

# Angular

## *Directivas*

CertiDevs

#  ndice de contenidos

1. Directivas .....	1
1.1. Directivas estructurales .....	1
1.2. Directivas de atributos .....	2
1.3. Creaci�n de directivas personalizadas .....	2
2. Ejemplo 1: Lista de personas .....	4
2.1. Paso 1: Crear un nuevo proyecto Angular .....	4
2.2. Paso 2: Generar el componente "PersonListComponent" .....	4
2.3. Paso 3: Agregar datos de ejemplo en el componente "PersonListComponent" .....	4
2.4. Paso 4: Utilizar *ngFor para mostrar la lista de personas .....	5
2.5. Paso 5: Agregar un campo de b�squeda para filtrar personas .....	5
2.6. Paso 6: Crear el pipe "FilterPipe" para filtrar personas .....	5
2.7. Paso 7: Importar FormsModule en AppModule .....	6
2.8. Paso 8: Utilizar el componente "PersonListComponent" en "AppComponent" .....	6
2.9. Paso 9: Probar la aplicaci�n .....	7
3. Ejemplo 2: highlight .....	7
3.1. Paso 1: Crear un nuevo proyecto de Angular .....	7
3.2. Paso 2: Crear una directiva personalizada .....	7
3.3. Paso 4: Utilizar la directiva personalizada .....	8
3.4. Paso 5: Ejecutar la aplicaci�n .....	9
4. Ejemplo 3: directivas predefinidas .....	9
4.1. Paso 1: Crear un nuevo componente .....	9
4.2. Paso 2: Modificar el componente 'directives-example' .....	9
4.3. Paso 3: Agregar el componente al m�dulo principal .....	10
4.4. Paso 4: Utilizar el componente en 'app.component.html' .....	11
4.5. Paso 5: A�adir estilos en 'directives-example.component.css' .....	11
4.6. Paso 6: Ejecutar la aplicaci�n .....	11

# 1. Directivas

Las directivas en Angular son elementos que permiten manipular el DOM y controlar la representación de la vista.

Hay dos tipos de directivas:

- ✦ Directivas estructurales: Modifican la estructura del DOM agregando, eliminando o reemplazando elementos.

- ✦ Directivas de atributos: Cambian la apariencia o el comportamiento de un elemento del DOM.

A diferencia de los componentes, las directivas no tienen una plantilla y solo afectan al elemento en el que se aplican.

## 1.1. Directivas estructurales

Las directivas estructurales son responsables de modificar la estructura del DOM agregando, eliminando o reemplazando elementos.

Estas directivas trabajan manipulando el flujo de control en la plantilla y utilizando la sintaxis de enlace de propiedades con un asterisco (\*) como prefijo.

Las directivas estructurales más comunes en Angular incluyen:

**\*ngIf**: Esta directiva se utiliza para agregar o eliminar un elemento del DOM según una condición dada. Si la condición es verdadera, el elemento se muestra; si la condición es falsa, el elemento se elimina del DOM.

```
<div *ngIf="isVisible">Este elemento es visible</div>
```

**\*ngFor**: Esta directiva se utiliza para iterar sobre una colección y crear elementos del DOM para cada elemento de la colección. La directiva **\*ngFor** crea un contexto de plantilla para cada iteración y proporciona variables locales que se pueden utilizar en la plantilla.

```
<ul>
  <li *ngFor="let item of items">{{ item }}</li>
</ul>
```

La variable **index** es una variable local opcional que representa el índice del elemento actual en la iteración. Puedes agregar **index** cuando necesites realizar operaciones basadas en el índice del elemento actual, como aplicar estilos específicos a elementos pares o impares, mostrar el número de la lista junto con el elemento, etc.

Si deseas agregar el **index** en **\*ngFor**, puedes hacerlo de la siguiente manera:

```
<ul>
```

```
É <li *ngFor="let item of items; index as i">{{ i + 1 }}. {{ item.name }}</li>
</ul>
```

En este ejemplo, hemos agregado la variable local `i` para almacenar el índice del elemento actual en la iteración, y lo mostramos junto con el nombre del elemento.

