Acceso a Datos

UT5 – ACCESO WEB CON SPRING BOOT

SPRING DATA JPA

1. Spring Data JPA

En el tema anterior *UT4-Herramientas de Mapeo Objeto Relacional (ORM),* vimos las tecnologías de Hibernate y JPA, y con ellas logramos reducir y simplificar el código de acceso a datos (comparado con lo que hacíamos únicamente con JDBC), obteniendo grandes beneficios.

Aun así, todavía tenemos gran cantidad de código repetitivo, por ejemplo, para cada entidad JPA debemos escribir las operaciones CRUD (crear, leer, actualizar, borrar). Spring Data genera automáticamente estas y otras operaciones que son de uso común, y además nos provee una manera simple para crear consultas y obtener los datos.

2. Consultas con Spring Data

Spring Data JPA ofrece varios mecanismos para realizar consultas:

- **Siguiendo una convención para el nombre del método**
- **Consultas JPQL**
- NamedQuery (ya vistas en el tema anterior)
- Consultas nativas SQL (ya vistas en el tema anterior**)**
- Consultas con QueryDSL
- Consultas con Query By Example
- Consultas con Criteria API
- …

2.1. Convención para el nombre del método

- Definimos métodos en nuestro repositorio. Dichos métodos no necesitan implementación (se la da Spring Data).

- El nombre del método debe seguir una serie de reglas para que de él se pueda derivar la consulta. El método tendrá el prefijo **`[find|read|query|get]`**…**`By`** y contendrá el nombre de los atributos por los que se quiera filtrar unidos por los operadores **`And, Or, Between, LessThan, GreaterThan`**…

2.1. Convención para el nombre del método

Por ejemplo:

```
List<Persona> findByLastname(String Lastname)
```

- Repositorio sobre Persona.
- Incluye una propiedad llamada Lastname.
- Queremos obtener aquellas Personas cuyo Lastname coincida con el que pasamos como argumento.

Y su traducción a lenguaje JPQL sería:

```
SELECT * FROM PERSON WHERE LASTNAME = '...';
```

2.1. Convención para el nombre del método

Además, se pueden usar el prefijo **`countBy`** para devolver el número de objetos que verifican los filtros definidos:

```
int countByLastname(String Lastname)
```

Este mecanismo es el más simple para hacer una query pero no nos servirá si tenemos muchos parámetros (el nombre del método será muy largo) o si la consulta es más compleja.

2.2. Consulta JPQL

Definiendo la query en **JPQL**. Se anota el método con **`@Query`** para definir consulta en JPQL. Los parámetros de esta consulta serán los parámetros que reciba el método.

```
@Query("select count(p) from Producto p where p.categoria = ?1")

public int findNumProductosByCategoria(Categoria categoria);
```

2.3. Limitación de los resultados de una consulta

First, Top

```
Persona find[Top|First]ByOrderByAgeDesc()
```

Podemos acompañarlo también de un número:

```
List<Person> find[Top|First]10ByOrderByAgeDesc()
```

Optional (Java 8) para aquellas consultas que devuelven un solo resultado y así evitar un posible NullPointerException

```
Optional<Person> find[Top|First]ByOrderByAgeDesc()
```

Dudas y preguntas