

Creación del Servicio de Producto



Creación del Servicio Producto en Angular para Consumir una API en Spring Boot

En esta guía, te mostraré cómo crear el servicio `ProductoService` en **Angular** para realizar peticiones al backend de **Spring Boot** a través de la URL:

`http://localhost:8080/inventario-app/productos`

Explicaré por qué se hacen así las peticiones desde el frontend en Angular al backend en Spring

1 Crear el Servicio de Producto

Ejecuta el siguiente comando en la terminal dentro de tu proyecto Angular:

```
ng g s producto --skip-tests
```

◆ Explicación del Comando

- `ng g s producto`: Genera un nuevo **servicio** llamado `ProductoService`.

- `g es de generate y s de service`
- `--skip-tests`: Omite la generación del archivo de pruebas (`producto.service.spec.ts`), ya que no lo necesitamos ahora.

Tras ejecutar este comando, se creará un archivo en:

`/src/app/producto.service.ts`

2 Configurar el Módulo HTTP en `app.config.ts`

Para que Angular pueda realizar peticiones HTTP, es necesario **habilitar el módulo HTTP** en las últimas versiones de **Angular**, ya que ahora los proyectos usan **componentes standalone**.

Abre `src/app/app.config.ts` y asegúrate de que contenga lo siguiente:

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideHttpClient()
  ],
};
```

◆ Explicación

- `provideHttpClient()`: Habilita el servicio `HttpClient`, necesario para hacer peticiones HTTP desde el frontend de Angular al backend de Spring Boot.
- En las últimas versiones de **Angular**, ya no usamos `HttpClientModule` en `app.module.ts`, sino que usamos `provideHttpClient()` en `app.config.ts`.

3 Crear y Configurar el Servicio `ProductoService`

Abre el archivo `producto.service.ts` y actualiza su contenido:

```
import {HttpClient} from '@angular/common/http';
```

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Producto } from './producto';

@Injectable({
  providedIn: 'root'
})
export class ProductoService {

  private urlBase = "http://localhost:8080/inventario-app/productos";

  constructor(private clienteHttp: HttpClient) { }

  obtenerProductosLista(): Observable<Producto[]>{
    return this.clienteHttp.get<Producto[]>(this.urlBase);
  }
}
```

◆ Explicación del Código

1. `@Injectable({ providedIn: 'root' })`
 - o Indica que el servicio estará disponible **globalmente** en la aplicación sin necesidad de importarlo manualmente en cada componente.
2. `private urlBase = "http://localhost:8080/inventario-app/productos";`
 - o Define la **URL base** del backend en Spring Boot.
 - o **Angular se comunica con Spring Boot a través de peticiones HTTP.**
3. `constructor(private clienteHttp: HttpClient) { }`
 - o HttpClient permite hacer peticiones HTTP (GET, POST, PUT, DELETE).
4. `obtenerProductosLista(): Observable<Producto[]>`
 - o Retorna un `Observable<Producto[]>`, lo que permite manejar la respuesta asincrónica del backend.
 - o `this.clienteHttp.get<Producto[]>(this.urlBase);`
 - Realiza una **petición GET** a la API en **Spring Boot** para obtener la lista de productos.

4 ¿Por qué Angular hace peticiones HTTP al Backend de Spring Boot?

Cuando construimos aplicaciones modernas con **frontend en Angular** y **backend en Spring Boot**, necesitamos que los dos sistemas **se comuniquen entre sí**:

◆ Angular (Frontend)

- Se ejecuta en el **navegador** del usuario.
- Se encarga de la **interfaz gráfica y la interacción con el usuario**.
- No almacena datos de forma persistente.

◆ Spring Boot (Backend)

- Se ejecuta en el **servidor**.
- Gestiona la **base de datos y la lógica de negocio**.
- Expone una **API REST** que permite que otras aplicaciones (como Angular) obtengan datos.

📌 ¿Cómo se comunican?

- Angular envía peticiones HTTP a la API en Spring Boot.
 - Spring responde con los datos solicitados.
 - Angular muestra la información en la interfaz del usuario.
-

5 Usar el Servicio en un Componente

Este paso se realizará en el siguiente video.

6 Mostrar los Datos en la Tabla (`producto-lista.component.html`)

Este paso se realizará en los siguientes videos.

Conclusión

- Creamos el servicio `ProductoService` para obtener datos desde Spring Boot.
- Configuramos `app.config.ts` para habilitar `HttpClient`.
- Explicamos por qué Angular hace peticiones HTTP al backend.

Ahora tienes una aplicación de **Angular** que está lista para **consumir datos de una API en Spring Boot** y poderlos mostrar más adelante en una **tabla de Bootstrap**. 

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx