5-Inject

Servicios inyectables en Angular

1. Inyección de dependencias

2. Inversión del control

1. Inyección de dependencias

Generación de servicios

Consumo de dependencias

Módulo y componente

```
ng g m converter --routing true
ng g c converter/converter
```

`app-routing.module.ts`

```
{
  path: 'converter',
  loadChildren: './converter/converter.module#ConverterModule'
},
```

converter-routing.module.ts

```
{
  path: '',
  component: ConverterComponent
}
```

header.component.html

```
<a routerLink="converter" class="button">
  <span> Converter</span>
</a>
```

1.1 Generación de servicios

```
ng g s converter/converter
```

Implementación

```
import { Injectable } from '@angular/core';

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

  public fromKilometersToMiles = kilometers => kilometers * 0.621;
}
```

1.2 Consumo de dependencias

```
export class ConverterComponent implements OnInit {
  public kilometers = 0;
  public miles: number;

  constructor(private converterService: ConverterService) {}

  public ngOnInit() { this.convert(); }

  public convert() {
    this.miles =
      this.converterService.fromKilometersToMiles(this.kilometers);
  }
}
```

Presentación en vista

```
<h2> Distance Converter.</h2>
<h3> From Europe to USA </h3>
<form>
  <fieldset>
    <section>
      <label for="kilometers">Kilometers</label>
      <input name="kilometers"
             type="number"
             [(ngModel)]="kilometers"
             placeholder="0" />
    </section>
  </fieldset>
  <input value="Convert"
         type="button"
         (click)="convert()">
</form>
<section>
  <h4>{{ miles | number:'1.2-2' }} miles</h4>
</section>
```

Recap:

1. Inyección de dependencias

Generación de servicios

Consumo de dependencias

2. Inversión del control

Interface y servicio base

Implementaciones

Provisión manual

Factoría

2.1 Interface y servicio base

```
ng g interface converter/culture-converter
ng g service converter/culture-converter
ng g component converter/culture-converter
```

```
export interface CultureConverter implements CultureConverter {
  sourceCulture: string;
  targetCulture: string;
  convertDistance: (source: number) => number;
  convertTemperature: (source: number) => number;
}
```

```
export class CultureConverterService implements CultureConverter {
  sourceCulture: string;
  targetCulture: string;
  convertDistance: (source: number) => number;
  convertTemperature: (source: number) => number;
  constructor() {}
}
```

Consumo

```
public source: string;
public target: string;
public sourceUnits = 0;
public targetUnits: number;
constructor(private cultureConverterService:CultureConverterService){
}

public ngOnInit() {
  this.source = this.cultureConverterService.sourceCulture;
  this.target = this.cultureConverterService.targetCulture;
  this.convert();
}
public convert() {
  this.targetUnits =
    this.cultureConverterService.convertDistance(this.sourceUnits);
}
```

```
<h2> Culture Converter.</h2>
<h3> From {{ source }} to {{ target }} </h3>
<form>
  <fieldset>
    <section>
      <label for="sourceUnits">Distance</label>
      <input name="sourceUnits"
             type="number"
             [(ngModel)]="sourceUnits"
             placeholder="0" />
    </section>
  </fieldset>
  <input value="Convert"
         type="button"
         (click)="convert()">
</form>
<section>
  <h4>Distance {{ targetUnits | number:'1.2-2' }} </h4>
</section>
```

2.2 Implementaciones

```
@Injectable({
  providedIn: 'root'
})
export class ConverterService {
  constructor() {}

  public fromKilometersToMiles = kilometers => kilometers * 0.621;

  public fromMilesToKilometers = miles => miles * 1.609;

  public fromCelsiusToFarenheit = celsius => celsius * (9/5) + 32;

  public fromFarenheitToCelsius = farenheit => (farenheit-32) * (5/9);
}
```

```
@Injectable()
export class EuropeConverterService {
  sourceCulture = 'USA';
  targetCulture = 'Europe';
  constructor(private converterService: ConverterService) {}
  convertDistance = this.converterService.fromMilesToKilometers;
  convertTemperature = this.converterService.fromFarenheitToCelsius;
}
```

```
@Injectable()
export class UsaConverterService implements CultureConverter {
  sourceCulture = 'Europe';
  targetCulture = 'USA';
  constructor(private converterService: ConverterService) {}
  convertDistance = this.converterService.fromKilometersToMiles;
  convertTemperature = this.converterService.fromCelsiusToFarenheit;
}
```

2.3 Provisión manual

```
{
  providers: [
    {
      provide: CultureConverterService,
      useClass: UsaConverterService
    }
  ]
}
```

2.4 Factoría

```
const cultureFactory = (converterService: ConverterService) => {
  if (environment.unitsCulture === 'metric') {
    return new EuropeConverterService(converterService);
  } else {
    return new UsaConverterService(converterService);
  }
};
```

```
export const environment = {
  appName: "Angular - Board",
  production: false,
  unitsCulture : 'metric'
};
```

La provisión del servicio apunta a la función factoría. Si además el servicio dependiese de otro tenemos que especificarlo en el sub-array `deps:[]`.

```
{
  providers: [
    {
      provide: CultureConverterService,
      useFactory: cultureFactory,
      deps: [ConverterService]
    }
  ]
}
```

Recap:

2. Inversión del control

Interface y servicio base

Implementaciones

Provisión manual

Factoría

Next:

Comunicaciones http en Angular

El cliente http

Operaciones con observables

Interceptores de llamadas

Blog de apoyo: Servicios inyectables en Angular

By Alberto Basalo