ANGULAR

Introducción a los

servicios

y

inyección de dependencias

Angular.io

El servicio **es una categoría amplia que abarca cualquier valor, función o característica que necesita una aplicación. Un servicio es típicamente una clase con un propósito estrecho y bien definido. Debe hacer algo específico y hacerlo bien.**

Angular distingue los componentes de los servicios para aumentar la modularidad y la reutilización. Al separar la funcionalidad relacionada con la vista de un componente de otros tipos de procesamiento, puede hacer que sus clases de componentes sean ágiles y eficientes.

Idealmente, el trabajo de un componente es permitir la experiencia del usuario y nada más. Un componente debe presentar propiedades y métodos para el enlace de datos, para mediar entre la vista (representada por la plantilla) y la lógica de la aplicación (que a menudo incluye alguna noción de *modelo*).

Un componente puede delegar ciertas tareas a los servicios, como obtener datos del servidor, validar la entrada del usuario o iniciar sesión directamente en la consola. Al definir tales tareas de procesamiento en una *clase de servicio inyectable* , usted pone esas tareas a disposición de cualquier

componente. También puede hacer que su aplicación sea más adaptable inyectando diferentes proveedores del mismo tipo de servicio, según corresponda en diferentes circunstancias.

Angular no hace *cumplir* estos principios. Angular lo ayuda a *seguir* estos principios al facilitar la factorización de la lógica de su aplicación en los servicios y hacer que esos servicios estén disponibles para los componentes a través de *la inyección de dependencia* .

ejemplos de servicio

Aquí hay un ejemplo de una clase de servicio que se registra en la consola del navegador.

src / app / logger.service.ts (class)

```
export class Logger {

    log(msg: any)   { console.log(msg); }

    error(msg: any) {
  console.error(msg); }

    warn(msg: any)  {
  console.warn(msg); }

  }
```

Los servicios pueden depender de otros servicios. Por ejemplo, aquí hay un HeroService que depende del Logger servicio y también se usa BackendService para obtener héroes. Ese servicio a su vez podría depender del HttpClient servicio para obtener héroes de forma asíncrona desde un servidor.

src / app / hero.service.ts (clase)

```
export class HeroService {

     private heroes: Hero[] = [];

constructor(

    private backend: BackendService,

      private logger: Logger) { }

getHeroes() {


   this.backend.getAll(Hero).then(
   (heroes: Hero[]) => {

    this.logger.log(`Fetched
   ${heroes.length} heroes.`);


   this.heroes.push(...heroes); //
   fill cache

       });
```

```
    return this.heroes;

  }

}
```

inyección de dependencia (DI)

DI está conectado al marco angular y se usa en todas partes para proporcionar nuevos componentes con los servicios u otras cosas que necesitan. Los componentes consumen servicios; es decir, puede *inyectar* un servicio en un

componente, dándole acceso al componente a esa clase de servicio.

Para definir una clase como un servicio en Angular, use el decorador @Injectable() para proporcionar los metadatos que permiten a Angular inyectarlo en un componente como una *dependencia* . Del mismo modo, use el decorador @Injectable() para indicar que un componente u otra clase (como otro servicio, una tubería o un NgModule) *tiene* una dependencia

- **El *inyector* es el mecanismo principal. Angular crea un inyector para toda la aplicación para usted durante el proceso de arranque e inyectores adicionales según sea necesario. No tiene que crear inyectores.**

- **Un inyector crea dependencias y mantiene un *contenedor* de instancias de dependencia que reutiliza si es posible.**

- **Un *proveedor* es un objeto que le dice a un inyector cómo obtener o crear una dependencia.**

Para cualquier dependencia que necesite en su aplicación, debe registrar un proveedor con el inyector de la aplicación, para que el inyector pueda usar el proveedor para crear nuevas instancias. Para un servicio, el proveedor suele ser la clase de servicio en sí.

> **Una dependencia no tiene que ser un servicio, podría ser una función, por ejemplo, o un valor.**

Cuando Angular crea una nueva instancia de una clase de componente, determina qué servicios u otras dependencias necesita ese componente al observar los tipos de parámetros del constructor.

Por ejemplo, el constructor de HeroListComponent necesita HeroService.

