

Índice

1. Plantillas	3
1.1. Componentes como directivas.....	3
1.2. Creando un nuevo componente.....	3
1.3. Usando el selector de un componente	4
1.4. Interpolación.....	4
1.5. Directivas estructurales (*ngFor, *ngIf)	5

1. Plantillas

Como hemos dicho antes, una plantilla es la vista asociada a un componente. La clase del componente se encarga de interaccionar y manipular su plantilla.

1.1. Componentes como directivas

Como vimos anteriormente, los componentes tienen un atributo llamado **selector**. Cada vez que situemos esa etiqueta en una plantilla HTML, se instanciará un objeto del componente y se cargará su plantilla asociada dentro de dicho selector.

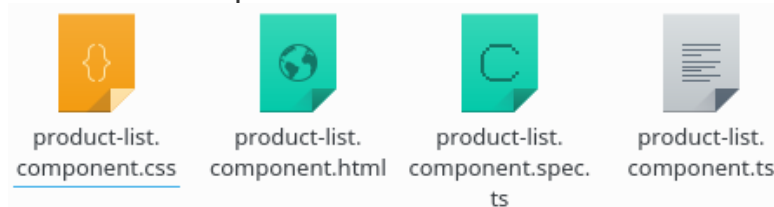
El componente AppComponent se incluye dentro de index.html mediante su selector (la etiqueta **<app-root>**). Vamos a crear un componente llamado **<product-list>** que gestionará una lista de productos y lo incluiremos dentro de AppComponent.

1.2. Creando un nuevo componente

Antes de nada, en el archivo **angular.json**, vamos a establecer un prefijo vacío para los selectores (por defecto es app), o podemos establecer uno más apropiado para nuestra aplicación:

```
"prefix": "",
```

Podemos crear componentes de forma manual o usando Angular CLI. Si vamos al directorio **src/app** y ejecutamos **ng g component product-list**, generará un directorio llamado como el componente con sus archivos:



Vamos a crear la plantilla (product-list.component.html) para el componente que acabamos de crear. Mostraremos una tabla (sólo los encabezados por ahora) donde listaremos los productos.

```
<div>
  <div style="background-color: blue; color: white; padding: 0.75rem">
    My product's list
  </div>
  <div style="padding: 1rem">
    <div>
      <table style="width: 100%">
        <thead>
          <tr style="background-color: lightgrey; text-align: center">
            <th>Product</th>
            <th>Price</th>
            <th>Available</th>
```

```

        </tr>
      </thead>
      <tbody>
        <!-- Aquí van los productos. Por ahora se queda vacío -->
      </tbody>
    </table>
  </div>
</div>
</div>

```

1.3. Usando el selector de un componente

Una vez creado el componente, para que sea visible, debemos incluir su selector (**product-list** en este caso) en la plantilla de otro/s componente/s. En este caso dentro del componente principal de la aplicación (AppComponent).

```

<div style="text-align: center">
  <h1>Welcome to {{ title }}!</h1>
  <product-list></product-list>
</div>

```

Para que lo encuentre debemos importarlo en `app.component.ts`:

```

import { Component } from '@angular/core';
import { ProductListComponent } from './product-list/product-list.component';

@Component({
  selector: 'app-root',
  imports: [ProductListComponent],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  title = 'angu';
}

```

Si ejecutamos `ng serve`, veremos lo siguiente:

Welcome to angu!

My product's list

Product	Price	Available
---------	-------	-----------

1.4. Interpolación

Como vimos en la sección "[Componentes](#)", se utiliza un concepto llamado interpolación para mostrar en la vista las propiedades de un componente. Cuando

ponemos la propiedad dentro de `{{dobles llaves}}`, esta se reemplaza por el valor.

Mediante la interpolación podemos hacer lo mismo que cuando asignamos un valor a alguna variable: concatenar valores, mostrar propiedades, llamar a métodos del componente (se muestra lo que devuelven), cálculos matemáticos, etc.

```
{{title}}
{{'Title: ' + title}}
{{'Title: ' + getTitle()}}
{{4*54+6/2}}
```

Vamos a poner el título de la lista de productos y los encabezados de la tabla en propiedades dentro del componente y mostrarlos en la plantilla mediante interpolación (esto permite modificarlas, traducirlas, etc. fácilmente y en tiempo real). Para ello añadimos en el archivo `product-list.component.ts`:

```
export class ProductListComponent {
  title = 'Mi lista de productos';
  headers = { desc: 'Producto', price: 'Precio', avail: 'Disponible' };
}
```

