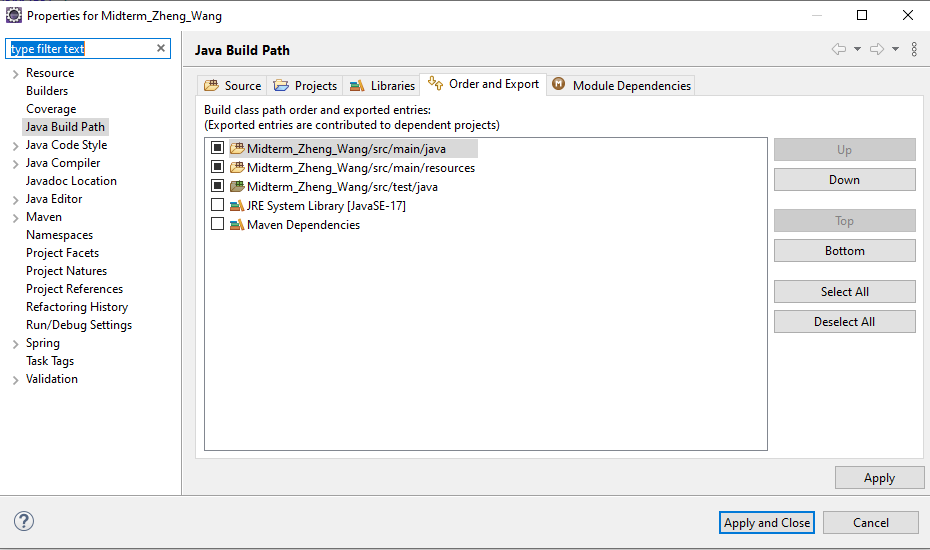
Create the ***DatabaseAccess.java*** and the ***DatabaseConfig.java*** files in a ***ca.sheridancollege.<yourUserName>.database*** package. Include the code to load the ***shema.sql*** file that creates the Airplanes table, etc.

*Part 8: PROOF IT WORKS (Screen shots) (30 marks)*

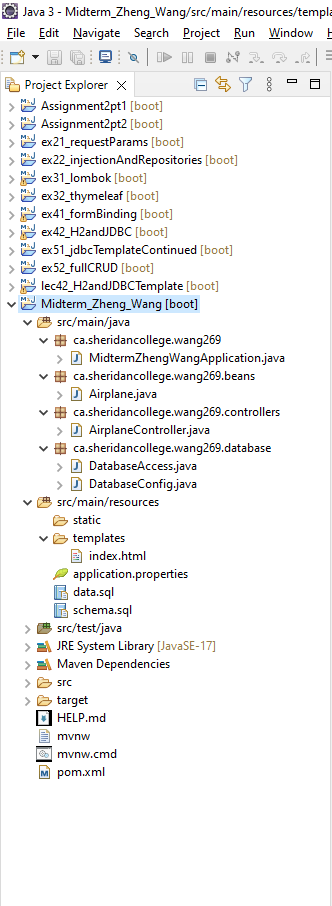
Upload a screenshot of a bunch of manufacturers in your HTML list. Edit one that would be obvious, and delete two of them. Take another screenshot and upload it as well. Remember to name your screenshots using your own name plus a 1 or 2 for their filenames.

In your submission include the following screen shots:

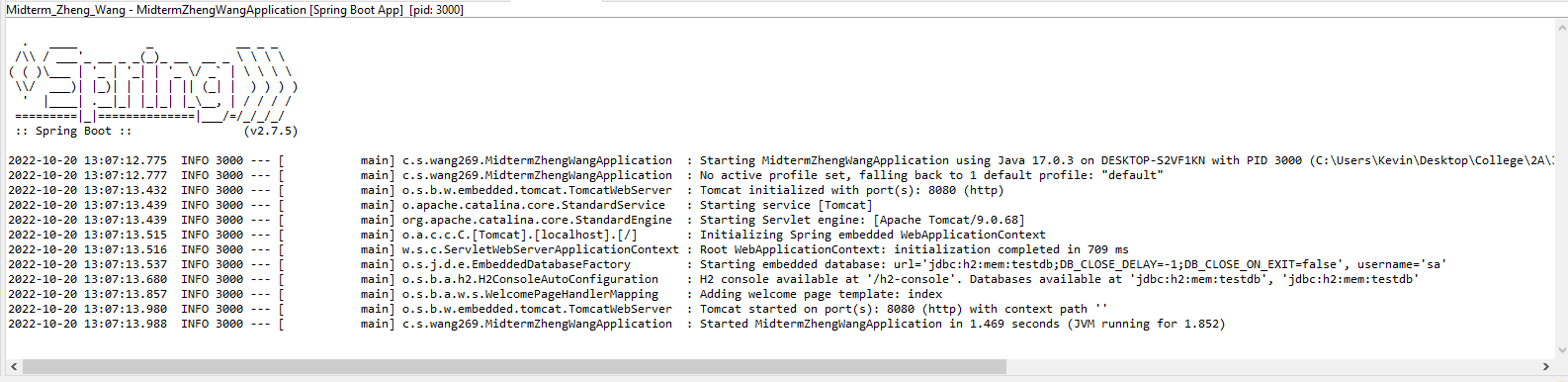
1. The Buildpath settings for your project



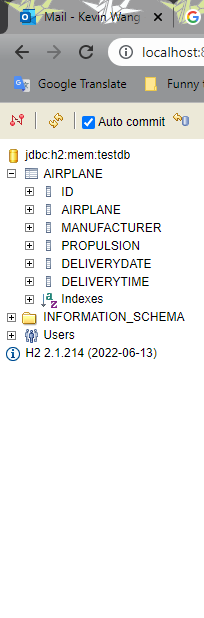
1. The complete and detailed project structure:



