Spring Data Repositories

Overview

In the previous lab you defined a class named **`ProductSuggestionRepositoryImpl`**, to perform persistence logic for **`ProductSuggestion`** entities. You implemented several methods to query and modify entities via JPA.

In this lab you'll adopt an alternative approach to persistence. You'll define an interface named **`ProductSuggestionCrudRepository`** that will inherit persistence logic from the predefined **`CrudRepository`** interface. This is a lot easier than implementing all of the persistence logic yourself!

IntelliJ starter project

If you're happy to continue where you left off in the previous lab, use the following project:

- **`student\student-online-retailer`**

If you'd prefer a fresh start, use the solution project from the previous lab instead:

- **`solutions\solution-querying-modifying-entities`**

IntelliJ solution project

The solution project for this lab is located here:

- **`solutions\solution-spring-data-repositories`**

Roadmap

There are 3 exercises in this lab. Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Defining a “product suggestion” CRUD repository interface
2. Declaring additional modifier queries in the repository interface
3. Using the repository interface

Exercise 1: Defining a “product suggestion” CRUD repository interface

Define an interface named **`ProductSuggestionCrudRepository`** that extends **`CrudRepository`**. Specify two type parameters:

- The type of entities maintained by the repository, i.e., **`ProductSuggestion`**
- The type of the primary key in that entity, e.g., **`Long`**

Exercise 2: Declaring additional modifier queries in the repository interface

Add custom modifier queries to your **`ProductSuggestionCrudRepository`** interface, as follow:

- Modify the recommended price for a particular product suggestion
- Modify the estimated annual sales for a particular product suggestion
- Increase the price by 10% for all products that exceed a certain number of sales per year

You will have to annotate each of these methods with the following annotations:

- **`@Query`** (enables you to specify the JPQL query for the operation)
- **`@Modifying`** (this ensures the JPA entity manager flushes its cache of entities)
- **`@Transactional`** (runs the query in a transactional context, which is necessary)

Note that when you specify these modifier queries, they will return an **`int`** indicating the number of rows affected, which enables you to verify if the operation worked or not.

Exercise 3: Using the repository interface

Your application class current has code to test the **`ProductSuggestionRepositoryImpl`** class from the previous lab. Write some additional code to achieve the same effect using your **`ProductSuggestionCrudRepository`** interface (remember your interface inherits many useful methods from **`CrudRepository`**, such as **`save()`**, **`findById()`**, **`findAll()`**, etc.)