Ahora, usando interpolación, los vincularemos en la plantilla:

```
<div>
  <div style="background-color: blue; color: white; padding: 0.75rem">
    {{ title }}
  </div>
  <div style="padding: 1rem">
    <div>
      <table style="width: 100%">
        <thead>
          <tr style="background-color: lightgrey; text-align: center">
            <th>{{ headers.desc }}</th>
            <th>{{ headers.price }}</th>
            <th>{{ headers.avail }}</th>
          </tr>
        </thead>
        <tbody>
          <!-- Aquí van los productos. Por ahora se queda vacío -->
        </tbody>
      </table>
    </div>
  </div>
</div>
```

1.5. Directivas estructurales (*ngFor, *ngIf)

Vamos a añadir algunos productos a nuestra tabla. Antes de nada, vamos a decirle a TypeScript qué propiedades (y sus tipos) debe tener un producto, lo que nos permitirá que no nos equivoquemos al definirlos y activará el autocompletado en el editor. Para ello creamos una interfaz llamada `IProduct` donde definiremos dichas

propiedades. Desde el directorio `src/app/` del proyecto ejecutamos:

```
mkdir interfaces
cd interfaces
ng g interface i-product
```

```
export interface IProduct {
  id: number;
  desc: string;
  price: number;
  avail: Date;
  imageUrl: string;
  rating: number;
}
```

Para empezar, mostraremos la tabla sólo cuando haya productos para listar. Si no tenemos productos, la tabla directamente no aparecerá (no estará en el DOM).

Esto se consigue con la directiva de Angular ***ngIf**. Esta directiva se añade como atributo a un elemento y se establece una condición. Cuando esta sea cierta, el elemento se mostrará, y cuando sea falsa, el elemento pasará a estar oculto. En este caso, la condición será que exista el array **products** y no esté vacío.

```
<div *ngIf="products && products.length">
  <table style="width: 100%">
    <thead>
      <tr style="background-color: lightgrey; text-align: center">
        <th>{{ headers.desc }}</th>
        <th>{{ headers.price }}</th>
        <th>{{ headers.avail }}</th>
      </tr>
    </thead>
    <tbody>
    </tbody>
  </table>
</div>
```

Para que las directivas funcionen observa en el código siguiente como se ha importado el módulo **CommonModule**.

Finalmente, vamos a añadir algunos productos. Crearemos un array de productos (definidos por la interfaz **Iproduct**) dentro de **ProductListComponent**. Debemos también importar la interfaz para usarla (el propio editor nos avisará).

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IProduct } from '../interfaces/i-product';
@Component({
  selector: 'product-list',
  imports: [CommonModule],
  templateUrl: './product-list.component.html',
```

```

    styleUrls: ['./product-list.component.css'],
  })
  export class ProductListComponent {
    title = 'Mi lista de productos';
    headers = { desc: 'Producto', price: 'Precio', avail: 'Disponible' };

    products: IProduct[] = [
      {
        id: 1,
        desc: 'SSD hard drive',
        avail: new Date('2016-10-03'),
        price: 75,
        imageUrl: 'assets/ssd.jpg',
        rating: 5,
      },
      {
        id: 2,
        desc: 'LGA1151 Motherboard',
        avail: new Date('2016-09-15'),
        price: 96.95,
        imageUrl: 'assets/motherboard.jpg',
        rating: 4,
      },
    ];
  }
}

```

La directiva ***ngFor** nos permite iterar por esta colección de productos y generar para cada uno el HTML necesario (una fila de la tabla) para mostrarlos.

```

<tbody>
  <tr *ngFor="let product of products">
    <td>{{ product.desc }}</td>
    <td>{{ product.price }}</td>
    <td>{{ product.avail }}</td>
  </tr>
</tbody>

```

Esto equivale a la instrucción **foreach** en muchos lenguajes. Para cada producto de la lista, se crea una variable llamada **product** y se le asignará el objeto correspondiente a cada iteración. La estructura donde se aplica la directiva (<tr>), se repetirá tantas veces como productos haya en el array.

Finalmente nos debe quedar así:

Welcome to angu!

Mi lista de productos		
Producto	Precio	Disponible
SSD hard drive	75	Mon Oct 03 2016 02:00:00 GMT+0200 (hora de verano de Europa central)
LGA1151 Motherboard	96.95	Thu Sep 15 2016 02:00:00 GMT+0200 (hora de verano de Europa central)