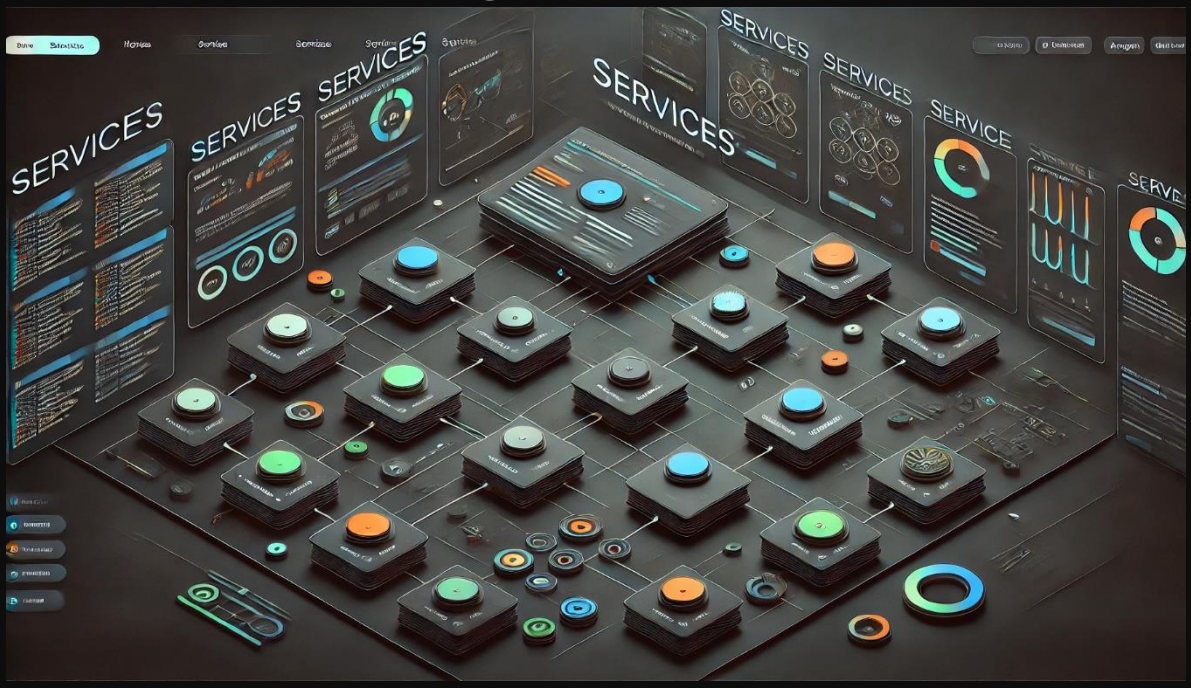
Servicios en Angular



Servicios en Angular

¿Qué es un servicio en Angular?

En Angular, un **servicio** es una clase que se utiliza para encapsular y organizar lógica que puede ser reutilizada a lo largo de varios componentes de la aplicación. Los servicios permiten compartir datos, realizar tareas complejas como la comunicación con APIs o la manipulación de datos sin duplicar lógica en múltiples lugares.

Una **API** (Application Programming Interface, o Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos que permite que diferentes aplicaciones o servicios se comuniquen entre sí. Las API sirven como un "puente" que facilita el intercambio de datos y la ejecución de funciones entre sistemas, sin que estos tengan que conocer los detalles internos de cada uno.

Los servicios son fundamentales para mantener la aplicación limpia, modular y aplican el principio de **separación de responsabilidades**, ya que los componentes se centran en la presentación y delegan la lógica de negocio a los servicios.

Creación de un servicio

Para crear un servicio en Angular, se utiliza el comando del CLI de Angular:

```
ng generate service nombre-del-servicio
```

O de manera resumida, el comando queda así:

```
ng g s nombre-del-servicio
```

Este comando generará una clase de servicio que se puede inyectar en los componentes o en otros servicios para su uso.

Sintaxis básica de un servicio

Un servicio básico de Angular es una clase decorada con `@Injectable()`. El decorador `@Injectable()` indica que el servicio puede ser inyectado a otros elementos de la aplicación a través de la **inyección de dependencias**.

La **inyección de dependencias** (Dependency Injection, DI) es un patrón de diseño que permite que los objetos reciban sus dependencias de fuentes externas, como otras clases, en lugar de crearlas por sí mismos. Este patrón es clave para la modularidad y mantenimiento del código en frameworks como Angular, pero no es exclusivo de Angular.

