Ejercicios Spring MVC con Thymeleaf

Ejercicio 1: Formulario de contacto

Objetivo: Practicar la creación de un formulario en Thymeleaf, manejar la recepción de datos en el controlador y mostrar una vista de confirmación.
Realizar el ejercicio implementado TDD

Requisitos:

1. Vista `contacto.html`:

- Un formulario con los siguientes campos:
  - Nombre (input de texto)
  - Email (input de tipo `email`)
  - Mensaje (textarea)
  - Botón de envío
- Usar `th:action` y `th:object` para enviar el formulario al backend.
- El formulario debe enviarse por `POST` al controlador.

2. Controlador:

- Clase `ContactoController`
- Método `mostrarFormulario()` que devuelve la vista `contacto`.
- Método `procesarFormulario()` que recibe el formulario con `@ModelAttribute`.
  - Guarda los datos en un objeto `Contacto` (puede ser un POJO con nombre, email, mensaje).
  - Agrega el objeto al modelo.
  - Redirige a una vista de confirmación.

3. Vista `confirmacion.html`:

- Mostrar un mensaje de agradecimiento personalizado con el nombre del usuario.
- Mostrar el mensaje enviado y el email.

Tips:

- Usar `@GetMapping("/contacto")` para el formulario.

- Usar `@PostMapping("/contacto")` para procesar el envío.

- No se necesita persistencia, solo mostrar cómo pasar los datos.

Ejercicio 2: Lista de productos y detalle

Objetivo: Mostrar una lista de productos en una vista, permitir al usuario hacer clic para ver el detalle de cada producto. Realizar el ejercicio implementado TDD

Requisitos:

1. Vista `productos.html`:

- Mostrar una tabla con una lista de productos (nombre, precio).
- Cada fila debe tener un botón o link para ver el detalle del producto.
- Usar Thymeleaf con `th:each` para iterar.

2. Controlador:

- Clase `ProductoController`.
- Método `listarProductos()`:
  - Devuelve una lista quemada en el controlador (ej., 3 productos hardcodeados en una lista).
  - Agrega esa lista al modelo.
  - Devuelve la vista `productos`.
- Método `verDetalle(@PathVariable Long id)`:
  - Busca el producto en la lista quemada por su `id`.
  - Lo agrega al modelo.
  - Devuelve la vista `detalle.html`.

3. Vista `detalle.html`:

- Mostrar el nombre, descripción y precio del producto seleccionado.
- Botón para volver a la lista (`/productos`).

Tips:

- Usar clases POJO para el producto (`id`, `nombre`, `descripcion`, `precio`).

- Usar rutas como `/productos` y `/productos/{id}`.

- Mostrar mensajes si no se encuentra el producto con ese ID (opcional).