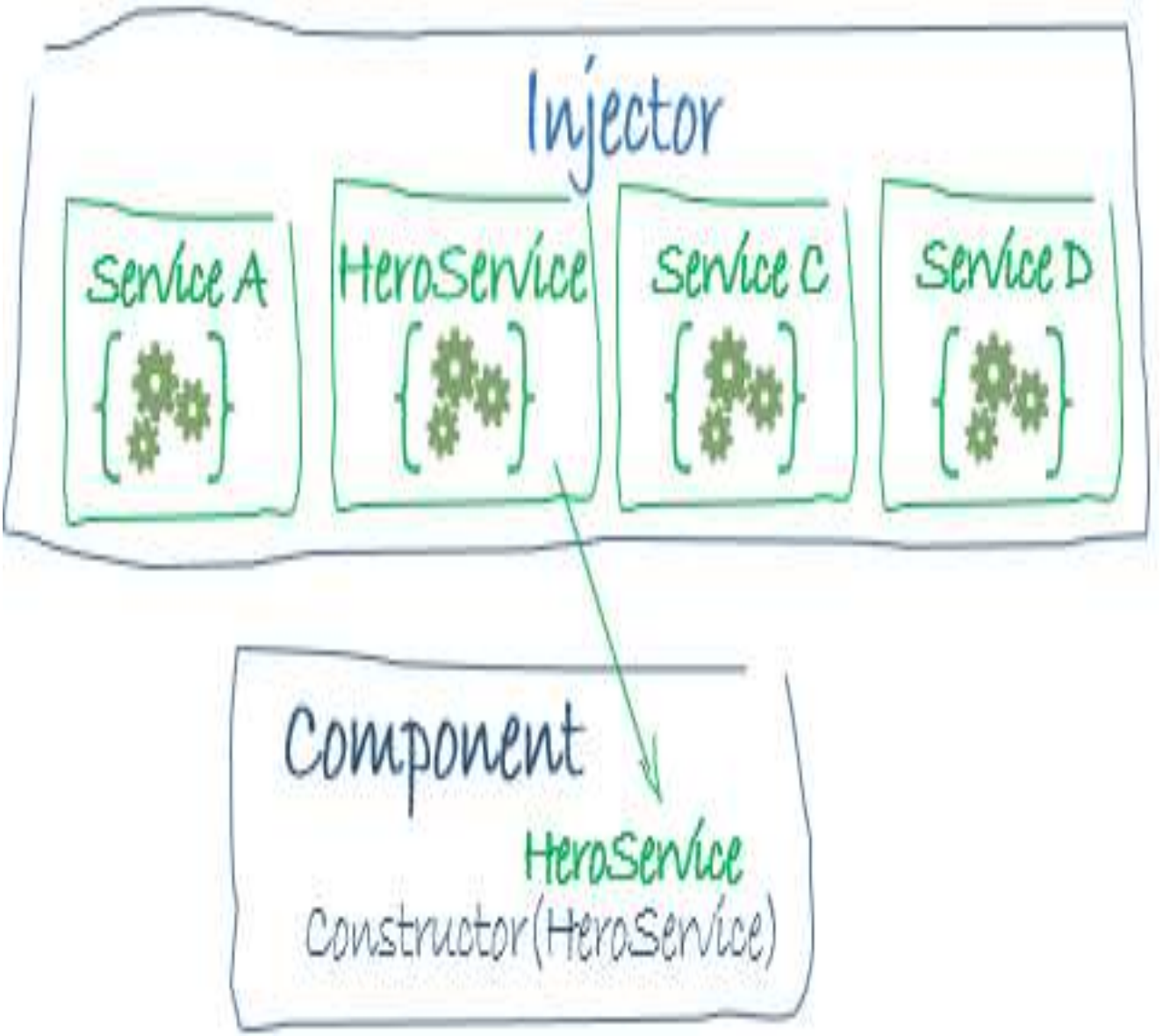
src / app / hero-list.component.ts (constructor)

```
constructor(private service: HeroService) { }
```

Cuando Angular descubre que un componente depende de un servicio, primero verifica si el inyector tiene alguna instancia existente de ese servicio. Si todavía no existe una instancia de servicio solicitada, el inyector hace una utilizando el proveedor registrado y la agrega al inyector antes de devolver el servicio a Angular.

Cuando todos los servicios solicitados se han resuelto y devuelto, Angular puede llamar al constructor del componente con esos servicios como argumentos.

El proceso de HeroServiceinyección se parece a esto.



Proporcionar servicios

Debe registrar al menos un *proveedor* de cualquier servicio que vaya a utilizar. El proveedor puede ser parte de los metadatos del servicio, haciendo que ese servicio esté disponible en todas partes, o puede registrar proveedores con módulos o componentes específicos. Se registran los proveedores en los metadatos del servicio (en el decorador @Injectable()), o en @NgModule() o los metadatos @Component ():

- **De manera predeterminada, el comando Angular CLI ng generate serviceregistra un proveedor con el inyector raíz para su servicio al incluir metadatos de proveedor en el decorador @Injectable()**
  . El tutorial utiliza este método para registrar el proveedor de la definición de clase HeroService.

```
@Injectable({
    providedIn: 'root',
    })
```

Cuando proporciona el servicio en el nivel raíz, Angular crea una única instancia compartida HeroService y la inyecta en cualquier clase que lo solicite. El registro del proveedor en los metadatos @Injectable() también le permite a Angular optimizar una aplicación al eliminar el servicio de la aplicación compilada si no se usa.

- **Cuando registra un proveedor con un NgModule específico , la misma instancia de un servicio está disponible para todos los componentes en ese NgModule. Para registrarse en este nivel, use `providers` property of the `@NgModule()` decorator**

```
@NgModule({
    providers: [
    BackendService,
    Logger
  ],
  ...
   })
```

- **Cuando registra un proveedor en el nivel de componente, obtiene una nueva instancia del servicio con cada nueva instancia de ese componente. En el nivel de componente, registre un proveedor de servicios en la providerspropiedad de los metadatos @Component()**

  src / app / hero-list.component.ts (proveedores de componentes)

```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
```

Para obtener información más detallada, consulte la sección Inyección de dependencias .