Obteniendo datos de una API 14/20

RECURSOS MARCADORES

Uno de los procesos asíncronos más comunes son las **peticiones de datos desde una API**. Para esto, Angular posee su propio cliente HTTP destinado a cumplir con el propósito llamado `HttpClientModule` .

Consumo de API con Angular

Paso 1: Comienza importando el módulo `HttpClientModule` , en el módulo principal de tu aplicación desde `@angular/common/http` .

```
// app.module.ts
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    // ..
  ],
  imports: [
    // ...
    HttpClientModule
  ],
  providers: [],
  bootstrap: [ /* .. */ ]
```

```
})
export class AppModule { }
```

Paso 2: Crea un servicio para realizar todos los **llamados a una API** que necesites. En este servicio tienes que importar el cliente HTTP de Angular y crear un método por cada endpoint para el que necesites efectuar una petición.

```
// services/api.service.ts
import { HttpClient } from '@angular/common/http';

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

  constructor(
    private http: HttpClient,
  ) { }

  getProducts() {
    return this.http.get(`https://example.com/api/productos`);
  }
}
```

Paso 3: Importa este nuevo servicio en tus componentes para efectuar los llamados a la API.

```
// components/catalogo/catalogo.component.ts
import { ApiService } from 'src/app/services/api.service';

@Component({
  selector: 'app-catalogo',
  templateUrl: './catalogo.component.html',
  styleUrls: ['./catalogo.component.scss']
})
export class CatalogoComponent implements OnInit {

  public productos: Producto[] = [];

  constructor(
    private apiService: ApiService
  ) { }

  ngOnInit(): void {
      this.apiService.getProducts()
        .subscribe(res => {
          this.productos = res;
        });
  }

  // ...
}
```

El método `ngOnInit()` es el lugar apropiado para los llamados asincrónicos, recuerda que no es buena práctica hacerlo en el constructor.

Todo el cliente HTTP de Angular está basado en **Observables**, por lo tanto, es recomendable suscribirse al método del servicio para obtener los datos cuando la API responda.

> *TIP: No es necesario que hagas un .unsubscribe() luego del llamado a la API. Angular ya lo hace por ti, cuando usas su cliente HTTP.*

Jugando con observables

Si no tienes a tu disposición una API Rest que devuelva datos para tu aplicación, voy a enseñarte un pequeño truco para que aun así puedas continuar con tu desarrollo con un mock de datos.

Un Mock es una simulación de los datos reales que devolverá la API, salvo que esta vez obtendrás dichos datos desde el servicio a través de un Observable.

```
// services/api.service.ts
import { Observable, of } from 'rxjs';

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

  constructor() { }

  getProducts(): Observable<Producto[]> {
    return of([
      {
        id: 1,
        name: 'Automobil de juguete',
        precio: 100,
```

```
        image:
'https://static3.depositphotos.com/1000865/118/i/600/depositphotos_1183767-stock-photo-
toy-car.jpg'
      },
      {
        id: 2,
        name: 'Muñeca de trapo',
        precio: 180,
        image: 'https://kinuma.com/8869-home_default/muneca-de-trapo-mali.jpg'
      },
      {
        id: 3,
        name: 'Pelota de futbol',
        precio: 120,
        image: 'https://media.istockphoto.com/photos/soccer-ball-isolated-3d-rendering-
picture-id1257575611?
k=20&m=1257575611&s=612x612&w=0&h=g530fFJspT42xFGY7HycLvpBKLXpJ2XAkKCRyY-SK80='
      },
      {
        id: 4,
        name: 'Castillo',
        precio: 220,
        image: 'https://i.imgur.com/44nzvkQ.jpg'
      }
    ])
  }

}
```

En el ejemplo anterior, desde **RxJS** se está importando **“Observable” y “of”** que te ayudarán a preparar tus datos.

Con **Observable** puedes tipear la respuesta de tus métodos de la siguiente manera `Observable<Producto[]>` para indicar que este devolverá un observable con un array de productos. La función `of` convierte lo que sea que le pongas dentro (un objeto, un array, un número, etc), en un observable.

De esta forma, sin tener una API real, puedes simular la interacción de tu componente con datos provenientes de una proceso asincrónico.

Ver código fuente del proyecto

Contribución creada con los aportes de Kevin Fiorentino.

Lecturas recomendadas

- GitHub - platzi/angular-componentes at 11-step
  https://github.com/platzi/angular-componentes/tree/11-step
- Fake Store API
  http://fakestoreapi.com/
- Curso de Consumo de APIs REST con Angular - Platzi
  https://platzi.com/cursos/angular-apis/