Acabamos de ver como usar Java equals y hashcode

Nestjs es un framework desarrollado completamente en TypeScript para Node,js con un enfoque de lado servidor .Este framework copia muchas de las ideas que tiene detrás Spring como Framework clásico . ¿Qué es lo que aporta NestJS? . Desde mi punto de vista aporta un primer punto de partida a la hora de madurar muchos de los desarrollos que se realizan con Node.js . Node.js siempre me ha parecido una plataforma con mucho futuro pero que a veces le faltaba algo de orden y concierto a la hora de construir soluciones sólidas y extensibles. La llegada de TypeScript como lenguaje de programación ha supuesto un salto importante ya que nos acerca JavaScript a las capacidades que tiene Java o C# como lenguaje compilado.

Ahora bien todos somos conscientes que Java es mucho más que un lenguaje compilado y multiplataforma . Java probablemente es la tecnología más sólida de lado de servidor a la hora de construir aplicaciones empresariales. En este tipo de aplicaciones Spring Framework destaca de una forma clara y desde mi punto de vista supera a los standares de Java EE en el uso cotidiano.

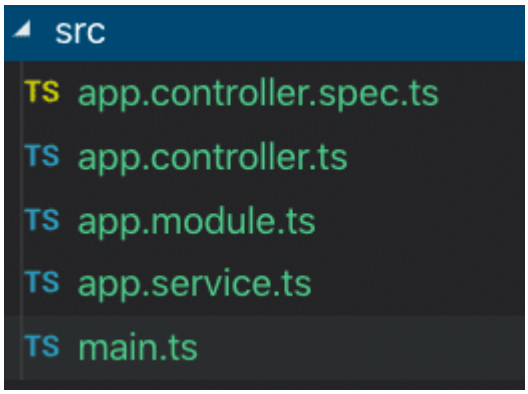
Así pues que aparezca un framework en el mundo de Node.js que se asemeje a Spring Framework es algo que tendremos que tener en cuanta ya que puede que parte del futuro de Node y las aplicaciones de lado Servidor pasen por algo del estilo a NestJS. Vamos a ver como construir el ejemplo de hola mundo con este framework y como sus anotaciones y divisiones de responsabilidades se acercan mucho a todo lo que conocemos con Spring. Lo primero que tendremos que hacer es instalar NestJS .

npm i -g @nestjs/cli

Esta instrucción nos recuerda mucho al mundo de Angular ya que nos instala un cliente con el cual se pueden hacer las distintas operaciones. Una vez realizada esta primera parte y contando ya con el cliente de Nest es momento de ejecutar un comando para construir la aplicación:

nest new miproyecto

Este comando nos genera una aplicación de HolaMundo con Nest. La aplicación contiene un Controller y un Servicio de ejemplo para que nos hagamos una idea de su funcionamiento así como la definición de módulo.



Si revisamos el código del controlador y del servicio nos sorprenderemos de lo similar que es al propio framework Spring.

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

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
```

```
  getHello(): string {
    return this.appService.getHello();
  }
}
```

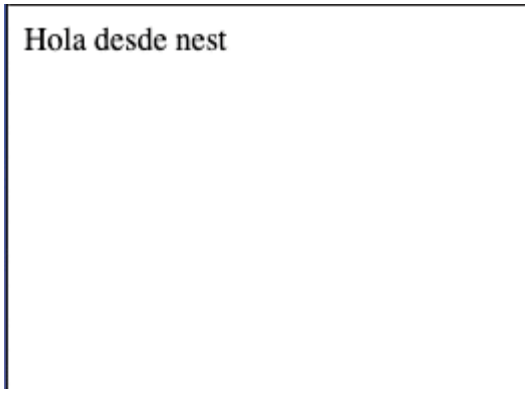
```
import { Injectable } from '@nestjs/common';

@Injectable()
export class AppService {
  getHello(): string {
    return 'Hola desde nest';
  }
}
```

NestJS y su código

El servicio contiene un método de getHello() y esta anotado con @Injectable que es la anotación clásica de Angular a la hora de dar de alta servicios. Su parecido con la anotación @Service de Spring es clara. Por otro lado tenemos un @Controller en este caso estamos ante una situación idéntica a la de Spring. Sus similitudes son más que claras. TypeScript ya soporta frameworks de ORM Mapping como por ejemplo:TypeORM. Es momento de lanzar la aplicación

npm run start

Esto levantará un servidor web en el puerto 3000 y accederemos a la url por defecto que nos muestra el mensaje de Hola desde nest.



Todo ha sido muy sencillo casi tan sencillo como usar Spring Boot. Recordemos que TypeScript se puede ejecutar en cliente y en servidor y poco a poco comenzará a competir de forma más fuerte con NET y Java . Hoy por hoy la pregunta más difícil es si TypeScript terminará substituyendo a estos lenguajes. Hoy por hoy a mi personalmente se me hace algo lejano . Es cierto que en la capa de FrontEnd terminará liderando el mercado . Pero en el lado servidor han salido tantas tecnologías y tantos frameworks que se interrelacionan unos con otros que pienso personalmente que falta mucho tiempo para ver algo tan maduro como Java en el mundo de TypeScript

Otros Articulos Relacionados:

1. TypeScript vs Java , NET y la ubicuidad
2. Mi Nuevo Curso de Typescript
3. TypeScript Union Type y como usarlos