## 1.2. Directivas de atributos

Las directivas de atributos son responsables de cambiar la apariencia o el comportamiento de un elemento del DOM.

Estas directivas trabajan manipulando los atributos del elemento, agregando clases CSS o escuchando eventos del DOM.

Algunas directivas de atributos comunes en Angular incluyen:

**ngClass:** Esta directiva se utiliza para agregar o eliminar clases CSS en un elemento del DOM según una expresión dada. Puede aceptar un objeto, un array o una cadena como entrada y actualizar dinámicamente las clases del elemento.

```
<button [ngClass]="{active: isActive, disabled: isDisabled}">Mi Botón</button>
```

**ngStyle:** Esta directiva se utiliza para agregar o actualizar estilos en línea en un elemento del DOM según una expresión dada. Puede aceptar un objeto que contenga pares clave-valor de nombres de propiedades CSS y sus valores.

```
<div [ngStyle]="{'background-color': backgroundColor, 'font-size': fontSize}">Estilos
dinámicos</div>
```

**ngModel:** Esta directiva se utiliza para implementar el **enlace bidireccional** en formularios y se encuentra en el módulo `FormsModule`. La directiva `ngModel` sincroniza el valor de un elemento del formulario con una propiedad del componente.

```
<input [(ngModel)]="username" placeholder="Introduce tu nombre de usuario">
```

## 1.3. Creación de directivas personalizadas

Además de las directivas proporcionadas por Angular, también es posible crear directivas personalizadas.

Para crear una directiva personalizada, se utiliza la clase `@Directive` como decorador de una clase TypeScript.

El decorador `@Directive` acepta un objeto de metadatos que define las propiedades clave de la directiva, como el `selector`.

A continuaci3n se muestra un ejemplo b3sico de una directiva personalizada que cambia el color de fondo de un elemento:

```
import { Directive, ElementRef, Input, OnInit } from '@angular/core';

@Directive({
  selector: '[appBackgroundColor]'
})
export class BackgroundColorDirective implements OnInit {

  @Input('appBackgroundColor') backgroundColor: string;

  constructor(private el: ElementRef) {}
  ngOnInit() {
    this.el.nativeElement.style.backgroundColor = this.backgroundColor;
  }
}
```

En este ejemplo, la directiva `BackgroundColorDirective` utiliza el decorador `@Input` para recibir un valor de color de fondo y el servicio `ElementRef` para acceder al elemento del DOM en el que se aplica la directiva.

La directiva se aplica a un elemento en la plantilla utilizando el selector de atributos `appBackgroundColor`.

Para utilizar la directiva personalizada en una plantilla, primero debe ser declarada en un m3dulo Angular:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { BackgroundColorDirective } from './background-color.directive';

@NgModule({
  declarations: [AppComponent, BackgroundColorDirective],
  imports: [BrowserModule],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Luego, la directiva puede ser utilizada en una plantilla de la siguiente manera:

```
<div [appBackgroundColor]=" 'red' ">Este elemento tiene un fondo rojo</div>
```

## 2. Ejemplo 1: Lista de personas

### 2.1. Paso 1: Crear un nuevo proyecto Angular

Ejecuta el siguiente comando para crear un nuevo proyecto Angular llamado `angular-directives-example`:

```
ng new angular-directives-example
```

### 2.2. Paso 2: Generar el componente "PersonListComponent"

Ejecuta el siguiente comando para generar un nuevo componente llamado "PersonListComponent":

```
ng generate component person-list
```

### 2.3. Paso 3: Agregar datos de ejemplo en el componente "PersonListComponent"

Agrega un array de objetos que representen a las personas en el archivo `src/app/person-list/person-list.component.ts`:

```
import { Component, OnInit } from '@angular/core';

interface Person {
  id: number;
  name: string;
}

@Component({
  selector: 'app-person-list',
  templateUrl: './person-list.component.html',
  styleUrls: ['./person-list.component.css']
})
export class PersonListComponent implements OnInit {
  people: Person[] = [
    { id: 1, name: 'Alice' },
    { id: 2, name: 'Bob' },
    { id: 3, name: 'Carol' },
    { id: 4, name: 'Dave' },
  ];

  constructor() {}
  ngOnInit(): void {}
}
```

```
}
```

