**Understanding Dependency Injection (DI) in Angular**

**Dependency Injection** is a design pattern in which a class receives its dependencies from external sources rather than creating them itself. Angular uses DI extensively to provide services and objects.

**✅ Constructor-based Hierarchical Injector**

Angular uses a **hierarchical injector tree**, which means each component can have its own injector, forming a hierarchy.

**Example:**

@Injectable()

export class LoggerService {

log(message: string) {

console.log('Logger:', message);

}

}

@Component({

selector: 'app-child',

template: `Child Component`,

providers: [LoggerService] // This creates a new instance (not singleton)

})

export class ChildComponent {

constructor(private logger: LoggerService) {

this.logger.log('Child component initialized');

}

}

-------

@Component({

selector: 'app-root',

template: `<app-child></app-child>`,

})

export class AppComponent {

constructor(private logger: LoggerService) {

this.logger.log('App component initialized');

}

}

If LoggerService is provided only in AppModule (root), both components share the same instance (singleton). If declared in @Component.providers, new instance per component (non-singleton).

**♻️ Singleton vs Non-Singleton Services**

| **Provided In** | **Instance Scope** |
| --- | --- |
| @Injectable({providedIn: 'root'}) | Singleton app-wide |
| providers: [Service] in component | New instance per component |

**🔄 Injector Class (Manual Injection)**

Use Injector to get dependencies programmatically:

constructor(private injector: Injector) {}

ngOnInit() {

const service = this.injector.get(LoggerService);

service.log('Logger manually injected!');

}

**🧪 Summary Table**

| **Feature** | **Description** |
| --- | --- |
| @Injectable() | Makes class injectable |
| Hierarchical Injector | Creates scoped services at component levels |
| Singleton Services | Shared instance across app |
| Non-Singleton Services | New instance per component |
| @Inject() | Manually inject token |
| Injector.get() | Programmatic DI |

Here’s a working **Angular project structure** demonstrating:

✅ Singleton vs Non-Singleton services  
✅ Use of @Injectable(), @Inject(), @Optional()  
✅ Bonus: Manual injection using Injector

**🧩 Project Structure**

src/

├── app/

│ ├── app.component.ts

│ ├── singleton/

│ │ ├── singleton.component.ts

│ │ └── counter.service.ts

│ ├── nonsingleton/

│ │ ├── nonsingleton.component.ts

│ │ └── counter.service.ts

│ ├── token/

│ │ ├── token.component.ts

│ │ ├── token.service.ts

│ │ └── injection.token.ts

│ └── app.module.ts

**📌 counter.service.ts (Singleton)**

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

@Injectable({ providedIn: 'root' })

export class CounterService {

count = 0;

increment() {

this.count++;

}

}

**📌 singleton.component.ts**

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

import { CounterService } from '../singleton/counter.service';

@Component({

selector: 'app-singleton',

template: `

<h3>Singleton Counter: {{ counter.count }}</h3>

<button (click)="increase()">Increment</button>

`

})

export class SingletonComponent {

constructor(public counter: CounterService) {}

increase() {

this.counter.increment();

}

}

**📌 counter.service.ts (Non-Singleton)**

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

@Injectable()

export class CounterService {

count = 0;

increment() {

this.count++;

}

}

**📌 nonsingleton.component.ts**

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

import { CounterService } from './counter.service';

@Component({

selector: 'app-nonsingleton',

template: `

<h3>Non-Singleton Counter: {{ counter.count }}</h3>

<button (click)="increase()">Increment</button>

`,

providers: [CounterService] // Creates new instance per component

})

export class NonSingletonComponent {

constructor(public counter: CounterService) {}

increase() {

this.counter.increment();

}

}

**📌 injection.token.ts**

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

export const API\_URL = new InjectionToken<string>('api.url');

**📌 token.service.ts**

import { Injectable, Inject, Optional } from '@angular/core';

import { API\_URL } from './injection.token';

@Injectable()

export class TokenService {

constructor(@Optional() @Inject(API\_URL) private apiUrl?: string) {

console.log('API\_URL:', this.apiUrl || 'Not provided');

}

}

**📌 token.component.ts**

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

import { TokenService } from './token.service';

@Component({

selector: 'app-token',

template: `<p>Check console for @Inject()/@Optional() output</p>`,

providers: [TokenService]

})

export class TokenComponent {

constructor(tokenService: TokenService) {}

}

**📌 app.module.ts**

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

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

import { SingletonComponent } from './singleton/singleton.component';

import { NonSingletonComponent } from './nonsingleton/nonsingleton.component';

import { TokenComponent } from './token/token.component';

import { API\_URL } from './token/injection.token';

@NgModule({

declarations: [AppComponent, SingletonComponent, NonSingletonComponent, TokenComponent],

imports: [BrowserModule],

providers: [

{ provide: API\_URL, useValue: 'https://api.example.com' }

],

bootstrap: [AppComponent]

})

export class AppModule {}

**📌 app.component.html**

<h2>Angular DI Demo</h2>

<app-singleton></app-singleton>

<app-nonsingleton></app-nonsingleton>

<app-token></app-token>