Nest

Integración con Angular

CertiDevs

Índice de contenidos

1. Comunicación entre NestJS y Angular

La **comunicación** entre **NestJS (backend)** y **Angular (frontend)** generalmente se realiza a través de APIs REST utilizando solicitudes HTTP.

NestJS expone endpoints que retornan datos en formato JSON, mientras que Angular utiliza servicios para hacer solicitudes a estos endpoints y procesar las respuestas.

A continuación, se muestra un ejemplo de cómo implementar una comunicación básica entre NestJS y Angular:

2. Crear un endpoint en NestJS

Primero, crea un **controlador** y un **servicio** en NestJS para manejar las solicitudes y generar una respuesta.

Archivo `data.controller.ts`:

```
import { Controller, Get } from '@nestjs/common';
import { DataService } from './data.service';

@Controller('data')
export class DataController {
    constructor(private readonly dataService: DataService) {}

    @Get()
    getData() {
        return this.dataService.getData();
    }
}
```

Archivo `data.service.ts`:

```
import { Injectable } from '@nestjs/common';

@Injectable()
export class DataService {
    getData() {
        return [
            { id: 1, name: 'Item 1' },
            { id: 2, name: 'Item 2' },
        ];
    }
}
```

En este ejemplo, el controlador `DataController` tiene un solo endpoint `GET /data` que devuelve un arreglo de objetos.

3. Crear un servicio en Angular para consumir el endpoint

En Angular, crea un **servicio** que utilizará `HttpClient` para hacer solicitudes a los endpoints expuestos por NestJS.

Archivo `data.service.ts`:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
    providedIn: 'root',
})
export class DataService {
    private apiUrl = 'http://localhost:3000/data';

    constructor(private http: HttpClient) {}

    getData(): Observable<any> {
        return this.http.get(this.apiUrl);
    }
}
```

Este **servicio** define un método `getData` que realiza una solicitud **GET** al endpoint `/data` de NestJS.

4. Consumir el servicio en un componente de Angular

Para consumir el servicio en un componente de Angular, primero inyéctalo en el **constructor** y luego llama al método `getData` para obtener los datos desde el backend.

Archivo `app.component.ts`:

```
import { Component, OnInit } from '@angular/core';
import { DataService } from './data.service';

@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css'],
})
export class AppComponent implements OnInit {
    data: any[];
```

```
    constructor(private dataService: DataService) {}

    ngOnInit() {
        this.dataService.getData().subscribe((response) => {
            this.data = response;
        });
    }
}
```

Archivo `app.component.html`:

```
<ul>
    <li *ngFor="let item of data">{{ item.name }}</li>
</ul>
```

En este ejemplo, el componente `AppComponent` consume el servicio `DataService`, obtiene los datos del endpoint en NestJS y los muestra en una lista en la plantilla.

Con estos pasos, has implementado una comunicación básica entre NestJS y Angular utilizando solicitudes HTTP y APIs REST.

El backend de NestJS expone endpoints para retornar datos, mientras que el frontend de Angular utiliza servicios para hacer solicitudes a estos endpoints y procesar las respuestas.