Servicios y Observables en Angular

Servicios y Observables en Angular

Como ya hemos visto, los **servicios** en Angular son clases que encapsulan lógica o funcionalidad que se puede compartir entre diferentes componentes. El propósito principal de un servicio es facilitar la **separación de responsabilidades** en una aplicación Angular.

¿Qué es un Observable en Angular?

Un **Observable** es una herramienta fundamental en Angular para manejar flujos de datos asíncronos. Un Observable puede emitir múltiples valores o eventos a lo largo del tiempo. Los Observables son parte de la librería **RxJS**, que Angular utiliza para manejar datos asíncronos como solicitudes HTTP, eventos del DOM o temporizadores.

Cuando los servicios en Angular interactúan con APIs o realizan tareas asincrónicas, suelen utilizar **Observables** para manejar los datos.

Relación entre Servicios y Observables

Los servicios son el lugar ideal para manejar la lógica de negocio de tu aplicación, como las llamadas a una API. Los **Observables** permiten que los servicios expongan datos que se pueden **suscribir** en los componentes, para que estos reaccionen cuando esos datos están disponibles. El patrón de uso típico es que un Servicio emita los datos a través de un

Observable, y los componentes se suscriban a esos datos para mostrar los resultados en la interfaz de usuario.

Sintaxis básica de un servicio con Observable

Para crear un servicio en Angular que utilice Observables, sigue estos pasos básicos:

1. Define el servicio utilizando el decorador `@Injectable()`.
2. Usa **`HttpClient`** para hacer solicitudes HTTP y devuelve los datos como un Observable.
3. En el componente que consume el servicio, suscríbete al Observable para obtener los datos emitidos.

Ejemplo de Servicio con Observable de tipo HTTP

Vamos a crear un servicio para mostrar una lista de usuarios de un servicio externo (API). Ustedes pueden usar cualquier otra API de prueba que deseen, en el ejemplo utilizaremos el siguiente sitio web:

https://jsonplaceholder.typicode.com/users

Creamos ahora un servicio para poder crear un Observable a partir de los datos que arroja la petición http de tipo GET:

```
ng g s usuarios --skip-tests
```

Código usuarios.service.ts:

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

@Injectable({
  providedIn: 'root'
})
export class UsuariosService {
  private apiUrl = 'https://jsonplaceholder.typicode.com/users';

  constructor(private http: HttpClient) {}

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

En este ejemplo, el servicio `UsuariosService` utiliza `HttpClient` para realizar una solicitud GET a una API. El método `obtenerDatos` devuelve un `Observable`, lo que permite manejar de manera asíncrona la respuesta de la API.

Uso de `UsuarioService` en un componente

Creamos el componente de listado de usuarios:

```
ng g c listado-usuarios --skip-tests
```

Código listado-usuarios.component.html:

```
<div class="row justify-content-center">
    <div class="col-md-6">
        <ul class="list-group">
            @if (usuarios.length > 0) {
            @for (usuario of usuarios; track usuario.id) {
            <li class="list-group-item">{{ usuario.id }} {{ usuario.name }}</li>
            }
            } @else {
            <li class="list-group-item text-danger">No hay usuarios disponibles.</li>
            }
        </ul>
    </div>
</div>
```

Código listado-usuarios.component.ts:

```
import { Component } from '@angular/core';
import { UsuariosService } from '../usuarios.service';

@Component({
  selector: 'app-listado-usuarios',
  standalone: true,
  imports: [],
  templateUrl: './listado-usuarios.component.html',
  styleUrl: './listado-usuarios.component.css'
})
export class ListadoProductosComponent {
  usuarios: any[] = [];

  constructor(private usuariosService: UsuariosService) {}

  ngOnInit(): void {
    this.usuariosService.obtenerDatos().subscribe((data) => {
      this.usuarios = data;
    });
  }
}
```

En este componente, cuando se inicializa (`ngOnInit`), se llama al método `obtenerDatos` del servicio `UsuariosService`, que realiza la solicitud HTTP. La respuesta se suscribe utilizando el método `subscribe`, y los datos se almacenan en el componente para ser mostrados en una lista (`<ul>`).

Modificamos `AppComponent`:

Código app.component.html:

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

Código app.component.html:

```
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';
import { ListadoProductosComponent } from "./listado-usuarios/listado-
usuarios.component";

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

Finalmente, para que todo funcione correctamente, es necesario registrar el proveedor HttpCliente. Debido a que estamos usando componentes standalone, entonces no existe el archivo app.module.ts. Por lo tanto, los servicios los vamos a registrar directamente en el archivo main.ts. El archivo se debe modificar y queda como sigue:

```
import { bootstrapApplication } from '@angular/platform-browser';
import { provideHttpClient } from '@angular/common/http';
```

```
import { appConfig } from './app/app.config';
import { AppComponent } from './app/app.component';

bootstrapApplication(AppComponent, {
  ...appConfig,  // Propagamos el appConfig
  providers: [
    provideHttpClient(),
    ...appConfig.providers  // Propagamos los providers del appConfig
  ]
}).catch((err) => console.error(err));
```

Recordar:

El operador `...` que ves en el código de TypeScript es conocido como el **operador de propagación** (**spread operator**). Este operador es parte de ECMAScript 2015 (ES6) y se utiliza para "expandir" elementos como **objetos** o **arrays**.

`...appConfig`:

- **Propósito**: Está copiando y expandiendo todas las propiedades del objeto `appConfig` dentro del objeto que se está pasando a la función `bootstrapApplication`.

Resultado Final:

Resumen

- **Servicios** en Angular encapsulan lógica y permiten compartir datos entre componentes.
- **Observables** son esenciales para manejar datos asíncronos en Angular, como solicitudes HTTP, eventos de usuario o flujos de datos continuos.
- Usar servicios con Observables mejora la separación de responsabilidades y hace que tu aplicación sea más modular y mantenible.
- Es importante siempre manejar errores y suscribirse adecuadamente a los Observables.

¡Este es el flujo completo para crear un servicio que utiliza Observables en Angular, junto con el consumo de los datos en un componente!

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx