Componentes en Angular

Creación de Componentes en Angular

¿Qué es un Componente en Angular?

En Angular, un componente es una pieza fundamental que representa una parte de la interfaz de usuario (UI). Cada componente consta de tres partes principales:

1. **HTML**: Define la estructura y el contenido visual del componente.
2. **CSS**: Aplica estilos específicos al componente.
3. **TypeScript**: Contiene la lógica del componente, como propiedades, métodos y el manejo de eventos.

Los componentes son la base de cualquier aplicación Angular, ya que permiten dividir la UI en partes reutilizables y modulares.

Sintaxis Básica para Crear un Componente

Para crear un componente en Angular, se utiliza el decorador `@Component`. Este decorador define los metadatos del componente, como su selector, plantilla, y estilos.

Ejemplo Básico:

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

@Component({
  selector: 'app-ejemplo-basico',
  template: `
    <h1>{{ titulo }}</h1>
    <p>{{ descripcion }}</p>
  `,
  styles: [
    `
      h1 {
        color: blue;
      }
    `,
  ],
})
export class EjemploBasicoComponent {
  titulo = 'Componente Básico';
  descripcion = 'Este es un componente simple en Angular.';
}
```

En este ejemplo:

- `selector`: Define el nombre de la etiqueta HTML que se usará para insertar este componente en otras partes de la aplicación.
- `template`: Contiene el HTML que se mostrará en el componente.
- `styles`: Define los estilos específicos para el componente.

Creación de un Componente Utilizando Angular CLI

La forma más común de crear un componente en Angular es usando Angular CLI, una herramienta de línea de comandos que facilita la creación y gestión de componentes.

Comando:

```
ng generate component nombre-del-componente
```

O usando la sintaxis resumida (g – generate, c – component):

```
ng g c nombre-del-componente
```

Este comando generará los archivos necesarios para el componente:

- Un archivo `.ts` con la lógica del componente.
- Un archivo `.html` con la plantilla del componente.

- Un archivo `.css` para los estilos del componente.
- Un archivo de prueba `.spec.ts`.

Para evitar que se cree este archivo de prueba .spec.ts, pueden agregar el parámetro --skipt-tests. Esta opción la estaremos usando a lo largo de curso:

```
ng g c nombre-del-componente --skip-tests
```

Si desean crear un componente que no sea standalone por default, pueden crear un componente con el parámetro --standalone=false.

```
ng g c nombre-del-componente --standalone=false
```

En este curso estaremos trabajando con componentes standalone, pero ya saben cómo pueden crear tanto proyectos con módulos y ahora componentes que se agreguen al archivo de configuración de módulos como se solía trabajar en versiones anteriores de Angular.

Si tu proyecto se creó con la opción de módulos (ng new mi-app --standalone=false) entonces los componentes que se creen NO serán standalone por default, sino basados en módulos, por lo que en ese caso se puede omitir el parámetro --standalone=false en la creación de los componentes, con esto se agregarán de manera automática al archivo app.module.ts para poder usar de inmediato los nuevos componentes creados.

Eliminar un componente:

No existe un comando específico de Angular CLI (`ng`) para borrar un componente directamente. Sin embargo, puedes eliminar manualmente un componente siguiendo estos pasos:

1. **Eliminar los archivos del componente**: Localiza y elimina los archivos `.ts`, `.html`, `.css` (o `.scss`), y `.spec.ts` correspondientes al componente en tu proyecto Angular.
2. **Eliminar la referencia en el módulo**: Si el componente no es standalone y está declarado en un módulo, abre el archivo del módulo (por ejemplo, `app.module.ts`) y elimina el componente de la lista `declarations`.
3. **Eliminar las referencias en otras partes del proyecto**: Revisa si el componente está siendo utilizado en otras partes del proyecto, como en otras plantillas, y elimina esas referencias para evitar errores.

Aunque estos pasos requieren intervención manual, son los pasos necesarios para asegurarte de que el componente se elimine completamente del proyecto.

Con esto todo lo anterior, podemos entender cómo funcionan los componentes en Angular, así mismo como podemos crearlos con cada una de las opciones más importantes, y finalmente eliminarlos en caso necesario.

Ejemplo de creación de un nuevo componente:

Cremoas una nueva aplicación de angular:

