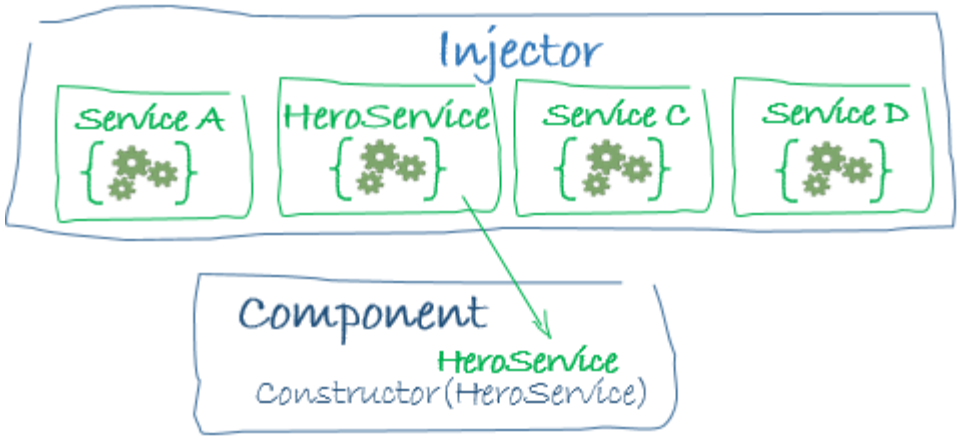
netmind
a bts company

Angular 14 - 10

Services. Dependency Injection

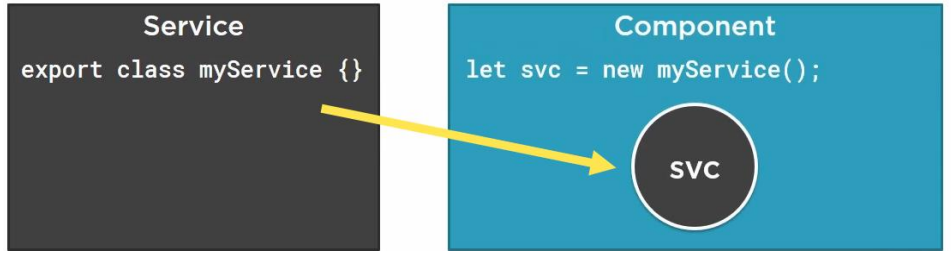
Angular services



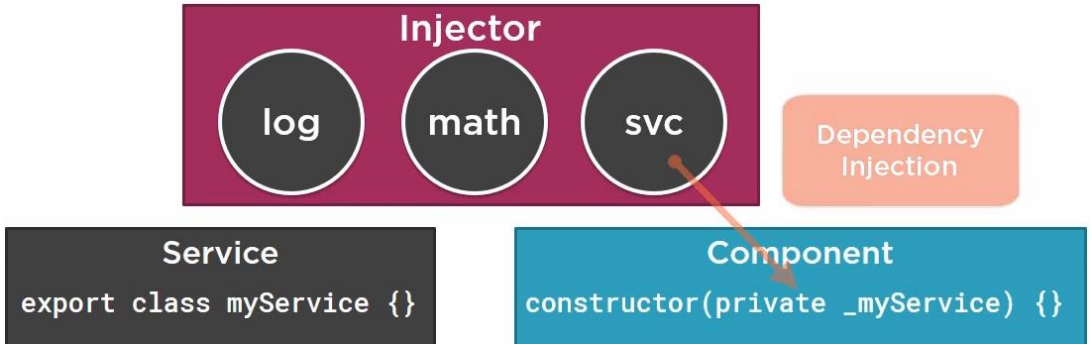
- Components are generally used only for rendering purposes, to define what appears on the user interface.
  - Ideally, other tasks, like data and image fetching, network connections, database management, are not performed.
  - Services take care of this. They perform all the operational tasks for the components.
- A service in Angular is a class that provides a certain functionality to other components.
- Its anotated with **@injectable** decorator.
- Clients of services can use them in two different ways:
  - As an instance (local level)
  - Injected (application level)

Using services

1. **Local instance**: we create an instance in the component.



2. **Injection:** The preferred way to work with services is by registering them at the application level.
    1. In this case, we can request the service by dependency injection in any component and we can share data and resources.
    2. It is equivalent to having a common area of resources for the application, managed by an "injector" which supplies "singleton" instances of those services to those clients.