## 2.4. Paso 4: Utilizar \*ngFor para mostrar la lista de personas

Abre el archivo `src/app/person-list/person-list.component.html` e implementa la siguiente plantilla utilizando la directiva estructural `*ngFor` para iterar sobre la lista de personas:

```
<ul>
  <li *ngFor="let person of people">
    {{ person.name }}
  </li>
</ul>
```

## 2.5. Paso 5: Agregar un campo de búsqueda para filtrar personas

Agrega un campo de entrada para buscar personas por nombre en el archivo `src/app/person-list/person-list.component.html`:

```
<label for="search">Buscar persona:</label>
<input [(ngModel)]="search" id="search" type="text">
<ul>
  <li *ngFor="let person of people | filter: search">
    {{ person.name }}
  </li>
</ul>
```

## 2.6. Paso 6: Crear el pipe "FilterPipe" para filtrar personas

Ejecuta el siguiente comando para generar un nuevo pipe llamado `FilterPipe`:

```
ng generate pipe filter
```

Implementa la lógica de filtrado en el archivo `src/app/filter.pipe.ts`:

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'filter'
})
```

```
export class FilterPipe implements PipeTransform {

  transform(items: any[], searchText: string): any[] {
    if (!items) {
      return [];
    }
    if (!searchText) {
      return items;
    }

    searchText = searchText.toLowerCase();
    return items.filter(item => item.name.toLowerCase().includes(searchText));

  }
}
```

## 2.7. Paso 7: Importar FormsModule en AppModule

Para utilizar `[(ngModel)]` en el componente `PersonListComponent`, necesitarás importar el módulo `FormsModule`.

Abre el archivo `src/app/app.module.ts` e importa el módulo:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { PersonListComponent } from './person-list/person-list.component';
import { FilterPipe } from './filter.pipe';

@NgModule({
  declarations: [AppComponent, PersonListComponent, FilterPipe],
  imports: [BrowserModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

## 2.8. Paso 8: Utilizar el componente "PersonListComponent" en "AppComponent"

Abre el archivo `src/app/app.component.html` y reemplaza su contenido con el siguiente código para utilizar el componente `PersonListComponent`:

```
<app-person-list></app-person-list>
```



## 2.9. Paso 9: Probar la aplicación

Ejecuta `ng serve` y abre tu navegador en <http://localhost:4200/>.

Verás la lista de personas y podrás filtrarlas por nombre utilizando el campo de búsqueda.

## 3. Ejemplo 2: highlight

### 3.1. Paso 1: Crear un nuevo proyecto de Angular

Vamos a crear un nuevo proyecto de Angular utilizando el siguiente comando:

```
ng new angular-directives-example
```

Ejecuta el comando y sigue las instrucciones en la terminal. Luego, navega hacia el directorio del proyecto:

```
cd angular-directives-example
```

### 3.2. Paso 2: Crear una directiva personalizada

Vamos a crear una directiva personalizada llamada `highlight`.

Esta directiva cambiará el color de fondo del elemento al que se aplique.

Ejecuta el siguiente comando para generar la directiva:

```
ng generate directive highlight
```

Esto generará el archivo `highlight.directive.ts` en la carpeta `src/app`.

Abre este archivo y reemplaza su contenido con el siguiente código:

```
import { Directive, ElementRef, HostListener, Input } from '@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  @Input() highlightColor: string;

  constructor(private el: ElementRef) {}

  @HostListener('mouseenter') onMouseEnter() {
```

```

    this.highlight(this.highlightColor || 'yellow');
  }

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

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}

```

En el código anterior, hemos importado las clases necesarias y hemos creado la directiva `HighlightDirective` con un selector `appHighlight`.

También hemos utilizado el decorador `@Input()` para permitir la personalización del color de fondo.

Los eventos `mouseenter` y `mouseleave` se manejan usando el decorador `@HostListener()`.

### 3.3. Paso 4: Utilizar la directiva personalizada

Para utilizar nuestra directiva personalizada, primero necesitamos importarla en el archivo `app.module.ts`:

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HighlightDirective } from './highlight.directive';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent,
    HighlightDirective
  ],
  imports: [
    BrowserModule,
  ]
})
export class AppModule { }

```

Ahora podemos usar la directiva en nuestro componente `app.component.html`:

```

<h1 appHighlight>¡Hola, mundo!</h1>
<p appHighlight highlightColor="lightblue">Soy un párrafo con color de fondo
personalizado.</p>

```

Al aplicar la directiva `appHighlight` a los elementos, cambiarán su color de fondo al pasar el cursor

sobre ellos.

