Hola Mundo con Módulos en Angular

A continuación, te mostraremos un ejemplo de cómo crear una aplicación "Hola Mundo" en Angular utilizando módulos (la forma clásica pero que empieza a estar en desuso) en lugar de standalone components. También se explican las diferencias clave entre el uso de módulos y standalone components.

1. Creación del Proyecto Angular

Primero, se debe crear un nuevo proyecto Angular usando Angular CLI, la diferencia entre una aplicación con módulos es que tenemos que agregar el parámetro --standalone=false:

```
ng new hola-mundo-angular --standalone=false
```

2. Estructura del Proyecto

Una vez creado el proyecto, la estructura del proyecto incluirá los siguientes archivos clave:

- **`src/app/app.module.ts`**: El módulo raíz de la aplicación.
- **`src/app/app.component.ts`**: El componente raíz de la aplicación.
- **`src/main.ts`**: El punto de entrada de la aplicación.

3. Código del Módulo Raíz (`AppModule`)

El archivo `app.module.ts` se verá de la siguiente manera:

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

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

4. Código del Componente Raíz (`AppComponent`)

El archivo `app.component.ts` se verá de la siguiente manera:

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

@Component({
  selector: 'app-root',
  template: `
    <h1>{{ title }}</h1>
  `,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Hola Mundo en Angular';
}
```

5. Código del Punto de Entrada (`main.ts`)

El archivo `main.ts` es donde se realiza el bootstrap del módulo:

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

6. Ejecución de la Aplicación

Para ejecutar la aplicación, usa el siguiente comando:

```
ng serve -o
```

Esto iniciará el servidor de desarrollo y podrás ver la aplicación en `http://localhost:4200/`. Deberías ver "Hola Mundo en Angular" en la pantalla.

Diferencias entre el Uso de Módulos y Standalone Components

1. Módulos (`AppModule`):

- **Organización**: Los módulos agrupan componentes, servicios y otros elementos relacionados en una unidad cohesiva. Esto facilita la organización de grandes aplicaciones dividiéndolas en módulos más pequeños y manejables.
- **Declaraciones y Bootstrap**: En los módulos, los componentes deben declararse en la propiedad `declarations` y el módulo raíz define el componente que se debe bootstrap en la propiedad `bootstrap`.
- **Reusabilidad**: Los módulos permiten crear conjuntos de funcionalidades reutilizables que se pueden importar en otros módulos.
- **Inyección de Dependencias**: Los módulos proporcionan un contexto para la inyección de dependencias, definiendo proveedores a nivel de módulo.

2. Standalone Components:

- **Menos Configuración**: En lugar de definir un módulo que agrupe componentes, cada componente puede ser independiente (`standalone: true`). Esto reduce la cantidad de archivos y configuración necesaria, especialmente para aplicaciones pequeñas o componentes simples.
- **Importación Directa**: Los standalone components pueden importar otros componentes y módulos directamente en su configuración, eliminando la necesidad de declararlos en un módulo.
- **Flexibilidad**: Facilita el desarrollo de componentes que pueden ser utilizados en diferentes contextos sin necesidad de estar acoplados a un módulo en particular.
- **Simplificación**: Al eliminar la necesidad de un `AppModule`, el flujo de trabajo para proyectos simples se simplifica, lo que permite una estructura de proyecto más ligera.

Conclusión

- **Con Módulos**: Es ideal para aplicaciones grandes y complejas donde la organización y la reusabilidad de componentes y servicios es crucial. Los módulos proporcionan una estructura clara y escalable.

- **Con Standalone Components**: Es más adecuado para aplicaciones más pequeñas o microfrontends, donde se busca simplicidad y menos configuración. Es un enfoque más moderno que simplifica la creación de componentes independientes.

Ambos enfoques son válidos, y la elección depende del tipo de proyecto y de las necesidades específicas en cuanto a organización, escalabilidad y simplicidad.

¡Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx