FORMULARIOS CON EDICIÓN Y BORRADO

BORRADO DE DATOS CON THYMELEAF

- Hasta ahora hemos insertado nuevos registros.
- ¿Cómo podemos borrar? Necesitamos lógica de negocio y algo de conocimientos sobre controladores, servicios y entidades.

BORRADO DE DATOS CON THYMELEAF

- Borrado de ***productos***
  - Diálogo de confirmación antes de borrar.
  - Sus puntuaciones se borrarán en *cascada*.
- Borrado de ***categorías***
  - Diálogo de confirmación antes de borrar.
  - Si la categoría tiene productos asociados, mostramos mensaje de error.

BORRADO DE DATOS CON THYMELEAF

- Borrado de ***productos***
  - Controlador

```
@GetMapping("/borrar/{id}")
public String borrarProducto(@PathVariable("id") Long id, Model model) {

    Producto producto = productoService.findById(id);

    if (producto != null) {
        productoService.delete(producto);
    }

    return "redirect:/admin/producto/";

}
```

Establecer la lógica de esta forma permite, en un futuro, añadir la visualización de mensajes de error.

  - Borrado en cascada

```
@OneToMany(mappedBy="producto", cascade=CascadeType.ALL, orphanRemoval=true, fetch=FetchType.EAGER)
private Set<Puntuacion> puntuaciones = new HashSet<Puntuacion>();
```

BORRADO DE DATOS CON THYMELEAF

- Borrado de ***categorías***
  - Controlador

```
@GetMapping("/borrar/{id}")
public String borrarCategoria(@PathVariable("id") Long id, Model model) {

    Categoria categoria = categoriaService.findById(id);

    if (categoria != null) {

        if (productoService.numeroProductosCategoria(categoria) == 0) {
            categoriaService.delete(categoria);
        } else {
            return "redirect:/admin/categoria/?error=true";
        }

    }

    return "redirect:/admin/categoria/";


}
```

Comprobamos que el número de productos de la categoría es 0 antes de borrarla. En caso contrario, redirigimos al listado de categorías, pero añadiendo un parámetro de error.

BORRADO DE DATOS CON THYMELEAF

- Borrado de ***categorías***
  - Plantilla

```
<div class="alert alert-warning alert-dismissible" role="alert"
 th:if="${param.error}">
    <button type="button" class="close" data-dismiss="alert"
        aria-label="Close">
        <span aria-hidden="true">&times;</span>
    </button>
    <span>No se puede borrar una categoria que tiene asociados productos</span>
</div>
```

El objeto *param* es un objeto especial, que tiene como propiedades los parámetros de la URL

Mostramos una alerta, indicando que no se pueden borrar categorías con productos asociados.

EDICIÓN DE DATOS CON THYMELEAF

- Podemos añadir los controladores necesarios, y *tunear* nuestros formularios, para poder reusarlos.
- Debemos añadir un campo al formulario que permita gestionar el ID del objeto a editar. Esta campo debe ser inmutable, o mejor, invisible.

```
<input type="hidden" th:field="*{id}" id="id" />
```

EDICIÓN DE DATOS CON THYMELEAF

- En función de si el ID tiene valor o no, podemos *tunear* el título del formulario.

```
<h1>
        <span th:text="${categoria.id} ? 'Editar' : 'Nueva'"></span>
                                categoría
</h1>
```

EDICIÓN DE DATOS CON THYMELEAF

- Controlador

(1) Enviar objeto al formulario	(2) Formulario	(3) Procesar los datos
- @GetMapping - Añadimos al modelo un *command object* ~~¿vacío?~~ **El objeto lo debemos obtener a partir de una consulta.** - Nos dirigimos a la plantilla del formulario.	- Tomamos del contexto el *command object*. - Asociamos cada uno de sus atributos a los campos correspondientes del formulario. - Dirigimos la acción del formulario hacia (3)	- @PostMapping - Recogemos el objeto enviado desde el formulario, ya relleno de datos. - Aplicamos la lógica de negocio correspondiente. - Redirigimos hacia otro controlador (¿de listado?)

EDICIÓN DE DATOS CON THYMELEAF

- Controlador

1

```
@GetMapping("/editar/{id}")
public String editarProducto(@PathVariable("id") Long id, Model model) {

    Producto producto = productoService.findById(id);

    if (producto != null) {
        model.addAttribute("producto", producto);
        model.addAttribute("categorias", categoriaService.findAll());
        return "admin/form-producto";
    } else {
        return "redirect:/admin/producto/";
    }

}
```

2 3

Para los pasos 2 y 3 reutilizamos el formulario y el controlador de un nuevo objeto.