Service Coding

- Injection requires the **declaration of an object of the type of the service**, and it is up to the injection system to supply it.
  - By doing it **in the constructor**, the functionality of the service will be present in the scope of the class.
- When we create a service we must mark it with the decorator that allows it to be handled by injection: **@injectable** .
  - Strictly speaking, it is not essential to mark it this way, unless -it- has other injection dependencies, but it is considered a good practice to do it this way.
  - Also, everything declared in the service is considered to be public by default.
  - If we want to protect that information, we must declare it as private or protected.
- When creating it, we will have to import all the elements it needs, just as we have seen in previous modules.

📎 **To create a service we will use the command:**

```
ng g s [service_name]
ng g s services/Products
```

Service Coding

- In the class we will add the functionalities we want to export to be used in components, pipes, directives or other services.

```
import { Injectable } from '@angular/core';
import { IProduct } from '../models/iproduct';

@Injectable({
  providedIn: 'root'
})
export class ProductsService {

  private _products: IProduct[] = [...];

  constructor() { }

  public getProducts=():IProduct[]=>{...}

  public deleteProduct=(code:string):boolean =>{...}

  ...

}
```

Registration of a service

- To register a service, we have to register a **Provider**.
- A provider is simply code that creates or returns a service.
  - To register a provider we must define it as **part of the component metadata or in the module**.
- In this way, the injector can handle it from the component or module where it is declared and also for any of its hierarchical descendants.
  - Hence, the site where it is registered is important and that always depends on the functionality that we want to give to the service according to the architecture of the application.
- There is also the possibility of indicating in the service itself who should create the service through the **providedIn** parameter.

```
@Injectable({
  providedIn: 'root'
})
export class ProductsService { ... }
```

  - In this case we indicate that it is the **root application injector** that injects the service and therefore it **is not necessary to declare it in the module**.

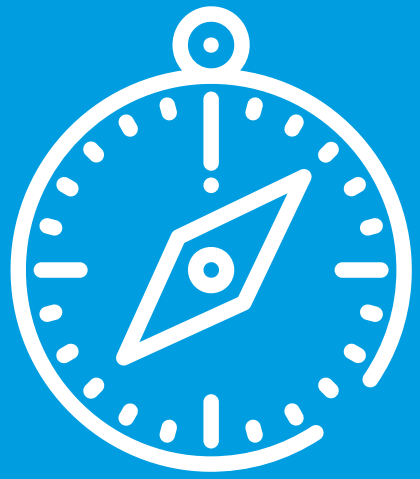
Using a registered service

- To use the service, we must **declare it in the metadata** of the component that uses it:
  - import it, and define the service by injection in the constructor of the corresponding class.
  - To start it once defined, we can use the ngOnInit event , which we previously only used to display something on the console.
  - Finally, we will delete the "embedded" data in the products property and empty other properties of the same type.

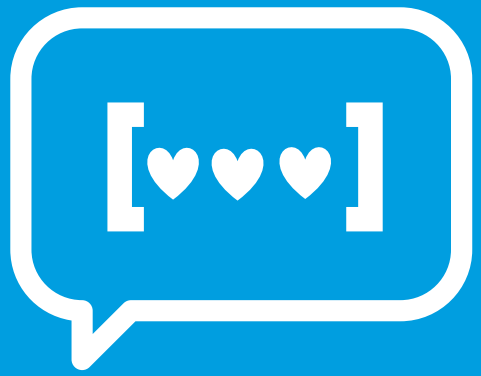
```
import { ProductsService } from
'src/app/services/products.service';

@Component()
export class ProductsListComponent{

  products: IProduct[]=[];

  constructor(private _productsService:ProductsService){
  }

  ngOnInit() {
    console.log(`Spy #${this.products} onInit`);
    this.products=this._productsService.getProducts();
  }

}
```

Let's put it into practice: Tasks/Projects App

1. Create a service for tasks that stores task data and task functionalities (like delete, duplicate, etc).
    1. Do the same for projects.
2. Create an utily service for processing the data filtering of different types of objects. Use it in the filter component.



Next steps



We would like to know your opinion!

Please, let us know what you think about the content.
From Netmind we want to say thank you, we appreciate time and effort you have taking in answering all of that is important in order to improve our training plans so that you will always be satisfied with having chosen us
quality@netmind.es

netmind
a bts company

Thanks!

Follow us: