

Formularios uno a muchos

cambios en nuestro ejemplo

Curso intermedio de Thymeleaf

Resultado esperado

Disponibilidad

Disponible Sin stock

Añadir oferta

Oferta n° 1

Nombre

Precio

0,0

Fecha Inicio **Fecha fin**

21/09/2019 00:00 22/09/2019 00:00

Oferta n° 2

Nombre

Precio

0,0

Fecha Inicio **Fecha fin**

21/09/2019 00:00 22/09/2019 00:00

Cambios (bastantes)

- Nueva clase Oferta, y refactorización de la clase Producto.

```
public class Oferta {  
  
    private Long id;  
    private String nombre;  
  
    private float nuevoPvp;  
    private float  
porcentajeDescuento;  
    private LocalDateTime  
fechaInicio;  
    private LocalDateTime  
fechaFin;  
    asociación bidireccional  
    private Producto producto;  
}
```

```
public class Producto {  
  
    private Long id;  
    private String nombre;  
    private String descripcion;  
private float descuento;  
    private String imagen;  
    private Disponibilidad  
disponibilidad;  
    private Categoria categoria;  
    private Set<Puntuacion>  
puntuaciones;  
    private List<Ofertas> ofertas;  
}
```

Cambios en el modelo

- Por tanto, también hay que añadir estos cambios en el esquema de la base de datos.
- En principio, no implementamos repositorio o servicio de *Ofertas* por simplicidad.

```
create table oferta (  
    id bigint not null,  
    nombre varchar(512),  
    nuevo_pvp float,  
    porcentaje_descuento  
float,  
    fecha_inicio timestamp,  
    fecha_fin timestamp,  
    producto_id bigint,  
    primary key (id)  
);
```

Cambios en el formulario

- Aquí es donde está el grueso del cambio de este ejemplo.
- Necesitamos modificar la plantilla, y añadir algunos métodos en el controlador y la plantilla.

Cambios en el formulario

- Nuestro command object, el *Producto*, tiene dentro un array de otro tipo de objeto, de tipo *Oferta*.
- Ese array es el que nos permite gestionar *los subformularios*.

Producto					
...					
List<Oferta> ofertas					
X					

Añadir oferta

Oferta nº 1 ✕

Nombre

Precio

0,0

Fecha Inicio

21/09/2019 00:00

Fecha fin

22/09/2019 00:00



Cambios en el formulario

- ¿Cómo conseguimos añadir un elemento nuevo sin enviar el formulario entero? Con algo de Javascript, AJAX y fragmentos.

1

Con AJAX, gestionamos el click sobre el botón de añadir. Cuando se produce, enviamos todo el contenido del formulario (serializado), añadiendo un nuevo parámetro llamado **addItem**.

Cambios en el formulario

- ¿Cómo conseguimos añadir un elemento nuevo sin enviar el formulario entero? Con algo de Javascript, AJAX y fragmentos.

2

Tenemos un nuevo método POST en el controlador, que sólo es invocado si la petición tiene el parámetro **addItem**.
Añadimos una nueva oferta, y comprobamos que la petición se ha hecho mediante AJAX para refrescar el fragmento donde se renderizan las ofertas.

Cambios en el formulario

- ¿Cómo conseguimos añadir un elemento nuevo sin enviar el formulario entero? Con algo de Javascript, AJAX y fragmentos.

3

Si revisamos de nuevo el código JS, veremos que si la petición tiene éxito, reemplazamos el contenido de la etiqueta `#ofertas` por este nuevo.

Cambios en el formulario

- El mecanismo para eliminar un elemento es similar.
 - Añadimos un parámetro, llamado `removeItem`, con el `id` (posición en el array) del elemento a eliminar.
 - El resto del mecanismo es idéntico al de añadir.

Cambios en el formulario

- ¿Cómo hacemos que un array de campos del formulario se rendericen para después enviarse? Hay que combinar
 - Lo que ya sabemos de bucles (th:each)
 - Manejo de arrays

Cambios en el formulario

- ¿Cómo hacemos que un array de campos del formulario se rendericen para después enviarse?

```
<div th:each="fila,stat : ${producto.ofertas}">
    ...
    Oferta nº <span th:text="${stat.count}"></span>
        <button name="eliminarOferta" th:value="${stat.index}"

                th:onclick="|javascript:remove_element(${stat.index})|">

            ...
            <input type="hidden"

                    th:field="${producto.ofertas[__${stat.index}__].producto.id}"

/>

    ...
    <label th:for="|ofertas__${stat.index}__.nombre|">Nombre</label>
    <input th:field="${producto.ofertas[__${stat.index}__].nombre}" />
```

Cambios en el servicio

- Añadimos un cambio menor en el servicio para que almacene correctamente los Productos y las Ofertas que se crean nuevos y de forma conjunta.

Cambios menores en la visualización

- Hemos cambiado la lógica de cómo hacer que un producto tenga un descuento.
- Modificamos la plantilla index, en la parte donde se visualiza el precio del producto.
- **Reto:** implementarlo en la plantilla de detalle de producto.