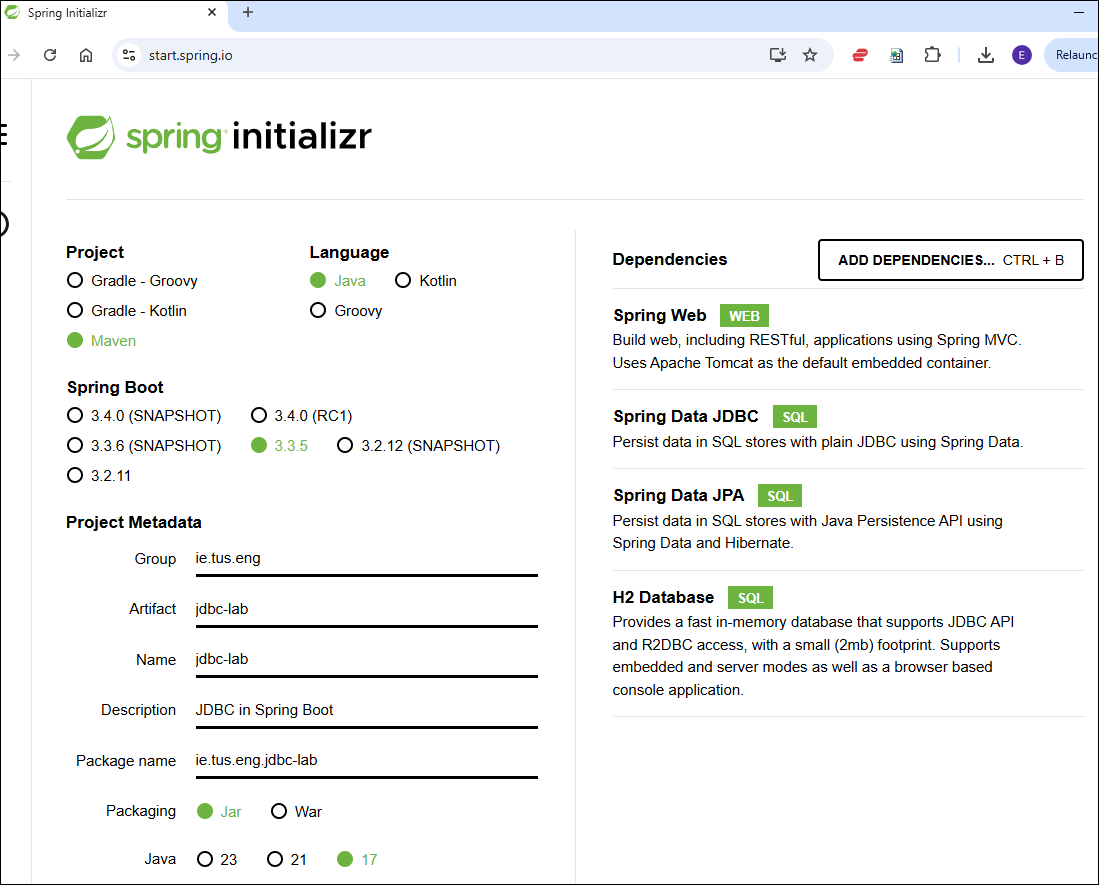
**Lab – Databases in Spring Boot - JDBC**

This lab introduces **databases** and **JDBC** in Spring Boot. The following is covered:

1. Setting up a database in Spring Boot
2. Inserting data into the database using JDBC
3. Selecting data from the database using JDBC

No RESTful Web Services are covered – this is just straightforward reading/writing to a DB.

