¿Qué es la inyección de dependencias? 13/20

RECURSOS MARCADORES

Es muy sencillo crear un servicio en Angular, inyectarlo en un componente y utilizar su lógica. Pero siempre es recomendable entender **¿qué es la inyección de dependencias?, cómo se está haciendo y qué sucede detrás en tu aplicación.

Patrones de diseño

Angular usa varios patrones de diseño para permitir que esto funcione.

Inyección de dependencias

Imagínate que tienes el siguiente panorama:
Un **Servicio A** que emplea el **Servicio B** y este a su vez utiliza el **Servicio C**.

Si tuvieses que instanciar el Servicio A, primero deberías:
instanciar el C para poder continuar con el B y luego sí hacerlo con el A. Se vuelve confuso y poco escalable si en algún momento también tienes que instanciar el Servicio D o E.

La inyección de dependencias soluciona las dependencias de una clase por nosotros.

Cuando instanciamos en el constructor el servicio A, Angular por detrás genera automáticamente la instancia del servicio B y C sin que nosotros nos tengamos que preocupar por estos.

Cuando creaste tu primer servicio con el CLI de Angular:

```
// services/test-name.service.ts
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: 'root'
})
export class TestNameService {
  constructor() { }
}
```

Este le proporcionó a la clase el decorador `@Injectable({ ... })` con el valor `providedIn: 'root'` que determina el *scope* del servicio, o sea, determina que el mismo estará disponible en toda el módulo de tu aplicación por default.

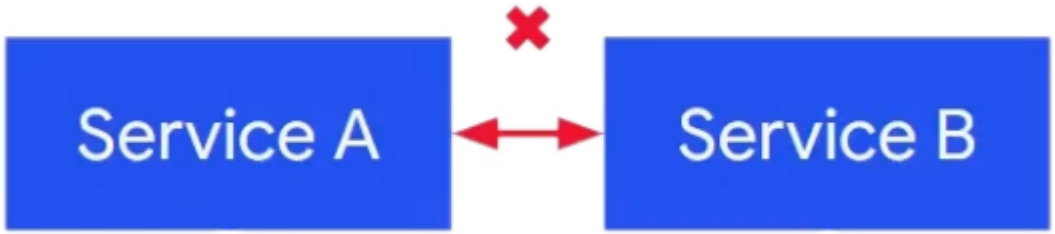
Singleton

La inyección de dependencias no es el único patrón de diseño que Angular usa con sus servicios. También hace uso del **patrón Singleton** para crear una instancia única de cada servicio.

Si tienes un servicio que se utiliza en N cantidad de componentes (u otros servicios), todos estarán utilizando la misma instancia del servicio y compartiendo el valor de sus variables y todo su estado.

Precauciones utilizando servicios

Ya has visto hasta aquí que un servicio puede ser importado en muchos componentes u otros servicios a la vez. Puedes inyectar la cantidad de servicio que quieras en un componente, siempre de una forma controlada y coherente.



Solo debes tener cuidado con las dependencias circulares. Cuando un servicio importa a otro y este al anterior. Angular no sabrá si vino primero el huevo o la gallina y tendrás un error al momento de compilar tu aplicación.

Contribución creada con los aportes de Kevin Fiorentino.

Lecturas recomendadas

Patrones de diseño: Singleton
https://platzi.com/clases/1630-mejor-codigo/22212-patrones-de-diseno-singleton/