JAVA TECHNOLOGY

(503111)

LAB 7

EXERCISE 1

An exercise on initializing a Spring Boot project using the Spring Initializr tool and running the program using the Command line Runner.

Use the Spring Initializr tool provided online at https://start.spring.io/ to create a spring project according to the requirements below:

- Project type: maven
- Language: java
- Project Metadata: set as you like
- Packaging: Jar
- Java version: choose according to the JDK version on your computer
- Add the following dependencies: Lombok, Spring Data JPA, H2 Database
- Download the project as a zip, extract it, and open it with one of the supported programming tools: Intellij Idea, Visual Studio Code, Spring Tool Suite, etc.

In the DemoApplication.java file marked with @SpringBootApplication, create a method marked with @Bean to declare a bean for the CommandLineRunner class to run the spring boot program as a console, then print any message to the console.

In addition to the content specified by the System.out.println() command, the console also prints the Spring Boot logo and a lot of other log information as shown below.

```
2022-07-20 18:03:55.290  INFO 3206 --- [           main] o.hibernate.jpa.internal.util.LogHelper
2022-07-20 18:03:55.302  INFO 3206 --- [           main] org.hibernate.Version
2022-07-20 18:03:55.343  INFO 3206 --- [           main] o.hibernate.annotations.common.Version
2022-07-20 18:03:55.372  INFO 3206 --- [           main] org.hibernate.dialect.Dialect
2022-07-20 18:03:55.413  INFO 3206 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator
2022-07-20 18:03:55.416  INFO 3206 --- [           main] j.LocalContainerEntityManagerFactoryBean
2022-07-20 18:03:55.462  INFO 3206 --- [           main] com.example.demo.DemoApplication
Môn học Công nghệ Java
2022-07-20 18:03:55.465  INFO 3206 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2022-07-20 18:03:55.465  INFO 3206 --- [ionShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl
2022-07-20 18:03:55.466  INFO 3206 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2022-07-20 18:03:55.467  INFO 3206 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
```

Example output to console from CommandLineRunner

You are required to disable the Spring Boot logo in the console and change the setting to only display log messages with Warning level or higher. Hint: set the logging.level.root and spring.main.banner-mode properties in the application.properties file.

EXERCISE 2

Continue working on the project completed in the previous exercise.

Create a Student class that includes the id, name, age, email and ielts score attributes. Use annotations provided by Spring Data JPA to set the Student class as an Entity, set other information such as primary key, not null constraints, generate getter/setter/constructor methods automatically using annotations of the Lombok library.

Next, create an interface named StudentRepository to handle basic tasks such as adding, deleting, updating and reading a list of students from the database. StudentRepository needs to be marked with the @Repository annotation and needs to be extended from the interface named CrudRepository (provided in Spring Data Jpa).

In the application.properties file, you need to add the necessary properties for the application to connect and work with the H2 Database.

In the main program, declare and inject StudentRepository as a dependency then use it in Command Line Runner to perform operations:

- Add at least 3 students to the database
- Read the student list and print it to the console
- Update any student's information and print out the results after updating
- Delete a student from the database and print the result after deleting

EXERCISE 3

The data was stored in the H2 Database in the previous exercise, which is an in-memory database. Those data will be deleted automatically after the program ends. In this exercise, you are required to change the settings in the application.properties file so that the program works with a different database (MySQL or MS SQL Server, depending on which database you have on your computer). If you have trouble installing a database on your local machine, you can use any online database service, as long as you can complete this exercise.

EXERCISE 4

Continue to work with the project in the previous exercise, add more Query Methods to the StudentRepository class perform the following functions:

- Read a list of students whose age is greater than or equal to x, where x is the input parameter of the method.
- Count the number of students whose ielts score is equal to x, where x is an input parameter of the method.

- Find the list of students whose name contains the word xxx passed as a parameter. The string comparison method is case-insensitive.

Query Methods are methods defined in the StudentRepository interface. They have no body, they just need to be named according to a specific rule of Spring Data JPA then their function will be generated automatically.

Naming query methods according to Spring Data rules

KEYWORD	EXAMPLES	JPQL SNIPPET
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull	findByAgeIsNull	… where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	… where x.age not in ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false

EXERCISE 5

Although the query methods in the previous exercise have solved the requirements, the limitation is that the method names are often long and can't be named arbitrarily. In this exercise, you need to reimplement the methods in the previous exercise using the @Query annotation to set up the query, then you can use shorter names for the methods.

EXERCISE 6

Create a clone of the StudentRepository class and add sorting and paging functionality using the PagingAndSortingRepository interface. In the main method, fulfill the following requirements:

- Read the list of students, sorted in descending order by age. If there are more than one student of the same age, sort them in ascending order according to their ielts scores.
- Assuming the list has more than 10 students counting from 1-10, write a function to read the students 4-5-6 and print them to the console.