6. Enlace de datos – Data binding en Angular

Anteriormente habíamos visto cómo utilizar el **binding por interpolación**, utilizando **"`{{propiedad}}`"**, que simplemente nos permite acceder a algún dato que tenemos guardado en la clase de nuestro componente.

Ahora vamos a ver el **"two way data binding"** (binding bidireccional – enlace bidireccional) y, mediante éste, vamos a enlazar de forma bidireccional un componente de la vista con un dato de la clase de forma que, si modifico la vista, se modifica de forma inmediata en la clase y viceversa. Este **binding bidireccional** nos da varias directivas más para trabajar en las vistas de Angular.

A continuación veremos un ejemplo de su uso, añadiendo a la vista de "**ZapatillasComponent**" un campo de texto que permita escribir ahí lo que será un nuevo elemento que se debe añadir al listado de marcas de la aplicación.

1. Primero vamos a crear una nueva propiedad en la clase llamada **"mi_marca"** de tipo **"string"**.

2. En la vista del componente, creamos un **párrafo** que diga "**Añadir marca:**" y vamos a crear un **"<input>"** de tipo texto que va a tener un atributo **"`[(ngModel)]`"**, que es una directiva para poder modificar una propiedad que tenemos en el modelo de datos. Esta directiva permite hacer el **"two way data binding"** y nos va a permitir modificar una propiedad de la clase del componente de manera instantánea y reactiva. Además, los cambios se verán reflejados de forma automática tanto en la vista como en los datos en la clase del componente.

```
export class ZapatillasComponent implements OnInit, CommonModule {
  public titulo: string = 'Componente Zapatillas';
  public zapatillas: Array<Zapatilla>;
  public marcas: String[];
  public color: string;
  public mi_marca: string;

  constructor() {
    this.marcas = new Array();
    this.zapatillas = [
      new Zapatilla('Reebook Classic', 'Reebook', 'Blanco', 80, true),
      new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
      new Zapatilla('Adidas Yezzy', 'Adidas', 'Gris', 180, false),
    ];
    this.color = 'yellow';
    this.mi_marca='';
  }
```

```
<h1>Añadir marca: </h1>
<p>
    <input type="text" [(ngModel)]="mi_marca" />
</p>
<p>{{mi_marca}}</p>
```

Solo para Angular 16 y anteriores (no standalone)

Por último, para poder trabajar con esta directiva "ngModel" y hacer el enlace, tenemos que cargar en nuestro **"app.module.ts"** un módulo para trabajar con los formularios:

```
import { FormsModule } from '@angular/forms';
```

Una vez importado, lo tenemos que cargar en los **"imports: []"** que es el array dentro del **"@NgModule"** que nos permite cargar módulos internos de Angular, módulos externos que hayamos creado nosotros o módulos externos que hayamos instalado en nuestro proyecto. Cargamos por tanto aquí el módulo importado:

```
imports: [BrowserModule, FormsModule]
```

Para componentes standalone

Por último, para poder trabajar con esta directiva "ngModel" y hacer el enlace, tenemos que cargar en nuestro **"zapatillas.component.ts"** un módulo para trabajar con los formularios:

```
import { FormsModule } from '@angular/forms';
```

Una vez importado, lo tenemos que cargar en los **"imports: []"**.

Cargamos por tanto aquí el módulo importado:

```
imports: [BrowserModule, FormsModule]
```

Ejercicios:

1. Añadir un nuevo botón en la vista y utilizar el evento **click** para llamar a un nuevo método llamado **"getMarca()"** que haga un "**alert**" de la marca que esté guardada en la propiedad "**mi_marca**".

2. Añadir un nuevo evento a la vista: Debajo del botón anterior, vamos a crear otro que va a ser **"Añadir marca"** y, utilizando el evento **"click"** (como ya hemos visto antes), vamos a llamar a un método llamado **"addMarca()"**. Éste método añadirá un nuevo elemento al **Array** de **marcas** (con el contenido del cambo anterior), para ver que se actualiza realmente el listado de forma reactiva e instantánea.

3. Entramos a la plantilla "zapatillas.component.html"y añadimos un input tipo texto antes de la lista desordenada que contiene la directiva "**ngSwitch**", para poder modificar la propiedad "**color**" de la clase automáticamente cuando escribamos en dicho input. Para eso, a nuestro input vamos a agregarle la directiva **"[(ngModel)]"** y la enlazaremos con la propiedad "**color**" de la clase.

Soluciones:

1. En la vista del componente Zapatillas:

```
<button (click)="getMarca()">Mostrar marca</button>
</p>
<strong>{{mi_marca}}</strong>
```

En la clase del componente:

```
45
46      getMarca(){
47          alert(this.mi_marca);
48      }
```

2. En la vista del componente:

```
6  <button (click)="getMarca()">Mostrar marca</button>
7  <button (click)="addMarca()">Añadir marca</button>
```

En la clase del componente:

```
49
50      addMarca(){
51          this.marcas.push(this.mi_marca);
52      }
```

Actualizamos la pantalla y vemos como al hacer clic al botón "**Añadir marca**" la nueva marca se añade de manera instantánea y reactiva.

En JavaScript puro, para hacer esto, habría que **capturar** el **evento** del **botón**, añadir al Array ese nuevo elemento y luego llamar a un método que volviese a pintar toda esta información. Sin embargo, en **Angular es todo reactivo**, si se añade otra marca se actualiza todo de forma inmediata y muy rápido.

---

3. Dentro de la vista del componente:

```
<div class="seccion">
    <h3>El color de la mayoría de nuestras zapatillas es:</h3>
    <input type="text" [(ngModel)]="color" /><br><br>
    <ul [ngSwitch]="color">
        <li *ngSwitchCase="'yellow'"><span
            [style]="{'background-color': color}">amarillo</span>.</li>

        <li *ngSwitchCase="'red'"><span
            [ngStyle]="{'background-color': color}">rojo</span>.</li>

        <li *ngSwitchCase="'blue'"><span
            [ngStyle]="{'background-color': color}">azul</span>.</li>

        <li *ngSwitchCase="'orange'"><span
            style="background-color: orange;">naranja</span>.</li>

        <li *ngSwitchDefault="'pink'"><span
            style="background-color: rgb(221, 0, 255);">rosa</span>.</li>

    </ul>
</div>
```