

# 4-Flow

**Flujo de datos entre componentes Angular**

- 1. Comunicación entre componentes**
- 2. El patrón Contenedor / Presentadores**
- 3. Otras comunicaciones**

# 1. Comunicación entre componentes

Necesidad de comunicación

Escenarios

# 1.1 Necesidad de comunicación

## ■ Aplicaciones Complejas

- Principio de desarrollo *Divide y Vencerás*
- Múltiples páginas SPA -

## 1.2 Escenarios

- Comunicar componentes **acoplados**
- Comunicar componentes entre **páginas distintas**
- Comunicar componentes entre **estructuras dinámicas**

| Recap:

# 1. Comunicación entre componentes

Necesidad de comunicación

Escenarios

# 2. Contenedor / Presentadores

El patrón Contenedor / Presentadores

El contenedor

Envío hacia el presentador con `@Input()`

Respuesta del presentador con `@Output()`

## 2.1 El patrón Contendor / Presentadores

Es una elección de arquitectura que promueve:

### Reparto de responsabilidades:

- **Contenedor:** gestión de datos
- **Presentadores:** interacción con usuario

### Reutilización de presentadores

- **Librerías:** presentadores genéricos



# Contenedor y presentadores

```
ng g m car --routing true
ng g c car/car
ng g c car/car/display
ng g c car/car/pedals
```

```
{
  path: 'car',
  loadChildren: './car/car.module#CarModule'
}
```

```
<a routerLink="car" class="button">
  <span> 4 - Car</span>
</a>
```

# Container

```
<app-display [model]="car.name"
              [currentSpeed]="car.currentSpeed"
              [topSpeed]="car.maxSpeed"
              [units]="'km/h'">
</app-display>
<app-pedals (brake)="onBrake($event)"
            [disableBrake]="disableBrake"
            (throttle)="onThrottle($event)"
            [disableThrottle]="disableThrottle">
</app-pedals>
```

# Manejo de datos

```
public car: CarModel;  
public disableBrake: boolean;  
public disableThrottle: boolean;  
  
constructor() {}  
  
public ngOnInit() {  
    this.car = { name: 'Roadster', maxSpeed: 120, currentSpeed: 0 };  
    this.checkLimits();  
}  
private checkLimits() {  
    this.disableBrake = false;  
    this.disableThrottle = false;  
    if (this.car.currentSpeed <= 0) {  
        this.car.currentSpeed = 0;  
        this.disableBrake = true;  
    } else if (this.car.currentSpeed >= this.car.maxSpeed) {  
        this.car.currentSpeed = this.car.maxSpeed;  
        this.disableThrottle = true;  
    }  
}
```

# Lógica de negocios

```
public onBrake(drive: number) {  
    this.car.currentSpeed -= this.getDelta(drive);  
    this.checkLimits();  
}  
  
public onThrottle(drive: number) {  
    this.car.currentSpeed += this.getDelta(drive);  
    this.checkLimits();  
}  
  
private getDelta = (drive: number) =>  
    drive + (this.car.maxSpeed - this.car.currentSpeed) / 10;
```

## 2.3 Envío hacia el presentador con @Input()

Envío de información **desde el contenedor hacia el presentador**

Usa `[propiedad]="expresion"` en el contenedor

Y `@Input()` `propiedad` en el presentador

## Recepción en el controlador

```
export class DisplayComponent implements OnInit {  
  @Input() public model: string;  
  @Input() public currentSpeed: number;  
  @Input() public topSpeed: number;  
  @Input() public units: string;  
  constructor() {}  
  ngOnInit() {}  
  public getSpeedClass = () =>  
    this.currentSpeed < this.getThreshold() ? 'primary' : 'secondary';  
  private getThreshold = () => this.topSpeed * 0.8;  
}
```

## Presentación en la vista

```
<h2> {{ model }} </h2>
<h3> Top speed: {{ topSpeed | number:'1.0-0' }}</h3>
<div class="card">
  <div class="section">
    {{ currentSpeed | number:'1.2-2' }} {{ units }}
  </div>
  <progress [value]="currentSpeed"
            [ngClass]="getSpeedClass()"
            [max]="topSpeed">
  </progress>
</div>
```

## 2.4 @Output()

Envío de información **desde el presentador hacia el contendor**

Usa `(evento)="instruccion"` en el contendor

Y `@Output() evento = new EventEmitter<any>()` en el presentador



## Emisión desde el controlador

```
export class PedalsComponent implements OnInit {  
  @Input() public disableBrake: boolean;  
  @Input() public disableThrottle: boolean;  
  @Output() public brake = new EventEmitter<number>();  
  @Output() public throttle = new EventEmitter<number>();  
  
  constructor() {}  
  
  ngOnInit() {}  
}
```

## Suscripción desde la vista

```
<h3>
  Pedals:
</h3>
<form>
  <input value="brake"
    class="secondary"
    type="button"
    [disabled]="disableBrake"
    (click)="brake.emit(1)" />
  <input value="throttle"
    class="tertiary"
    type="button"
    [disabled]="disableThrottle"
    (click)="throttle.emit(1)" />
</form>
```

Recap:

## 2. Contenedor / Presentadores

El patrón Contenedor / Presentadores

El contenedor

Envío hacia el presentador con `@Input()`

Respuesta del presentador con `@Output()`

# 3. Otras comunicaciones

Comunicación entre distintas páginas

Comunicación entre estructuras desacopladas

## 3.1 Comunicación entre distintas páginas

- A través del `RouterModule`

En `about-routing.module`

```
{  
  path: 'authors/:id',  
  component: AuthorComponent  
}
```

En `authors-component.html`

```
<a routerLink="albertobasalo"> <span> Alberto Basalo</span> </a>  
<a routerLink="johndoe"> <span> John Doe</span> </a>
```

En `author.component.ts`

```
export class AuthorComponent implements OnInit {  
  public authorId = '';  
  constructor(activatedRoute: ActivatedRoute) {  
    this.authorId = activatedRoute.snapshot.params['id'];  
  }  
  ngOnInit() {}  
}
```

## 3.2 Comunicación entre estructuras desacopladas

- Usando `Observables`

En el `ShellComponent`

```
<header class="sticky">
  <a routerLink="/"> <span class="icon-home"></span>4-Flow</a>
  <a routerLink="car"> <span class="icon-home"></span>Car</a>
  <a routerLink="contacts"> <span class="icon-home"></span>Contacts</a>
</header>
<main class="container ">
  <router-outlet></router-outlet>
</main>
<app-footer></app-footer>
```

Lo que pasa dentro de *main* no se conoce en *header*...

| Recap:

## **3. Otras comunicaciones**

**Comunicación entre distintas páginas**

**Comunicación entre estructuras desacopladas**



| Next:

# Servicios inyectables en Angular

## Inyección de dependencias

## Inversión del control

| **Blog de apoyo:** [Flujo de datos entre componentes Angular](#)

| **By** [Alberto Basalo](#)