Dependency Injection

NestJS

Dependency Injection

In software engineering, dependency injection is a technique in which an object receives other objects that it depends on, called dependencies.

Dependencies are services or objects that a class needs to perform its function. Dependency injection, or DI, is a design pattern wherein a class requests dependencies from external sources rather than creating them.

- Service - the injected object
- Interface - the contract
- Client - the injectable object
- Injector - the utility responsible for injection

NestJS Modularity

- The angular module is a logical container
- IOC
- Providers

Providers

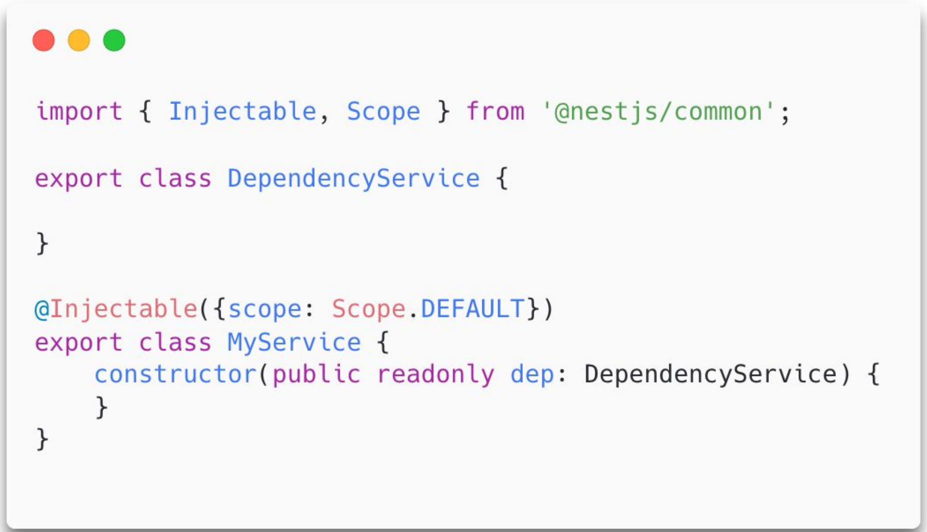
A provider object defines how to obtain an injectable dependency associated with a DI token.

An injector uses the provider to create a new dependency instance for a class that requires it.

- Value Provider
- Type Provider
- Class Provider
- Existing Provider
- Factory Provider
- Type Provider

Injectable Decorators

Injectable decorator gives a synthetic sugar for creating a type provider.



```
import { Injectable, Scope } from '@nestjs/common';

export class DependencyService {

}

@Injectable({scope: Scope.DEFAULT})
export class MyService {
    constructor(public readonly dep: DependencyService) {
    }
}
```

Scopes and Injector Hierarchy

- Default scope
- Request scope
- Transient scope

NestJS vs. Angular DI

- Providers signature
- Resolving dependencies

```
process.exit(0);
```