Query Creation with Spring Data JPA

1 Document Description

The exercises in this sheet continue from Lab Week 04 Part 2.

2 Find OrderType by Name

2.1 Define A New Abstract Method

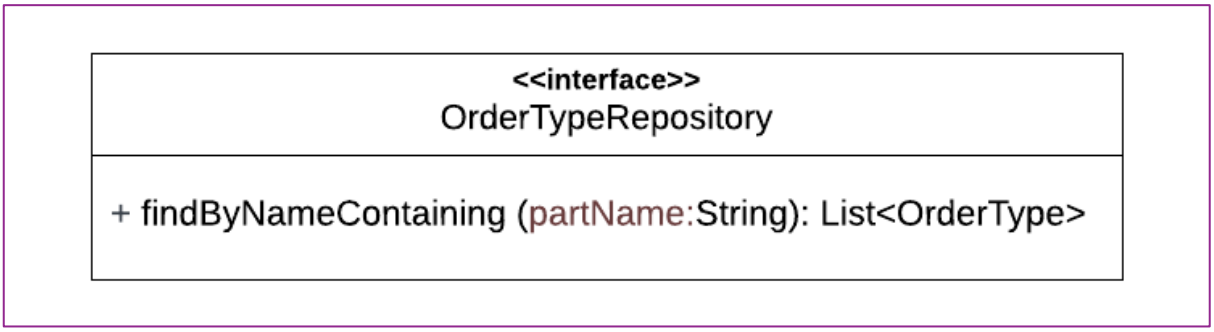

Figure 1: An abstract method to find order type by name

Figure 1 shows an update of the interface OrderTypeRepository. The updates define a new abstract method named findByNameContaining.

1. Double-click interface `OrderTypeRepository.java` from the **Project Explorer**.
2. Declare the method in Figure 1 in the interface.
3. Import the necessary class.
4. Save OrderTypeRepository.java.

2.2 Define A New Method and Mapping

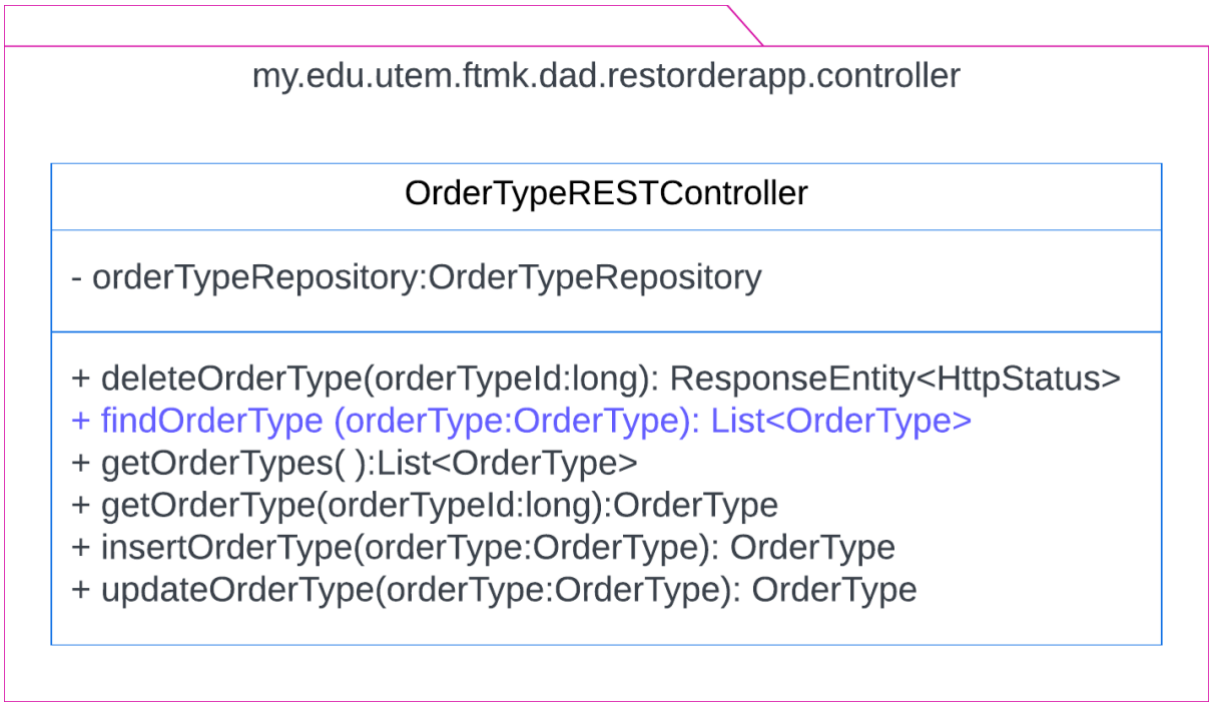

Figure 2: Definition of method findOrderType( ) in OrderTypeRestController

Figure 2 shows a new method, findOrderType( ), defined in OrderTypeRESTController. This method will find order type data according to the name specified in the attribute.

