Providers and dependency injection

**Dependency:**A dependency would be a service, function, object or value that a class needs to perform its function.(example api service)

**Injector:**An injector is responsible for creating dependency instances and injecting them into classes like component. Angular creates an application-wide injector for you during the bootstrap process, and additional injectors as needed. You don’t have to create injectors.(injectable decorator) introduced in angular 6

📖 What we’re going to learn

1. Dependency Injection (DI) recapitulation (optional😉)
2. The Old Way™ of doing DI in Angular — providers: []
3. The New Way™ of doing DI in Angular — providedIn: 'root' | SomeModule
4. Possible scenarios when using providedIn
5. Recommendation on how to use new syntax in your projects
6. Summary

Old way dependency injection with providers:

Example1

Product service—this gona inject to the appcomponent.ts

import {Product} from './Product'—predefined from library

export class ***ProductService***{------class name

    public  ***getProducts()*** {----------method name

        let products:Product[];

        products=[

            new Product(1,'Memory Card',500),

            new Product(1,'Pen Drive',750),

            new Product(1,'Power Bank',100)

        ]

        return products;

    }

}

App component

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

import { ProductService } from './product.service';---see we import here

import { Product } from './product';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

})

export class AppComponent

{

   products:Product[];

   productService;

   constructor(){

***this.productService=new ProductService***();--- we create th reference to the product services to access the all its menthods and variables

   }

   getProducts() {

     this.products=this.productService.getProducts();

   }

}

So the above example tells the old way of creting the references of the parent class(product service) by using the new keyword and access the it’s methods and variables.

Another way by creating the reference in the constructor method:

Example2:

import { ProductService } from './product.service'

export class AppComponent {

   products:Product[];

   constructor(private productService:ProductService) {    -- simply create the reference of the product service in the constructor methid.

   }

   getProducts() {

       this.products=this.productService.getProducts();

   }

}

Next level way of declaration in component level declaration

here we use Providers[] array declaration in the **app component** to access the product service

|  |  |
| --- | --- |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | import { Component } from '@angular/core';    import { ProductService } from './product.service';----as usaul imported here  import { Product } from './product';    @Component({    selector: 'app-root',    templateUrl: './app.component.html',  ***providers: [ProductService]-------this is main thing to access the product services methide and variables for this component only***  })  export class AppComponent  {       products:Product[];       constructor(private productService:ProductService){     }       getProducts() {       this.products=this.productService.getProducts();     }    } |

## **Injecting Service into Service using @injectable{()}**

1. Product service
2. Quality service

Quality service.ts

import { Injectable } from '@angular/core';

@Injectable()-------- indicates it is service

export class Quality {

  log(message:any) {

    console.log(message);

  }

}

Product service.ts

|  |  |
| --- | --- |
| 3  4 | @Injectable()------indicate service  export class ProductService{} |

Next, In the constructor of the ProductService ask for the QualityService

Productservice .ts file: injecting the quality service here

constructor(private Qualityservice: qualityservice) {

    this.qualityservice.log("Product Service Constructed");

}

Before this

we need to register Qualityservice  with the Providers metadata.

Angular  does not have any options add providers in the Service Class. The Providers must be added to the Component/Directive/Pipe

Open the AppComponent Update the Providers array to include Qualityservice

|  |  |
| --- | --- |
| import { Component } from '@angular/core';    import { ProductService } from './product.service';  import { Product } from './product';  import{qualityservice} from’/.qualityservice’;    @Component({    selector: 'app-root',    templateUrl: './app.component.html',    providers: [ProductService,Qualityservice]----here there should be register of two services  })  export class AppComponent  {        } |  |

Disadvantage; In the above all, we registered the dependencies in the Providers array of the component class. The dependencies are only available to the component where it is registered and to its child components only.( Across the application to inject into the multiple components not possible).

## **Using NgModule to Provide Dependencies**

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    HttpModule,

    FormsModule

  ],

  providers: [ProductService,Qualityservice],-------this is the declaration

  bootstrap: [AppComponent]

})

We are Providing the service in the root module will create a single, shared instance of service and injects into any class that asks for it.

Disadvantage :of declaration of providers in ngmodule

If we want to share one instance of a service across the entirety of our application we configure it on our NgModule.

providers: [ProductService]

If we want to share one instance of a service across the entirety of our application we configure it on our NgModule

Explanation:

Since we’ve bound the input field(changing the product services array values) directly to the productservice **and** it’s the same instance(its reference used int the other components constructors) of product service used everywhere, then when we change the one value of array it automatically updates the components and also the child components.

ProvidedIn from Angular 6 to over come previous problems.

import {Injectable} from '@angular/core';

import {HttpClient} from '@angular/common/http';

@Injectable({

    providedIn:'root',--root indicates it pointing the root module nothing but appmodule.ts

})

export class apiservice{

public id:any;

    constructor(private http:HttpClient)

    {

}