Reactividad en Angular

Pelayo Santos

1. Creación del Proyecto
2. Creación del componente secundario y consumo de API
3. Servicio con señales
4. Reactividad con signals en el consumo de APIS
5. Como se resuelve el error HttpClient provider
6. Ciclo de vida en componentes vs servicios
7. Envolver signal en otra signal

1.CREACIÓN DE UN PROYECTO EN ANGULAR 18

Compruebo que tengo la versión 18 de Angular CLI.

```
C:\Users\Pelayo>ng v


     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 18.2.21
Node: 22.19.0
Package Manager: npm 11.6.2
OS: win32 x64
```

Creó el proyecto con CSS y sin SSR

```
PS D:\Mis_Documentos\Nebrija_DAM_2\Acceso_A_Datos\Angular> ng new actividadReactividad
? Which stylesheet format would you like to use? CSS             [ https://developer.mozilla.org/docs/Web/CSS
        ]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? no
```

2.CREACIÓN DE LOS COMPONENTES Y CONSUMO API

Creó el componente posts

```
PS D:\Mis_Documentos\Nebrija_DAM_2\Acceso_A_Datos\Angular\actividadReactividad> ng g c posts
CREATE src/app/posts/posts.component.html (21 bytes)
CREATE src/app/posts/posts.component.spec.ts (608 bytes)
CREATE src/app/posts/posts.component.ts (242 bytes)
CREATE src/app/posts/posts.component.css (0 bytes)
```

En “posts.component.ts” consumo la api mediante mi servicio posts

```ts
import { Component, inject, Signal } from '@angular/core';
import { PostsService, Post } from '../services/posts.service';

@Component({
  selector: 'app-posts',
  standalone: true,
  imports: [],
  templateUrl: './posts.component.html',
  styleUrl: './posts.component.css'
})
export class PostsComponent {
  // Injecto mi servicio
  private postsService = inject(PostsService);

  // Envuelvo mi signal del servicio en otra
  public posts: Signal<Post[]> = this.postsService.posts;
}
```

En “post.component.html” que muestra los datos usando control Flow

```
<h1>JSONplaceholder:</h1>

<!-- Compruebo que el signal "post" no este vacio -->
@if (posts().length > 0) {
<ul>
    <!-- Usamos @for para iterar sobre la signal "posts" -->
    @for (post of posts(); track post.id) {
    <li>
        <h3>{{ post.title }}</h3>
        <p>{{ post.body }}</p>
    </li>
    }
</ul>
} @else {
<p>Posts esta vacio o Cargando</p>
}
```

También creó un padre.component

```
PS D:\Mis_Documentos\Nebrija_DAM_2\Acceso_A_Datos\Angular\proyectoReactividad> ng g c padre
CREATE src/app/padre/padre.component.html (21 bytes)
CREATE src/app/padre/padre.component.spec.ts (608 bytes)
CREATE src/app/padre/padre.component.ts (242 bytes)
CREATE src/app/padre/padre.component.css (0 bytes)
```

En el que solo modificó “padre.component.html”

```
Go to component
1   <p>Componente Padre</p>
```

3.CREACIÓN DE UN SERVICIO CON SEÑALES

Creó el servicio posts

```
PS D:\Mis_Documentos\Nebrija_DAM_2\Acceso_A_Datos\Angular\actividadReactividad> ng g s posts
CREATE src/app/posts.service.spec.ts (368 bytes)
CREATE src/app/posts.service.ts (143 bytes)
```

Estructura del servicio con interfaz y el manejo de observables mediante una señal

```ts
import { Injectable, inject, signal } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { toSignal } from '@angular/core/rxjs-interop';
import { of } from 'rxjs';
import { catchError, map } from 'rxjs/operators';

// Interfaz de la estrutura de un post
export interface Post {
  userId: number;
  id: number;
  title: string;
  body: string;
}

@Injectable({
  providedIn: 'root'
})
export class PostsService {
  private http = inject(HttpClient);
  private postsUrl = 'https://jsonplaceholder.typicode.com/posts';


  // Convierte el onservable a una señal usando el metodo toSignal() que se encarga de gestionar las subscripciones
  public readonly posts = toSignal(
    this.http.get<Post[]>(this.postsUrl).pipe(
      // Controlo los posibles errores de carga de posts y en caso de error devuelvo un array vacio
      catchError(error => {
        console.error("No se han cargado los posts");
        return of([]);
      })
    ),
    { initialValue: [] }
  );
}
```

4.REACTIVIDAD CON SIGNALS EN EL CONSUMO DE APIS

La reactividad con signals consiste en que la señal guarda un valor y si dicho valor cambia envía una señal para volver a renderizar ese componente.

En el caso de las APIs se suele devolver un valor de tipo observable que convertimos en señal con el método toSignal() que se encarga de suscribirse y des suscribirse a dicho observable además de devolverlo como una señal.

```ts
// Convierte el onservable a una señal usando el metodo toSignal() que se encarga de gestionar las subscripciones
public readonly posts = toSignal(
  this.http.get<Post[]>(this.postsUrl).pipe(
    // Controlo los posibles errores de carga de posts y en caso de error devuelvo un array vacio
    catchError(error => {
      console.error("No se han cargado los posts");
      return of([]);
    })
  ),
  { initialValue: [] }
);
```

5.CÓMO SE RESUELVE EL ERROR HTTPClient PROVIDER

El problema ocurre cuando angular no encuentra el provider de HTTPClient par solucionarlo hay que añadir la función provideHttpClient() en “app.config.ts”

```
export const appConfig: ApplicationConfig = {
  providers: [
    provideRouter(routes),
    provideHttpClient()
  ]
};
```

6.CICLO DE VIDA EN COMPONENTE VS SERVICIO

La diferencia principal entre ambos ciclos de vida es que el componente se crea al mostrarse y muere cuando desaparece. En cambio cuando se crea una instancia de un servicio no se destruye a no ser que desaparezca su inyector, por lo que siempre se utiliza el mismo servicio.

7.Envolver signal en otra signal

Consiste en crear un signal que depende de otro y se actualiza automáticamente cuando el original cambia.

Ejemplo:

```
// Envuelvo mi signal del servicio en otra
public posts: Signal<Post[]> = this.postsService.posts;
```