~~Make the following project at start.spring.io:~~

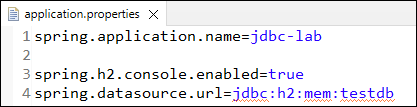


~~As usual, download, unzip and import into Eclipse.~~

Import -> Existing Maven Project

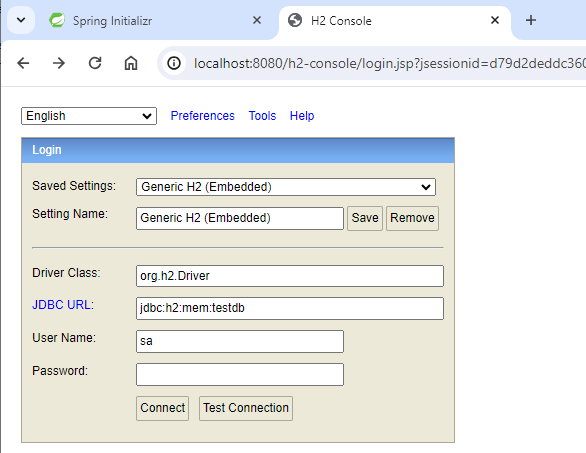
**~~Part 1: Connecting to and configuring the Database:~~**

First, to enable the H2 database, go to the ***application.properties*** file and add the following lines:



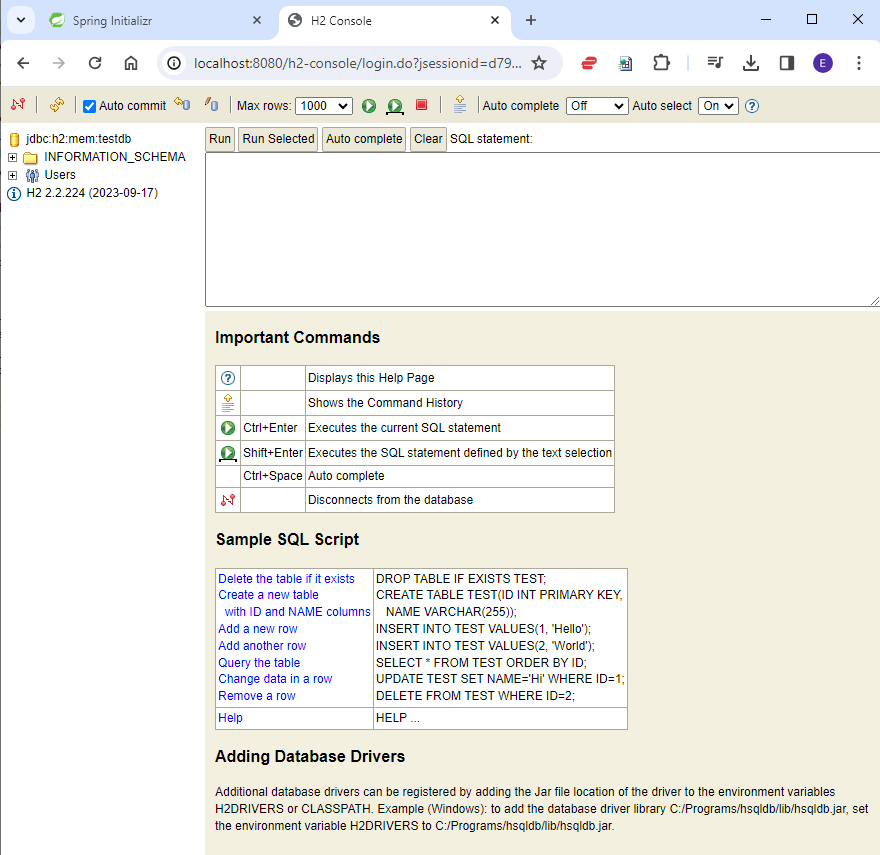
~~Now, start/restart the application.~~

~~We can access the database configuration console at~~ **localhost:8080/h2-console**:



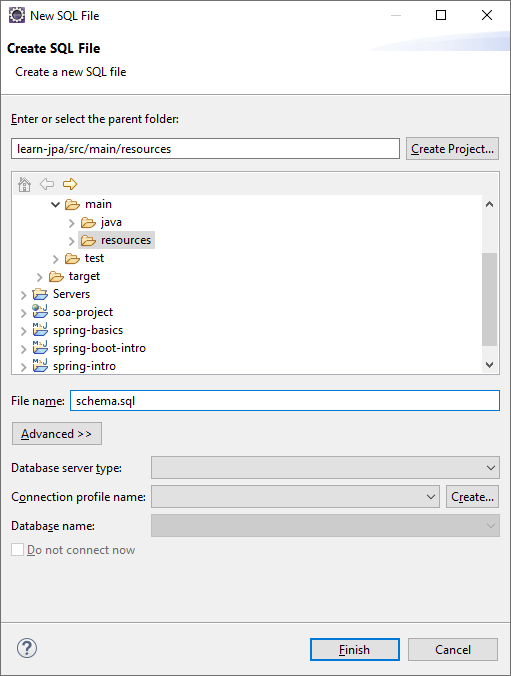
~~We need to fill in the JDBC URL. We use the URL we configured in~~ *~~application.properties~~* ~~file as shown, and click connect.~~

