

Creación Clase de Producto



Creación de la Clase Producto en Angular con TypeScript

En esta guía, te explicaré paso a paso cómo crear una clase de producto en **Angular** utilizando **TypeScript**, por qué realizamos cada acción y cómo asegurarnos de que todo funcione correctamente.

1 Generar la Clase Producto

Ejecuta el siguiente comando en la terminal dentro de tu proyecto de Angular:

```
ng g class producto --skip-tests
```

◆ Explicación del comando

- `ng g class producto`: Genera una nueva clase llamada **Producto**.
- `--skip-tests`: Evita la creación de un archivo de pruebas (`producto.spec.ts`), ya que en este caso solo necesitamos la clase.

Después de ejecutar el comando, Angular creará el archivo `producto.ts` dentro de la carpeta `src/app`.

2 Modificar la Configuración en `tsconfig.json`

Abrimos el archivo `tsconfig.json` y agregamos o modificamos la siguiente propiedad en la sección `compilerOptions`:

```
"strictPropertyInitialization":false,
```

◆ ¿Por qué hacemos esto?

- Por defecto, TypeScript exige que todas las propiedades de una clase sean inicializadas en el constructor o en su declaración.
- Al establecer `"strictPropertyInitialization": false`, desactivamos esta restricción, permitiéndonos definir propiedades sin inicializarlas inmediatamente.
- Esto es útil en casos como este, donde la inicialización de las propiedades se realizará al asignar datos externos (como una API o una base de datos).

Si no desactivamos esta opción, TypeScript nos obligaría a inicializar cada propiedad en el constructor de la clase, lo cual no siempre es necesario.

Vamos a ver cómo usar el operador `!` para no tener que agregar esta configuración.

3 Definir la Clase Producto (`producto.ts`)

Ahora creamos la clase en el archivo `producto.ts`:

```
export class Producto {  
    idProducto!: number;  
    descripcion!: string;  
    precio!: number;  
    existencia!: number;  
}
```

◆ Explicación del Código

- `export class Producto`: Exportamos la clase para poder usarla en otros archivos.
- `idProducto!: number`: Identificador único del producto (tipo `number`).
- `descripcion!: string`: Breve descripción del producto (tipo `string`).

- `precio!: number;`: Precio del producto (tipo `number`).
- `existencia!: number;`: Cantidad de productos disponibles en el inventario (tipo `number`).
- Usar la propiedad opcional `!` (Definitive Assignment Assertion)
- Si estás seguro de que las propiedades siempre serán asignadas antes de su uso, puedes usar `!:` Con esto evitamos inicializar los valores predeterminados sin necesidad de modificar la opción "strictPropertyInitialization"

◆ ¿Por qué usamos una Clase y no una Interfaz?

- **Las clases** en TypeScript permiten definir métodos y lógica adicional dentro de la misma estructura.
- **Las interfaces** solo definen la estructura de los datos, pero no permiten métodos ni lógica adicional.
- Si en el futuro necesitamos agregar **métodos** a `Producto`, como cálculos o validaciones, podemos hacerlo fácilmente con una **clase**. Sin embargo, si no se requiere agregar más lógica, bien podemos usar una interfaz en lugar de una clase.

Conclusión

- **Creamos una clase en Angular con TypeScript** para representar un producto en un sistema de inventario.
- **Modificamos `tsconfig.json`** para desactivar `strictPropertyInitialization`, evitando la necesidad de inicializar propiedades en el constructor. Esto es opcional, si no queremos modificar este archivo podemos usar el operador `:=` para no tener que inicializar las propiedades de nuestra clase.

 Ahora tienes una base sólida para trabajar con clases en **Angular** y **TypeScript** en un sistema de inventarios. 

Saludos!

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)