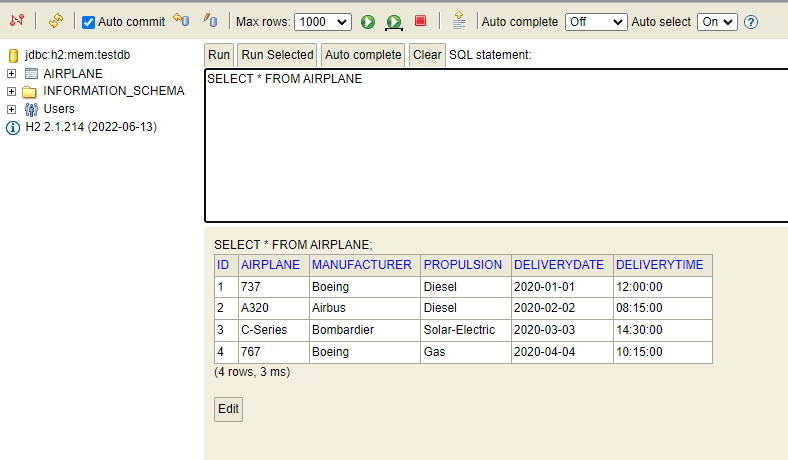
1. The Console output of your WebAPP when deployed to the Spring Framework



1. The H2 Web Console showing the table. Make sure to expland an clearly show the table structure including all columns and fields.

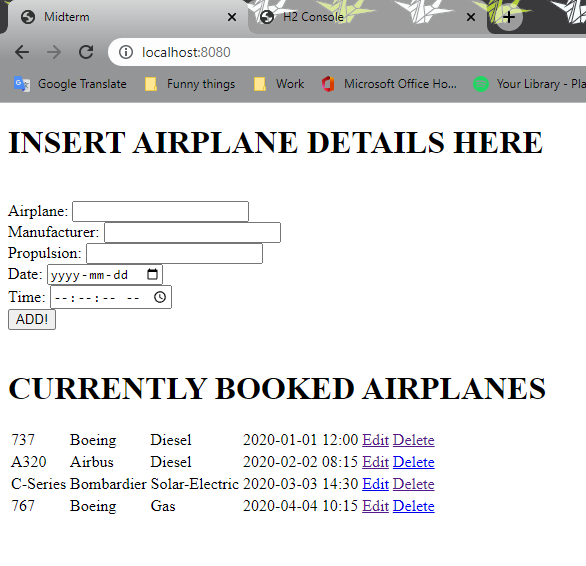


1. A SELECT ALL or QUERY operation

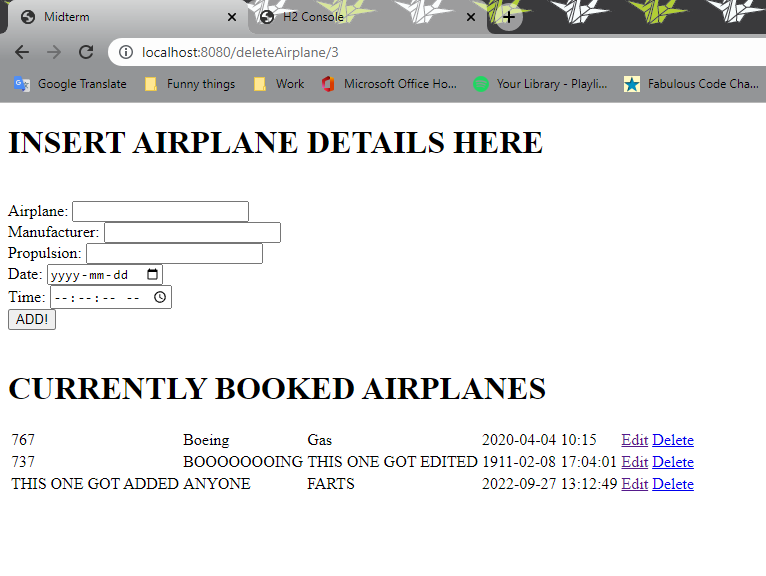


1. Either or of: INSERT, UPDATE and DELETE.

Not modified:



After Modification:



*SUPPLEMENTAL QUESTIONS (20 marks; 4 marks each)*

*For each question, point to or show the source code that is putting into practice the concept you are asked about. Be as detailed as possible.*

1. *Where are the @Data and @Configuration annotations typically used ? What are their roles ? Explain through code examples.*

**The @Data is a Lombok call for Lombok to create getters and setters for the class as well as creating a toString() method. @Configuration tells this class that it’s going to use @Bean annotation to make beans for the Spring container at runtime.**

**For example, in this program the @Data is used in the airplane POJO and @Configuration is used in the DatabaseConfig class**

1. *What is a named parameter ? How is it used in your project ? Explain.*

**A namedparameter is a parameter object that the database class instantiates to map sql parameters with the sql query. In my project it’s used to map the airplane, manufacturer, propulsion, deliveryDate, and deliveryTime attributes to its corresponding values.**

1. *What is the difference between the DatabaseAccess.java and DatabaseConfig.java classes ?*

**DatabaseAccess is the class that has all the methods which the controller can call upon for manipulating the database. The database config is there to set up (configure) the database on boot.**

1. *What is the role of a Bean ? Explain.*

**A bean, or POJO, is like a container for the information of a single object. In our project our objects are airplanes, so our bean contains all the pertinent information about the details of the airplane.**

1. *What is the different between the @RequestParam and @PathVariable annotations used in a Spring Controller class ? Explain.*

**RequestParam asks the html what’s in the text field with that specific id and gets the information directly from the textfield while PathVariable takes the URI path and gets information from the path.**