The following screen will appear:

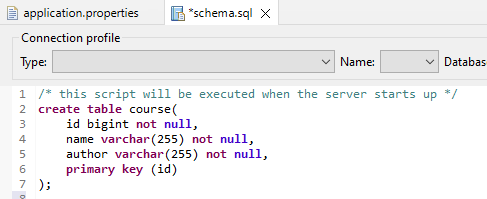


There are no tables created. We’ll create a file schema.sql which contains the script we want to run. This will then be automatically executed when we start the application.

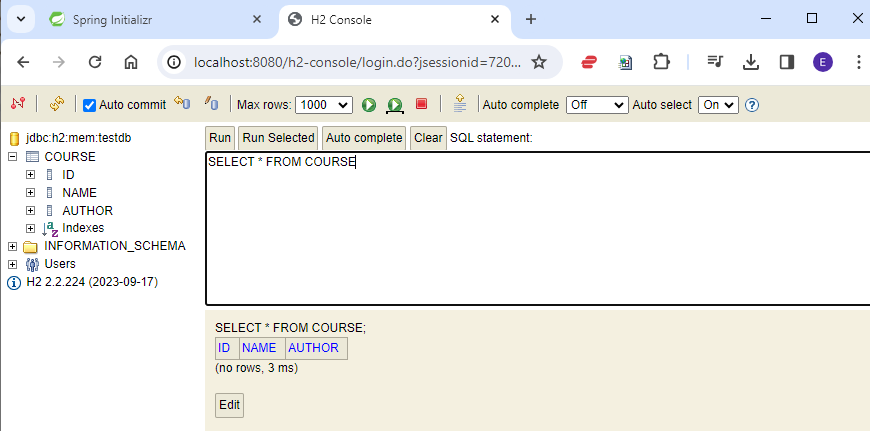
In src/main/resources, create a file called **schema.sql** (right-click – New - SQL file):



Type in the following script to create a **Course** table:



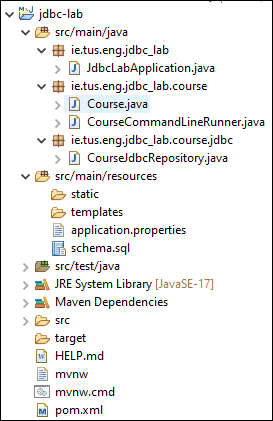
After you restart the server you should see the new table created:



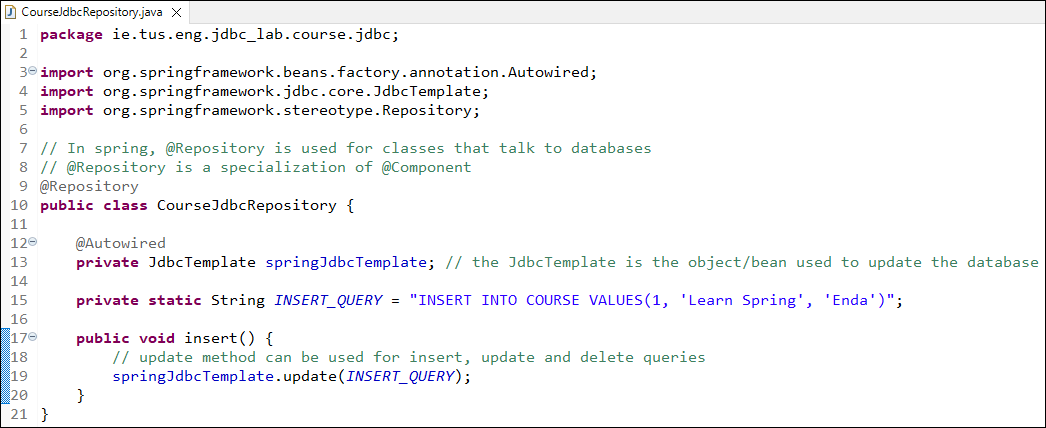
~~When you run “~~**~~Select \* from course~~**~~” there are no results as the table is empty. The next step is to add data to the table.~~

