
ANGULAR 15

CONTENIDO

- Angular.
- Instalación.
- Commandos.
- Estructura (Scaffolding).
- Components.
- Routing.
- Directives.
- Interfaces.
- Pipes.
- Guards.
- Data Binding.
- Comunicación entre componentes.
- Manipulación del DOM.
- Services.
- Interceptors.
- async-await.
- RxJS.

Routing

Un routing module permite navegar entre diferentes vistas o componentes dentro de una aplicación.

Comando

```
ng generate module app-routing --flat --module=app
```

Actualización src/app/app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { MiComponente} from '.../mi-componente/mi-componente.component';

const routes: Routes = [
  { path: 'mi-componente', component: MiComponente}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})

export class AppRoutingModule { }
```

Routing

Declaración de rutas

```
import ...
const routes: Routes = [
  { path: 'mi-componentente', component: MiComponente },
  { path: 'ruta/a/miComponentente', component: MiComponente },

  // guards
  { path: 'mi-componentente', component: MiComponente, canActivate: [AuthGuard] },

  // parámetros
  { path: 'mi-componentente/:param1/:param2/:paramN', component: MiComponente },
  { path: 'mi-componentente/:param1/ruta/:param2/:paramN', component: MiComponente },

  // outlets
  { path: 'mi-componentente', outlet: 'seccion1', component: MiComponente },
  { path: 'mi-componentente', outlet: 'seccionB', component: MiComponente },

  // la ruta comodín va al final siempre
  {path: '**', pathMatch: 'full', redirectTo: 'login'},
];
```

Routing

Captura de parámetros

```
...
export class MiComponente implements OnInit {
      constructor(private route: ActivatedRoute,
                  private router: Router) {
      }

      ngOnInit() {
            // ActivatedRoute
            this.route.params.subscribe(params => {
                  const parametro1 = params['param1'];
            });

            // Snapshot
            const parametro1 = this.route.snapshot.params['param1'];

            // Router
            const currentUrl = this.router.url;
      }
}
```

Routing

Redireccionamiento

```
<!- Router Link ->
<a [routerLink]="['/ruta']">Ir a Componente</a>
<button [routerLink]="['/ruta']">Ir a Componente</button>
<!- Router ->
<button (click)="navegar1()">Ir a Componente</button>

...
export class MiComponente {
      constructor(private router: Router) { this.navegar2()}

      // Router Navigate
      navegar1() {
            this.router.navigate(['/ruta']);
            this.router.navigate(['/ruta', id]);
      }

      // Router NavigateByUrl
      navegar2() {
            this.router.navigateByUrl('/ruta');
      }
}
```

Directives

Las Directivas extienden la funcionalidad del HTML usando para ello una nueva sintaxis. Con ella podemos usar lógica que será ejecutada en el DOM.

Tipos de directivas

- Atributo.
  - ngModel: Implementa binding.
  - ngClass: permite añadir/eliminar varias clases.
  - ngStyle: permite asignar estilos inline.
- Estructurales.
  - *ngIf: Nos permite incluir condicionales de lógica.
  - *ngFor: Permite ejecutar bucles.
  - ngSwitch: esta directiva es similar al *ngIf, y es como el switch en lógica de programación.
  - Otras.
- Componentes.

Directives

Directivas de atributo

- ngModel -> Data Binding
- ngClass

```
<some-element [ngClass]="Class">...</some-element>

<div [ngClass]=" 'clase1 clase2 ... claseN' ">...</div>
<div [ngClass]=" [ 'clase1', 'clase2', ..., 'claseN' ] ">...</div>
<div [ngClass]=" { 'clase1': a == b, 'clase2': true, ..., 'claseN': false } ">...</div>
<div [ngClass]=" {'clase1 clase2 … claseN' : condicion } ">...</div>
```

- ngStyle

```
<some-element [ngStyle]="{'style1':‘value1’, 'style2':‘value2’, ..., 'styleN':‘valueN’
}">...</some-element>

<div [ngStyle]=" {'background-color':'green', 'color': 'red' } ">...</div>
<div [ngStyle]=" {'background-color': bgColor, 'color': color } ">...</div>
<div [ngStyle]=" {'background-color': style1 ? 'red' : 'green'} ">...</div>
<div [style.font-size.px]="24">...</div>
```

Directives

Directivas estructurales

- *ngIf

```
<some-element *ngIf="condition">...</some-element>

<div *ngIf="isActive">
      <p>This is some text in a paragraph.</p>
</div>

<div *ngIf="isActive && isVisible">
      <p>This is some text in a paragraph.</p>
</div>

<div *ngIf="user.role === 'admin'">
      <p>This is some text in a paragraph.</p>
</div>
```

Directives

Directivas estructurales

- *ngFor

```
<some-element *ngFor="iterative structure”>...</some-element>

<li *ngFor="let user of users">
      nombre: {{user.name}}
</li>

<li *ngFor="let user of users; index as i">
      usuario de la pos {{i}} con nombre {{user.name}}
</li>
```

Directives

Directivas estructurales

- ngSwitch

```
<container-element [ngSwitch]="switch_expression">
   <some-element *ngSwitchCase="match_expression_1">...</some-element>
...
   <some-element *ngSwitchDefault>...</some-element>
</container-element>

<div [ngSwitch]="color">
   <section *ngSwitchCase="'red'">...</section>
   <section *ngSwitchCase="'verde'">...</section>
   <section *ngSwitchCase="'azul'">...</section>

   <section *ngSwitchDefault>...</section>
</div>
```

Directives

Directivas componentes

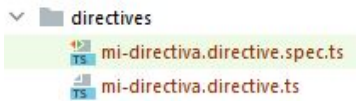
Las Directivas de Componente son directivas con un Template. Los componentes tienen decoradores "@Component", el componente es un decorador @Directive que es extendido con características propias de los templates.

Comando

```
ng generate directive directive-name [options]
```

Resultado

```
├── .../
├──────── mi-directive.directive.spec.ts
└──────── mi-directive.directive.ts
```



Directives

Resultado

```
...
@Directive({
      selector: '[appMiDirectiva]'
})
export class MiDirectivaDirective {

      constructor() { }

}
```

Directives

Uso

```
...
export class MiDirectivaDirective {

      constructor(private eleRef: ElementRef) {
            eleRef.nativeElement.style.background = 'red';
      }

}


<h1 appMiDirectiva>Resaltar!</h1>
```

Directives

Uso

```
export class MiDirectivaDirective {

      @Input('appMiDirectiva') highlightColor: string | undefined;

      constructor(private eleRef: ElementRef) {
            eleRef.nativeElement.style.background = this.highlightColor;
      }

}
```

```
<h1 [appMiDirectiva]="'blue'">Resaltar!</h1>
```

Directives

Uso

```
export class MiDirectivaDirective {

      constructor(private el: ElementRef, private renderer: Renderer2) {
      }

      @HostListener('mouseenter') onMouseEnter() {
            this.highlight('yellow');
      }

      @HostListener('mouseleave') onMouseLeave() {
            this.highlight(null); }

      private highlight(color: string) {
            this.renderer.setStyle(this.el.nativeElement, 'backgroundColor', color);
      }

}

<h1 appMiDirectiva>Resaltar!</h1>
```