1. Double-click class `OrderTypeRESTController.java` from the **Project Explorer**.
2. Define the method findOrderType( ) as shown in Figure 2 in the class.
3. Annotate the method's parameter `@RequestBody`.
4. Annotate the method to `@RequestMapping`. Map the method to `"/find/name"`.
5. Implement the method body using the codes in Figure 3.

```
List<OrderType> orderTypes =
        orderTypeRepository.findByNameContaining(orderType.getName());

return orderTypes;
```

Figure 3: Implementation of findOrderType

6. Import the necessary class, if any.
7. Add a description to the method.
8. Save OrderTypeRESTController.java. The complete implementation of the method should be similar as shown in Figure 4

```
/**
 * This method find order type data according to the order type's name.
 *
 * @param orderType
 * @return A list of order types record.
 */
@RequestMapping("/find/name")
public List<OrderType> findOrderType (@RequestBody OrderType orderType) {

    List<OrderType> orderTypes =
            orderTypeRepository.findByNameContaining(orderType.getName());

    return orderTypes;

}
```

Figure 4: A complete implementation of the method findOrderType( )

2.3 Test Web Request to findOrderType

1. Open Postman
2. Create a new request in the REST OrderTypeRequests Collection.
3. Paste `http://localhost:8080/orderapp/api/ordertypes/find/name` in the request address bar.
4. Add the JSON data shown in Figure 5 to the requestor body.

```
{
    "orderTypeId": 0,
    "code": "",
    "name": "eat"
}
```

Figure 5: JSON data to find order type by name

5. After that, click the **Send** button. The response to the request should be similar, as shown in Figure 6.

Figure 6: A sample output from finding order type by name

6. Finally, click the **Save** button.
7. Give an appropriate name for the request.
8. Change the value of `name` in step 4 to "EAT".
9. Repeat steps 5 and 6.
10. Observe the output. Record the number of data from the response.
11. Repeat steps 8 to 10 using the following value for the `name`.
    a. "pick"
    b. "eat-in"

3 Find OrderType by Code

3.1 Define A New Abstract Method

1. Define a new abstract method named `findByCodeStartingWith( )` in the interface `OrderTypeRepository.java`. This method will find order type data according to a few characters at the beginning code.
2. Complete the definition of the method with the following specification.
    a. The method will have a string parameter that represents the partial code.
    b. The method will return a list of order type data.
3. Import the necessary class, if any.

4. Save `OrderTypeRepository.java`.

3.2 Define A New Method and Mapping

1. Define a new public method named `findOrderType` in `OrderTypeRESTController.java`.
2. Complete the definition of the method with the following specification.
    a. The method will receive a string parameter.
    b. Annotate the parameter with `@PathVariable`.
    c. The method will return a list of order type data.
    d. Annotate the method with `@RequestMapping`.
    e. Map it to `"/find/code/{parameter-name}"`.
    f. Replace the parameter-name with the parameter declared in step a.
    g. Invoke the method declared in step 1 of exercise 3.1 using the appropriate repository object.
    h. Complete the implementation with an appropriate return statement.
3. Add a suitable description of the method.
4. Import the necessary class, if any.
5. Save the class `OrderTypeRESTController.java`.

3.3 Test Web Request to findOrderType

1. Make a request from Postman for `http://localhost:8080/orderapp/api/ordertypes/find/code/G`.
2. Record your observation.
3. Change the parameter to "`dr`".
4. Repeat steps 1 and 2.
5. Repeat step 3 and 4 with the following parameters:-
    a. `adl`
    b. `PF`

4 Count Order Type Data

1. Define a method to count a list of order type data from the repository.
2. The method shall return the number of the order type data from the repository.
3. Annotate the method with an appropriate web annotation.
4. Map the method to a new URL.
5. Implement the method's body using the `count( )` method from the repository.
6. Describe the method appropriately.
7. Test the method using Postman.
8. Observe the output from the test.

5 Retrieve A Sorted Order Type Data

5.1 Define a New Abstract Method

1. Define a new abstract method named `findOrderByNameAsc( )` in the interface `OrderTypeRepository.java`.
2. This method does not receive any parameters.
3. The method will return a list of order type data.
4. Save the interface.

5.2 Define A New Method and Mapping

1. Define a new method in the class `OrderTypeRESTController.java`. This method retrieves a list of order type data sorted in ascending order.
2. Invoke the method defined in step 1 of 5.1 using an appropriate repository object.
3. This method will process a web request. Map the method appropriately.
4. Complete the implementation of the method.
5. Describe the method appropriately.
6. Save the class.

5.3 Test the Method to Retrieve A Sorted Order Type Data

1. Test the method defined in step 5.2 using Postman.
2. Observe the output from the test.

6 A Conclusion

1. What can you conclude from the relationship between interface `OrderTypeRepository.java` and class `OrderTypeRESTController.java`?

7 An Option for the Sorted Order Type

1. Create a web service method that allows the requestor to sort the order type either in ascending or descending order to the order type code.
2. Test the web service using Postman.

More exercises will be added soon. Stay tuned.