**~~Part 2 – Spring JDBC~~**

~~The project has the following structure. Create the Java files taking care to put them in the correct packages. The content and explanation of the Java files is given in the following pages.~~



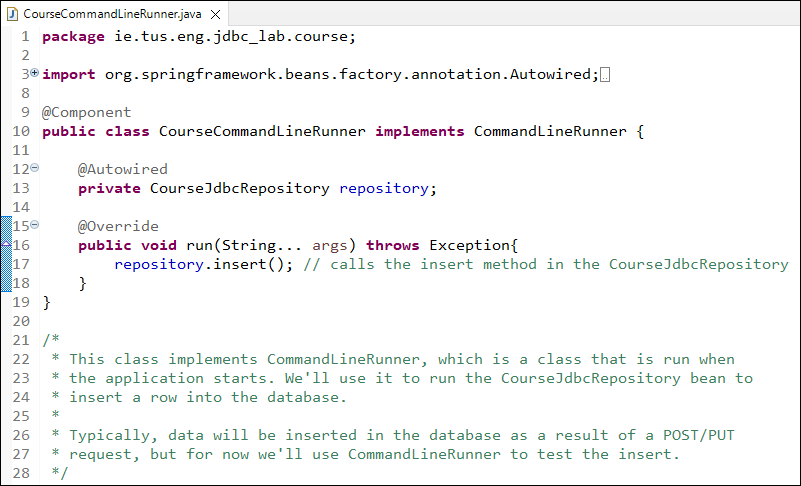
~~As a first example of how to insert something into the database, we’ll use the following class.~~



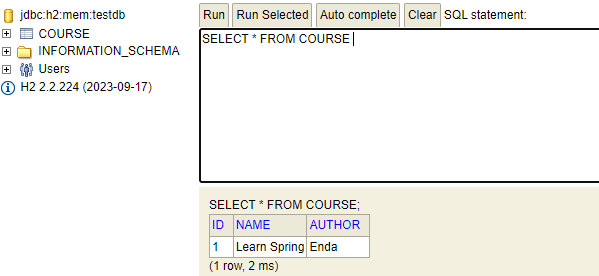
~~The @Repository annotation is used for classes that communicate with a database; similar to the DAO classes we used before.~~

~~As we’re using the spring-data-jdbc package, the springJdbcTemplate is automatically created and autowired into the class.~~

~~In order to test the class, we want to run it when Spring starts. For that we’ll use CommandLineRunner (Note the package):~~

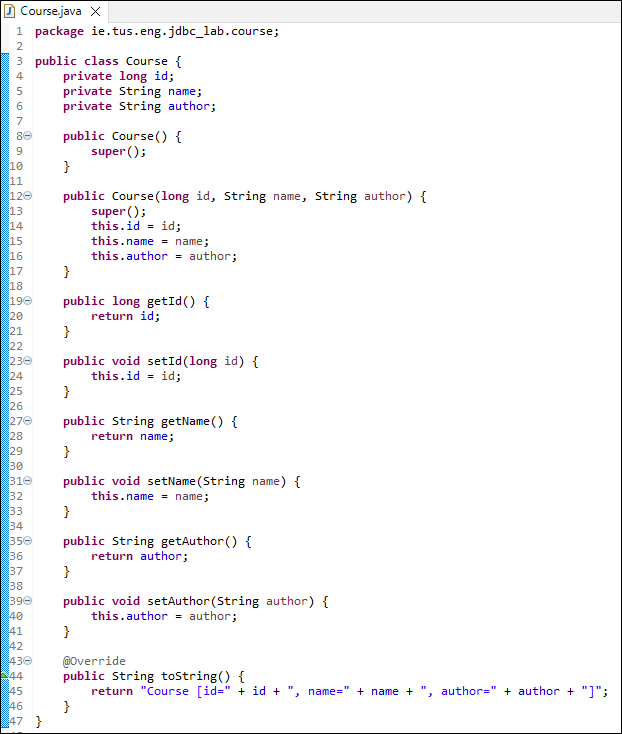


~~Now restart the application and check to see if the data was inserted into the database:~~