```
ng new app-angular
```

Abrir el folder del proyecto con VSC (Visual Studio Code). Si no estamos dentro de un proyecto, no es posible ejecutar comandos de CLI para crear nuevos componentes.

Crear un nuevo componente, para ello ejecutamos el siguiente comando:

```
ng g c nuevo-componente --skip-tests
```

Angular CLI genera una nueva carpeta llamada nuevo-componente y dentro de esta carpeta varios archivos que conforman el componente `nuevo-componente`. Veamos qué se crea, qué incluye y para qué sirven:

1. `nuevo-componente.component.ts`

Este es el archivo principal del componente, donde se define la clase del componente.

- **Propósito:** Contiene la lógica del componente, incluyendo propiedades y métodos que se utilizarán en la plantilla.
- **Incluye:**
  - La importación del decorador `@Component` desde `@angular/core`.
  - El decorador `@Component` que especifica el selector, la plantilla y los estilos asociados con el componente.
  - La clase `NuevoComponenteComponent`, que es la que contiene la lógica del componente.

Ejemplo de contenido:

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

@Component({
  selector: 'app-nuevo-componente',
  standalone: true,
  imports: [],
  templateUrl: './nuevo-componente.component.html',
  styleUrl: './nuevo-componente.component.css'
})
export class NuevoComponenteComponent {
  tituloNuevoComponente = 'Nuevo componente';
}
```

2. `nuevo-componente.component.html`

Este archivo contiene la plantilla HTML del componente.

- **Propósito:** Define la estructura visual del componente, es decir, el contenido que se mostrará en la página.
- **Incluye:** Etiquetas HTML que definen cómo se verá el componente en la interfaz de usuario.

Ejemplo de contenido:

```
<p>{{tituloNuevoComponente}}</p>
```

3. `nuevo-componente.component.css` (o `.scss`, dependiendo de la configuración)

Este archivo contiene los estilos específicos para este componente.

- **Propósito:** Permite definir estilos CSS que solo afectarán al HTML de este componente.
- **Incluye:** Reglas CSS que se aplican a los elementos definidos en la plantilla HTML.

Ejemplo de contenido:

```
/* Estilos específicos para el componente */
p {
  color: blue;
}
```

4. `nuevo-componente.component.spec.ts` (no generado debido a `--skip-tests`)

Normalmente, Angular CLI también generaría un archivo `.spec.ts` para pruebas unitarias, pero como usaste el parámetro `--skip-tests`, este archivo no se crea.

- **Propósito:** (Si se generara) Contiene pruebas unitarias para el componente, escritas utilizando el framework de pruebas Jasmine. Estas pruebas ayudan a garantizar que el componente funcione correctamente.

5. Actualización en el módulo. Únicamente si es una aplicación de módulos (normalmente `app.module.ts`)

Además de los archivos específicos del componente, Angular CLI agrega automáticamente el componente al módulo donde se debe declarar. Si estás usando módulos (no standalone), el componente se añade a la sección `declarations` del módulo correspondiente. Si estás usando componentes standalone, no hay necesidad de registrar el nuevo componente en ningún otro lugar.

Ejemplo de actualización en `app.module.ts`:

```
import { NuevoComponenteComponent } from './nuevo-componente/nuevo-componente.component';

@NgModule({
  declarations: [
    NuevoComponenteComponent,
    // otros componentes...
  ],
  // ...
})
export class AppModule { }
```

Resumen

- **Archivos Generados:** `nuevo-componente.component.ts`, `nuevo-componente.component.html`, `nuevo-componente.component.css`.
- **Archivos No Generados:** `nuevo-componente.component.spec.ts` (debido a `--skip-tests`).
- **Actualización:** Se añade el componente al módulo correspondiente en el archivo `app.module.ts` (o el módulo específico en el que estés trabajando). Recuerda que esto solo ocurre si estás trabajando con módulos en Angular.

Este proceso facilita la creación de componentes de manera rápida y estructurada, siguiendo las mejores prácticas recomendadas por Angular.

La documentación oficial de Angular para aprender más sobre componentes lo puedes revisar aquí:

https://angular.dev/guide/components

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx