Asociaciones *many-to-one*

Elementos avanzados en tu API REST con Spring Boot

Modelo de datos

- No es lo mismo crear un controlador para una entidad sin asociaciones, que para una entidad asociada con otras entidades.
- Si la entidad en cuestión tiene asociaciones nos tenemos que preguntar qué hacer:
  - Al listar todos
  - Al listar uno
  - Al crear/editar
  - ...

Asociación Many To One

- Suele ser una de las asociaciones más frecuente.
- Diversas variantes (agregación, composición, ...)
- Asocia una instancia de una entidad con una varias instancias de otra.

Asociación Many To One

- Ejemplos
  - *Producto* → *Categoría*: un producto se asocia a una categoría, pero una categoría puede estar asociada a muchos productos
  - *Pedido* → *Cliente*: un pedido se asocia a un único cliente, pero un cliente puede realizar muchos pedidos.
  - *Población* → *Provincia*: una población pertenece a una provincia, pero una provincia tiene muchas poblaciones.

Modelos de respuesta

- Dada esta asociación, si nuestro controlador de obtener todos los productos devuelve un *List/Page* o un *Producto*, obtendremos todos los datos de *Producto* y de *Categoria*.


```
{
    "id": 33,
    "nombre": "Mussels - Frozen",
    "precio": 95,
    "imagen": "http://dummyimage.com/206x125.bmp/cc0000/ffffff",
    "categoria": {
        "id": 1,
        "nombre": "Comida"
    }
}
```

Modelos de respuesta

- Si a su vez, *Categoria* estuviera asociado con más objetos, nuestra respuesta podría ser *excesivamente grande*.

```
{
    ...
    "categoria": {
        "id": 7,
        "nombre": "Refrescos"
        "categoriaPadre": {
            "id": 1,
            "nombre": "Bebidas"
         }
    }
}
```

¿Qué hacer entonces?

- Deberíamos diferenciar entre nuestras entidades y los objetos que viajan en las peticiones/respuesta.
- De hecho, para los expertos, *no es buena práctica usar las entidades en los controladores, para que transporten la información a peticiones/respuestas.*
- Analizar, para cada petición, qué datos concretos queremos enviar/recibir.

Petición GET

- GET de TODOS/Consulta
  - *Normalmente* no requiere de una representación completa del objeto.
  - Evaluamos los atributos necesarios.
  - Seguramente, la mejor solución sería componer un *Data Transfer Object*.


```
{ "id": 10, "nombre": "Quail - Eggs, Fresh", "categoria": "Bebida"
  "imagen": "http://dummyimage.com/133x134.bmp/dddddd/000000" }
```

Petición GET

- GET por ID
  - Suele necesitar de una representación más completa del objeto.
  - Con todo, es posible que no necesite todos los atributos de la entidad (*createdBy*, *createdAt*, …)
  - Posiblemente, también necesitemos un DTO (diferente al anterior)

Petición POST

- Crear un nuevo recurso
  - El escenario puede ser variopinto
  - Depende mucho de los datos que estrictamente nos proporciona el cliente al hacer la petición.
  - Puede que existan datos derivados (i.e.: nombreCompleto = nombre + apellidos).
  - Lo más probable es que para una entidad con varias asociaciones necesitemos un nuevo DTO.

Petición PUT

- Editar un recurso existente
  - En la mayoría de situaciones, podremos usar o bien la entidad o bien el DTO de la petición POST.
  - Si algunos atributos no son modificables, o el proceso de modificación no es el habitual, posiblemente también necesitemos un nuevo DTO.

Petición DELETE

- Borrado de un recurso
  - No suele necesitar representación.
  - Si va bien, se suele devolver *204 No Content*.
  - Si en alguna circunstancia, queremos devolver el recurso recién borrado, podemos usar el DTO de GET por ID.

En nuestro código

- Ya tenemos una asociación *many-to-one* de Producto a Categoría.
- La forma de trasladar cada petición ya está implementada desde lecciones anteriores:
    - GET Todos: Con un DTO
    - GET por ID: Objeto completo.
    - POST: Con un DTO
    - PUT: Con un DTO
    - DELETE: Tan solo con el ID