(/)

Spring Data JPA - Métodos de eliminación derivados

Última modificación: 25 de marzo de 2019

> por baeldung (https://www.baeldung.com/author/baeldung/) (https://www.baeldung.com/author/baeldung/)

Primavera (https://www.baeldung.com/category/spring/) + Datos de primavera (https://www.baeldung.com/category/persistence/spring-persistence/spring-data/)

1. Introducción

Spring Data JPA nos permite definir métodos derivados (https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa) que leen, actualizan o eliminan registros de la base de datos. Esto es muy útil, ya que reduce el código repetitivo de la capa de acceso a datos.

En este tutorial, **nos centraremos en definir y utilizar los métodos de eliminación derivados de Spring Data** con ejemplos prácticos de código.

2. Derivado d *eleteBy* Methods

En primer lugar, vamos a configurar nuestro ejemplo. Definiremos una entidad de *frutas* para guardar el nombre y el color de los artículos disponibles en una tienda de frutas:

```
@Entity
public class Fruit {
    @Id
    private long id;
    private String name;
    private String color;
    // standard getters and setters
}
```

A continuación, agregaremos nuestro repositorio para operar en entidades de *frutas* , extendiendo la interfaz *JpaRepository* y agregando nuestros métodos derivados a esta clase.

Los métodos derivados pueden definirse como un atributo VERB + definido en una entidad. Algunos de los verbos permitidos son *findBy, deleteBy* y *removeBy* .

Derivemos un método para eliminar *Fruit* s por su *nombre* :

```
@Repository
public interface FruitRepository extends JpaRepository<Fruit, Long> {
    Long deleteByName(String name);
}
```

En este ejemplo, el método *deleteByName* devuelve el recuento de registros eliminados.

Del mismo modo, también podemos derivar un método de *eliminación* del formulario:

```
List<Fruit> deleteByColor(String color);
```

Aquí, el método *deleteByColor* elimina todas las frutas con un color dado y devuelve una lista de registros eliminados.

Vamos a probar los métodos de eliminación derivados. Primero, insertaremos algunos registros en la tabla de *frutas* , definiendo los datos en *test-fruit-data.sql:*

```
insert into fruit(id,name,color) values (1,'apple','red');
insert into fruit(id,name,color) values (2,'custard apple','green');
insert into fruit(id,name,color) values (3,'mango','yellow');
insert into fruit(id,name,color) values (4,'guava','green');
```

Luego, eliminaremos todas las frutas "verdes":

```
@Transactional
@Test
@Sql(scripts = { "/test-fruit-data.sql" })
public void givenFruits_WhenDeletedByColor_ThenDeletedFruitsShouldRetu
     List<Fruit> fruits = fruitRepository.deleteByColor("green");

     assertEquals("number of fruits are not matching", 2, fruits.size(
     fruits.forEach(fruit -> assertEquals("It's not a green fruit", "g
}
```

Además, tenga en cuenta que necesitamos usar la anotación *@Transactional* para los métodos de eliminación.

A continuación, agreguemos un caso de prueba similar para el segundo método *deleteBy* :

```
@Transactional
@Test
@Sql(scripts = { "/test-fruit-data.sql" })
public void givenFruits_WhenDeletedByName_ThenDeletedFruitCountShouldF

    Long deletedFruitCount = fruitRepository.deleteByName("apple");

    assertEquals("deleted fruit count is not matching", 1, deletedFru
}
```

3. Métodos de *eliminación* derivados

También podemos usar el verbo *removeBy* para derivar los métodos de eliminación:

```
Long removeByName(String name);
List<Fruit> removeByColor(String color);
```

Tenga en cuenta que no hay diferencia en el comportamiento de los dos tipos de métodos.

La *interfaz* final se verá como:

```
@Repository
public interface FruitRepository extends JpaRepository<Fruit, Long> {

    Long deleteByName(String name);

    List<Fruit> deleteByColor(String color);

    Long removeByName(String name);

    List<Fruit> removeByColor(String color);
}
```

Agreguemos pruebas unitarias similares para los métodos *removeBy* :

```
@Transactional
@Test
@Sql(scripts = { "/test-fruit-data.sql" })
public void givenFruits_WhenRemovedByColor_ThenDeletedFruitsShouldRetu
    List<Fruit> fruits = fruitRepository.removeByColor("green");

    assertEquals("number of fruits are not matching", 2, fruits.size()
}
```

```
@Transactional
@Test
@Sql(scripts = { "/test-fruit-data.sql" })
public void givenFruits_WhenRemovedByName_ThenDeletedFruitCountShouldF
    Long deletedFruitCount = fruitRepository.removeByName("apple");

    assertEquals("deleted fruit count is not matching", 1, deletedFru
}
```

4. Métodos eliminados derivados vs *@Query*

Es posible que nos encontremos con un escenario que hace que el nombre del método derivado sea demasiado grande o que implique un SQL JOIN entre entidades no relacionadas.

En este caso, también podemos utilizar las anotaciones *@Query* y *@Modifying (https://www.baeldung.com/spring-data-jpa-modifying-annotation)* para implementar operaciones de eliminación.

Veamos el código equivalente para nuestros métodos de eliminación derivados, utilizando una consulta personalizada:

```
@Modifying
@Query("delete from Fruit f where f.name=:name or f.color=:color")
List<Fruit> deleteFruits(@Param("name") String name, @Param("color") S
```

Aunque las dos soluciones parecen similares, y logran el mismo resultado, adoptan un enfoque ligeramente diferente. **El método *@Query* crea una única consulta JPQL contra la base de datos. En comparación, los métodos *deleteBy* ejecutan una consulta de lectura y luego eliminan cada uno de los elementos uno por uno.**

5. Conclusión

En este artículo, nos centramos en los métodos derivados de eliminación derivados de Spring Data. El código fuente completo utilizado en este artículo se puede encontrar en GitHub (https://github.com/eugenp/tutorials/tree/master/persistence-modules/spring-data-jpa-2) .

Deja una respuesta

Start the discussion...

Suscribir

LAS CATEGORÍAS

PRIMAVERA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
CLIENTE HTTP (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL DE JAVA "BACK TO BASICS" (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
TUTORIAL HTTPCLIENT 4 (/HTTPCLIENT-GUIDE)
DESCANSO CON TUTORIAL DE PRIMAVERA (/REST-WITH-SPRING-SERIES)
TUTORIAL DE PERSISTENCIA DE PRIMAVERA (/PERSISTENCE-WITH-SPRING-SERIES)
SEGURIDAD CON SPRING (/SECURITY-SPRING)

ACERCA DE

ACERCA DE BAELDUNG (/ABOUT)
LOS CURSOS (HTTPS://COURSES.BAELDUNG.COM)
TRABAJO DE CONSULTORÍA (/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
EL ARCHIVO COMPLETO (/FULL_ARCHIVE)
ESCRIBIR PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORES (/EDITORS)
NUESTROS COMPAÑEROS (/PARTNERS)
PUBLICIDAD EN BAELDUNG (/ADVERTISE)

TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)