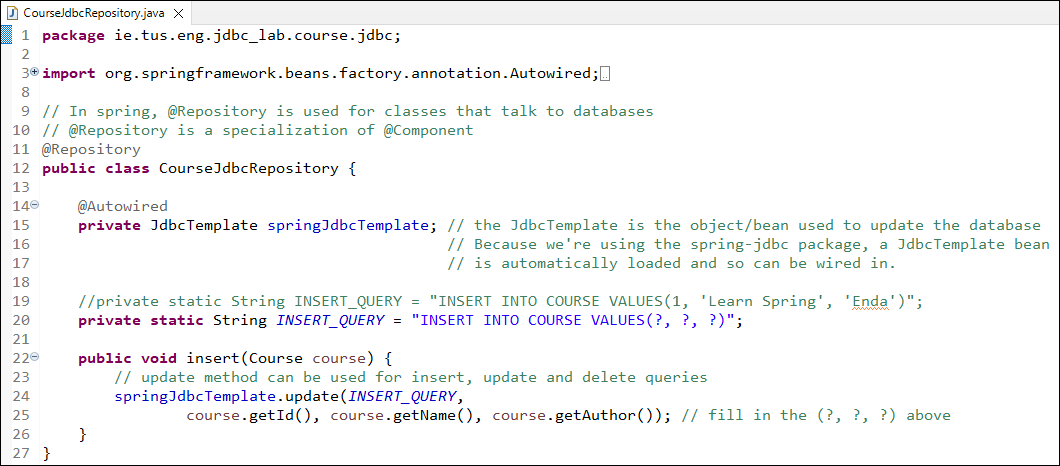


~~Next, instead of inserting a simple SQL string, we want to be able to insert a Course object.~~

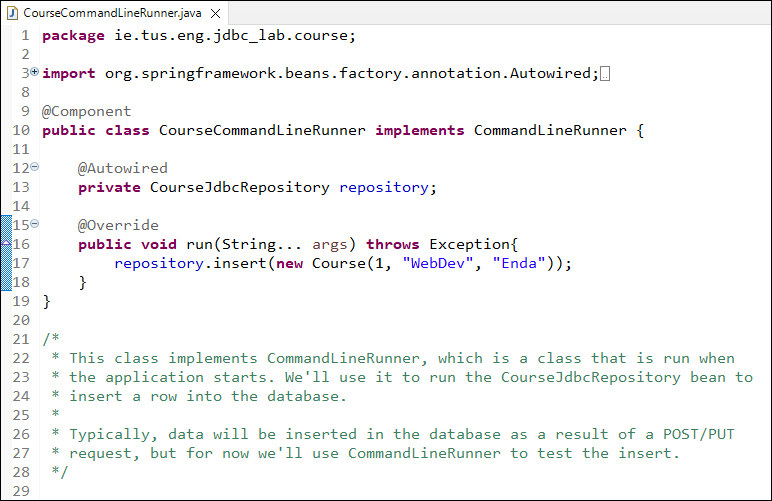
~~First make a POJO for the course: (Add all getters and setters and a toString() method)~~



~~Now make the following changes to the CourseJdbcRepository. We want to insert a Course object into the database:~~



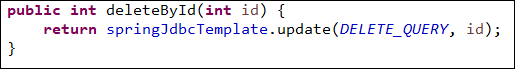
~~The CourseJdbcCommandLineRunner.java also needs to be changed. This will create a new object and use the repository to insert it.~~