Ejemplo de un servicio básico

Creación del servicio de mensaje.service.ts con CLI:

```
ng g s mensaje --skip-tests
```

Código del servicio de mensaje-servicio.service.ts:

```
import { Injectable } from '@angular/core';

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

  private mensaje: string = 'Hola desde el servicio de Mensajes!';

  obtenerMensaje(): string {
    return this.mensaje;
  }
}
```

En este ejemplo, `MensajeService` tiene un método `obtenerMensaje` que devuelve una cadena de texto. El servicio está decorado con `@Injectable` y tiene un `providedIn: 'root'`, lo que indica que el servicio es **singleton** a nivel de toda la aplicación y será inyectado automáticamente cuando sea necesario.

S**ingleton** es un patrón de diseño que asegura que una clase solo tenga **una única instancia** en todo el ciclo de vida de una aplicación, y que esta instancia esté disponible en todo

momento. Es decir, no importa cuántas veces se intente instanciar la clase, siempre se reutilizará la misma instancia creada inicialmente.

Uso de un servicio en un componente

Para utilizar el servicio en un componente, simplemente se inyecta en el constructor del componente.

Ejemplo de inyección de servicio en un componente

```
export class AppComponent {
  titulo = 'Servicios en Angular';

  mensaje: string;

  constructor(private mensajeService: MensajeService) {
    this.mensaje = this.mensajeService.obtenerMensaje();
  }

}
```

En este componente, el servicio `MensajeService` se inyecta en el constructor y luego se llama al método `obtenerMensaje` para obtener el mensaje almacenado en el servicio, el cual se muestra en la plantilla del componente.

¿Qué es `providedIn: 'root'`?

La opción `providedIn: 'root'` indica que el servicio estará disponible en toda la aplicación sin necesidad de declararlo en los módulos manualmente. Esto convierte al servicio en un **singleton**.

Por último, utilizamos el servicio desde app.component:

Código app.component.html:

```
<section>
    <div class="container text-center">
        <h1 class="text-center text-warning my-4">{{ titulo }}</h1>
        <p>{{ mensaje }}</p>
    </div>
</section>
```

Código app.component.ts:

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
```

```
import { NuevoComponenteComponent } from './nuevo-componente/nuevo-componente.component';
import { ComponenteInlineComponent } from './componente-inline/componente-
inline.component';
import { PadreComponent } from './padre/padre.component';
import { MostrarMensajeComponent } from "./mostrar-mensaje/mostrar-mensaje.component";
import { ReplicadorComponent } from "./replicador/replicador.component";
import { SaludarComponent } from "./saludar/saludar.component";
import { ComponenteIfComponent } from "./componente-if/componente-if.component";
import { ComponenteForComponent } from "./componente-for/componente-for.component";
import { AgregarTareaComponent } from "./agregar-tarea/agregar-tarea.component";
import { ViewchildComponent } from "./viewchild/viewchild.component";
import { MensajeService } from './mensaje.service';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [
    RouterOutlet,
    NuevoComponenteComponent,
    ComponenteInlineComponent,
    PadreComponent,
    MostrarMensajeComponent,
    ReplicadorComponent,
    SaludarComponent,
    ComponenteIfComponent,
    ComponenteForComponent,
    AgregarTareaComponent,
    ViewchildComponent
],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css',
})
export class AppComponent {
  titulo = 'Servicios en Angular';

  mensaje: string;

  constructor(private mensajeService: MensajeService) {
    this.mensaje = this.mensajeService.obtenerMensaje();
  }

}
```

Aquí lo importante es el constructor de la clase AppComponent. En el cual podemos ver la inyección de dependencias para usar el servicio MensajeService.

Resultado de Ejecutar la aplicación de Servicios:

Si prefieres controlar el alcance del servicio y solo usarlo en un módulo específico, puedes registrarlo en el `providers` de dicho módulo en lugar de usar `providedIn`.

Resumen y puntos clave

- Los **servicios** permiten encapsular lógica reutilizable y compartirla entre múltiples componentes.
- Se crean usando la clase `@Injectable()` y se inyectan en componentes u otros servicios.
- **Inyección de dependencias** permite que Angular maneje la creación y el ciclo de vida de los servicios.
- La opción `providedIn: 'root'` hace que el servicio esté disponible en toda la aplicación como un singleton.
- Los servicios son ideales para manejar operaciones como llamadas a APIs, manipulación de datos o cualquier otra lógica de negocio que no esté directamente relacionada con la vista.

En la siguiente lección vamos a profundizar en el tema de Servicios en Angular.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx