60 - TypeScript y Angular: decorador @Component

En Angular, el decorador @Component es esencial para la creación y configuración de componentes. Este decorador se utiliza para asociar metadatos con la clase de un componente, proporciona información sobre su comportamiento, aspecto y configuración.

Los metadatos incluyen propiedades como selector (identificador HTML para el componente), template (código HTML o referencia a un archivo HTML), styleUrls (archivos de estilo o estilos en línea) etc.

Crearemos un proyecto para probar distintas variantes sobre la configuración de los parámetros del decorador @Component.

```
ng new proyecto038
```

Como ya sabemos Angular CLI al crear un proyecto nos crea una primer componente llamada AppComponent:

app.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'proyecto038';
}
```

Ahora que ya conocemos mejor el concepto de decorador, pasemos a ver las distintas propiedades del objeto que le pasamos a la función Component.

La propiedad 'selector' hace referencia a la etiqueta que debemos agregar a nuestra página para crear una componente de dicho tipo:

```
  selector: 'app-root',
```

En este caso se encuentra en el archivo index.html:

```
<body>
  <app-root></app-root>
</body>
```

Otra propiedad que debemos configurar es imports, que se utiliza para declarar otras componentes que hacemos uso, o módulos que contienen componentes, directivas u otros artefactos que el componente actual necesita para funcionar correctamente.

La propiedad templateUrl en el decorador @Component se utiliza para especificar la ubicación del archivo HTML externo que contiene la plantilla del componente. En otras palabras, templateUrl se utiliza para separar el código HTML de la lógica del componente, lo que facilita la organización y mantenimiento del código.

```
  templateUrl: './app.component.html',
```

Otra variante que vamos a probar es eliminar la propiedad 'templateUrl' y proceder a insertar la propiedad 'template' con la plantilla HTML directamente dentro del decorador (podemos eliminar el archivo 'app.component.html'), puede ser útil si nuestra componente es muy sencilla y queremos concentrar la plantilla HTML y la clase en el mismo archivo:

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  template: `<h1>Componente de Angular</h1>
  <p>La plantilla de la componente se encuentra definida
    directamente en el decorador.</p>`,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'proyecto038';
}
```

De forma similar podemos definir la hoja de estilo de la componente en un archivo o un conjunto de archivos CSS separados:

```
  styleUrls: ['./app.component.css']
```

O borrar la propiedad 'styleUrls' por la propiedad 'style':

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  template: `<h1>Componente de Angular</h1>
  <p>La plantilla de la componente se encuentra definida
    directamente en el decorador.</p>`,
  styles: `h1 {
    color:red
  }
  p {
    color:blue
  }`
})
export class AppComponent {
  title = 'proyecto038';
}
```

Es importante hacer notar que la función decoradora 'Component' debemos importarla para su uso (cuando creamos una componente, dicho código ya se ha agregado):

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

Retornar