~~If you restart the server the table should contain the line entered above.~~

**~~Exercise 1:~~**

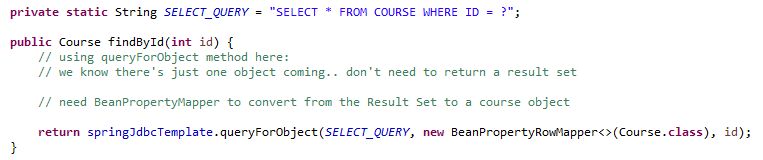
1. ~~Add a few more lines in the run method of the CourseJdbcCommandLineRunner to add more courses to the database.~~
2. ~~Write a method in CourseJdbcRepository.java to delete a course, given the id (deleteById).~~



1. ~~Add a line in the run method of the CourseCommandLineRunner to delete the course with id=2.~~

**Retrieving data:**

~~To retrieve data, i.e. SELECT command,~~ ***~~add~~*** ~~the following to the CourseJdbcRepository (don’t delete what’s already there!):~~

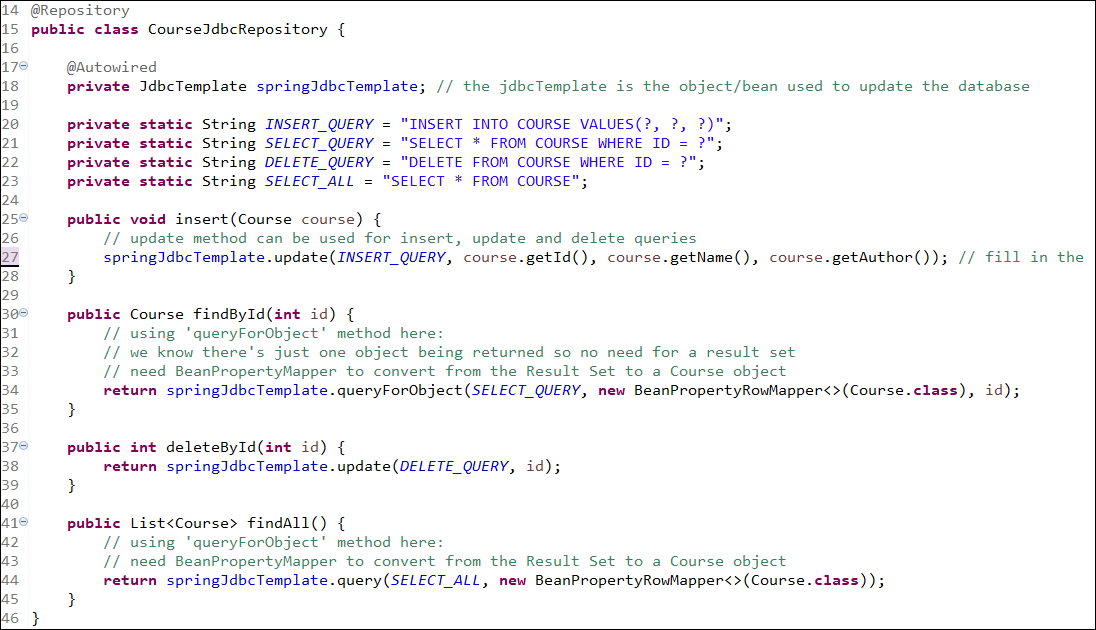


~~Add the following line to the CourseCommandLineRunner to find the record with id=4 in the database:~~

System.***out***.println(repository.findById(4));

~~Restart the server and you should see the 4~~~~th~~ ~~entry printed out (provided you added at least 4 entries).~~

~~To retrieve all the course objects, it’s similar. The code for the complete~~ **~~CourseJdbcRepository~~** ~~is as follows:~~



**Exercise:**

Revisit the ***rest-basics*** project we did before.

1. ~~Make a copy of the project and call it~~ **~~rest-basics-jdbc~~**
2. ~~Replace the~~ **~~StudentDao~~** ~~class with a~~ **~~StudentJdbcResource~~** ~~class.~~

~~i.e. The~~ **~~StudentResource~~** ~~class should use~~ **~~StudentJdbcResource~~** ~~instead of~~ **~~StudentDao~~** ~~– so the application is using a database instead of the HashMap.~~

1. ~~Use the steps above to create a suitable database.~~

Use Postman to test the API – the same tests as last week should work.