### 3.4. Paso 5: Ejecutar la aplicación

Para ver nuestra directiva en acción, ejecuta el siguiente comando en la terminal:

```
ng serve
```

Abre tu navegador y visita 'http://localhost:4200'. Verás que al pasar el cursor sobre los elementos con la directiva 'appHighlight', su color de fondo cambiará según lo especificado.

## 4. Ejemplo 3: directivas predefinidas

En este ejercicio, exploraremos las principales directivas predefinidas en Angular, incluidas las directivas estructurales y las directivas de atributo.

Las directivas estructurales modifican la estructura del DOM, mientras que las directivas de atributo cambian la apariencia o el comportamiento de un elemento.

### 4.1. Paso 1: Crear un nuevo componente

Vamos a crear un componente llamado `directives-example` para ilustrar el uso de las directivas predefinidas en Angular.

Ejecuta el siguiente comando en la terminal:

```
ng generate component directives-example
```

Esto creará una carpeta `directives-example` en `src/app` con los archivos necesarios para el componente.

### 4.2. Paso 2: Modificar el componente 'directives-example'

Abre el archivo 'directives-example.component.html' y reemplaza su contenido con el siguiente código:

```
<h2>Directivas Estructurales</h2>
<h3>*ngIf</h3>
<p *ngIf="showParagraph">Este párrafo es visible porque 'showParagraph' es
verdadero. </p>

<h3>*ngFor</h3>
<ul>
```

```

    <li *ngFor="let item of itemsList">{{ item }}</li>
  </ul>

  <h2>Directivas de Atributo</h2>
  <h3>[ngStyle]</h3>
  <p [ngStyle]='{"color": textColor, "font-weight": textBold ? "bold" : "normal"}">
    Este párrafo tiene estilos aplicados dinámicamente.
  </p>

  <h3>[ngClass]</h3>
  <p [ngClass]="{'red-text': applyRedText, 'large-text': applyLargeText}">
    Este párrafo tiene clases aplicadas dinámicamente.
  </p>

```

Ahora, abre el archivo 'directives-example.component.ts' y reemplaza su contenido con el siguiente código:

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-directives-example',
  templateUrl: './directives-example.component.html',
  styleUrls: ['./directives-example.component.css']
})
export class DirectivesExampleComponent {
  showParagraph = true;
  itemsList = ['Item 1', 'Item 2', 'Item 3'];
  textColor = 'blue';
  textBold = true;
  applyRedText = true;
  applyLargeText = false;
}

```

## 4.3. Paso 3: Agregar el componente al módulo principal

Abre 'src/app/app.module.ts' e importa el nuevo componente:

```

import { DirectivesExampleComponent } from './directives-example.component';
import { AppComponent } from './app.component';
import { NgModule } from '@angular/core';

@NgModule({
  declarations: [
    AppComponent,
    DirectivesExampleComponent
  ],
})

```

```
export class AppModule { }
```

## 4.4. Paso 4: Utilizar el componente en 'app.component.html'

Abre el archivo 'app.component.html' y reemplaza su contenido con el siguiente código:

```
<app-directives-example></app-directives-example>
```

## 4.5. Paso 5: Agregar estilos en 'directives-example.component.css'

Abre el archivo 'directives-example.component.css' y agrega los siguientes estilos:

```
.red-text {  
  color: red;  
}  
.large-text {  
  font-size: 24px;  
}
```

## 4.6. Paso 6: Ejecutar la aplicación

Ejecuta el siguiente comando en la terminal para iniciar la aplicación:

```
ng serve
```