Desarrollo de páginas con Angular

Módulo 2 - Lectura Video 01: Inyección de Dependencias

Para registrar una dependencia, tenemos que vincularla a algo que identifique esa dependencia. Esta identificación se llama el token de dependencia. Por ejemplo, si queremos registrar la URL de un API, podemos usar la cadena API_URL como el token. Del mismo modo, si estamos registrando una clase, podemos utilizar la clase en sí misma como su token como veremos a continuación.

La inyección de dependencia en Angular consta de tres piezas:
• el Proveedor (también conocido como enlace) mapea un token (que puede ser una cadena o una clase) a una lista de dependencias. Le dice a Angular cómo crear un objeto, dado un token.
• el inyector que contiene un conjunto de enlaces y es responsable de resolver las dependencias e inyectándolos al crear objetos
• La dependencia que es lo que se inyecta.

Proporcionando Dependencias con NgModule

Si bien es interesante ver cómo se crea directamente un inyector, esa no es la forma típica que usaríamos inyecciones.

En cambio, lo que normalmente haríamos es
• usar NgModule para registrar lo que inyectaremos, se llaman proveedores y
• usar decoradores (generalmente en un constructor) para especificar qué estamos inyectando

Al realizar estos dos pasos, Angular gestionará la creación del inyector y la resolución de las dependencias.

Convirtamos un objeto, supongamos un UserService para ser inyectable como un singleton en nuestra aplicación.
Primero, vamos a añadirlo a la clave de proveedores de nuestro NgModule:

```
@NgModule ({
        providers: [
                UserService // <- añadido aquí
        ],
})
```

Ahora podemos inyectar UserService en nuestro componente simplemente agregándolo en nuestro constructor como un argumento de entrada.

Lo bueno de esto es que Angular está administrando cuándo crear no solo el componente, sino que también se ocupa de crear sus dependencias, y no tenemos que preocuparnos por hacerlo nosotros mismos.

Cada clase que inyecta el UserService, recibirá el mismo objeto, esto se conoce como el patrón de diseño singleton.