

Modificación Listado Productos y Servicio



Modificación del Componente `ProductoLista` para Usar el Servicio HTTP en Angular

En esta guía, te explicaré cómo modificar el componente `ProductoLista` para que utilice el servicio HTTP `ProductoService` y realice peticiones al backend de Spring Boot.

También explicaremos por qué Angular hace peticiones HTTP al backend de Spring y cómo funciona el código actualizado.

1 ¿Por Qué Angular Hace Peticiones HTTP al Backend de Spring Boot?

En una aplicación moderna, ya sabemos que el frontend (Angular) y el backend (Spring Boot) trabajan juntos para ofrecer datos actualizados a los usuarios.

❖ ¿Por qué Angular no accede directamente a la base de datos?
Porque Angular se ejecuta en el navegador y no tiene acceso directo a la base de datos.

En su lugar, **hace peticiones HTTP al backend** en Spring Boot, el cual obtiene los datos y los envía a Angular.

2 Modificar `producto-lista.component.ts` para Consumir el Servicio HTTP

Abre `producto-lista.component.ts` y reemplázalo con el siguiente código:

```
import { Component, inject } from '@angular/core';
import { Producto } from '../producto';
import { ProductoService } from '../producto.service';

@Component({
  selector: 'app-producto-lista',
  templateUrl: './producto-lista.component.html'
})
export class ProductoListaComponent {
  productos: Producto[] = [];

  // Nueva forma de inyectar servicios
  private productoServicio = inject(ProductoService);

  ngOnInit() {
    //Cargamos los productos
    this.obtenerProductos();
  }

  private obtenerProductos(): void {
    this.productoServicio.obtenerProductosLista().subscribe({
      next: (datos) => {
        this.productos = datos;
      },
      error: (error) => {
        console.error("Error al obtener los productos", error);
      }
    });
  }
}
```

3 Explicación del Código

1. Importaciones necesarias

```
import { Component, inject } from '@angular/core';
import { Producto } from '../producto';
import { ProductoService } from '../producto.service';
```

- **Producto**: Importamos la clase que define el modelo de un producto.
- **ProductoService**: Importamos el servicio que realiza las peticiones HTTP.

2. Definición del componente

```
@Component({
  selector: 'app-producto-lista',
  templateUrl: './producto-lista.component.html'
})
```

- **selector: 'app-producto-lista'** → Define cómo se usará en HTML:
`<app-producto-lista></app-producto-lista>`.

3. Definir el array de productos y el servicio

```
productos: Producto[] = [];
private productoServicio = inject(ProductoService);
```

- **productos: Producto[] = []**; → Almacena la lista de productos que se mostrarán en la tabla.
- **private productoServicio = inject(ProductoService)**;
 - `inject(ProductoService)` → Nueva forma en Angular 19 de injectar servicios.

4. Cargar productos al iniciar el componente

```
ngOnInit(): void {
  this.obtenerProductos();
}
```

- **ngOnInit()** → Método que se ejecuta cuando el componente se carga.
- **Llama a obtenerProductos()** para obtener la lista de productos.

5. Método para obtener los productos

```
private obtenerProductos(): void {
  this.productoServicio.obtenerProductosLista().subscribe({
    next: (datos) => {
      this.productos = datos;
    },
    error: (error) => {
      console.error("Error al obtener los productos", error);
    }
});
```

{}

- o **obtenerProductos():**
 - Llama al servicio `ProductoService`.
 - Se suscribe al **Observable** devuelto por `obtenerProductosLista()`.
 - Cuando los datos llegan correctamente, los almacena en `productos`.
 - Si hay un error, lo muestra en la consola.
-

4 Mostrar la Lista de Productos en `producto-lista.component.html`

Esta modificación la realizaremos en el siguiente video para obtener dinámicamente los datos del listado de productos.

6 ¿Cómo Funciona la Comunicación Angular - Spring Boot?

- Angular **envía una petición HTTP** (GET) al backend en `http://localhost:8080/inventario-app/productos`
- Spring Boot responde con un **JSON** que contiene la lista de productos.
- Angular **recibe la respuesta** y actualiza la tabla con los datos.

Este enfoque permite que la aplicación **obtenga datos en tiempo real desde el backend**, en lugar de manejar datos estáticos.

7 ¿Qué Logramos con Esta Modificación?

- El componente `ProductoLista` ahora obtiene datos reales del backend.
 - Usamos el servicio HTTP para consumir la API de Spring Boot.
-  Ahora tienes un componente Angular completamente funcional que obtiene datos en tiempo real desde un backend en Spring Boot. 

Saludos!

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)