Services in Angular

Dependency Injection (DI) is a mechanism where the required resources will be injected into the code automatically.

Angular comes with an in-built dependency injection subsystem.

**Why Dependency Injection?**

It is because DI:

* allows developers to reuse the code across applications.
* makes the code loosely coupled.
* makes application development and testing much easier.
* allows the developer to ask for the dependencies from Angular. There is no need for the developer to explicitly create/instantiate them.

Services Basics

A service in Angular is a class that contains some functionality that can be reused across the application. A service is a singleton object. Angular services are a mechanism of abstracting shared code and functionality throughout the application.

Angular Services come as objects which are wired together using dependency injection.

Angular provides a few inbuilt services also can create custom services.

**Why Services?**

Services can be used to:

* share the code across components of an application.
* make HTTP requests.

**Creating a Service**

To create a service class, use the following command:

1. ng generate service book

The above command will create a service class as shown below:

1. @Injectable({
2. providedIn:'root'
3. })
4. export class BookService
5. {
6. }

@Injectable() decorator makes the class injectable into application components.

**Providing a Service**

Following are the ways to provide services in an Angular application:

1. The first way to register service is to specify providedIn property using @Injectable decorator. This property is added by default when you generate a service using Angular CLI.

1. import { Injectable } from '@angular/core';
2. @Injectable({
3. providedIn: 'root'
4. })
5. export class BookService {}

Line 4: providedIn property registers BookService at the root level  (app module).

When the BookService is provided at the root level, Angular creates a singleton instance of the service class and injects the same instance into any class that uses this service class. In addition, Angular also optimizes the application if registered through providedIn property by removing the service class if none of the components use it.

2. Services can also be provided across the application by registering it using the providers property in the @Ngmodule decorator of any module.

1. @NgModule({
2. imports: [BrowserModule],
3. declarations: [AppComponent, BookComponent],
4. providers: [BookService],
5. bootstrap: [AppComponent]
6. })

Line 4: When the service class is added in the providers property of the root module, all the directives and components will have access to the same instance of the service.

3. There is also a way to limit the scope of the service class by registering it in the providers' property inside the @Component decorator. Providers in component decorator and module decorator are independent. Providing a service class inside a component creates a separate instance for that component and its nested components.

1. import { BookService } from './book/book.service';
2. @Component({
3. selector: 'app-root',
4. styleUrls: ['./app.component.css'],
5. templateUrl: './app.component.html',
6. providers:[BookService]
7. })

**Injecting a Service**

The only way to inject a service into a component/directive or any other class is through a constructor.

Add a constructor in a component class with service class as an argument as shown below:

1. constructor(private bookService: BookService){ }

BookService will then be injected into the component through constructor injection by the framework.

**Best Practices - Coding Style Rules**

 Use services for sharing data and functionality.

 Use a service with a single responsibility to make the testing easier.

# Demo : Services

**Highlights:**

* Creating a service
* Injecting a service into a component

**Demosteps:**

**Problem Statement**: Create a Book Component which fetches book details like id, name and displays them on the page in a list format. Store the book details in an array and fetch the data using a custom service. The output is as shown below



1. Create **BookComponent**by using the following CLI command

1. D:\MyApp>ng generate component book

2. Create a file with the name**book.ts** under the book folder and add the following code.

1. export class Book {
2. id!: number;
3. name!: string;
4. }

3. Create a file with the name **books-data.ts** under the book folder and add the following code.

1. import { Book } from './book';
2. export let BOOKS: Book[] = [
3. { id: 1, name: 'HTML 5' },
4. { id: 2, name: 'CSS 3' },
5. { id: 3, name: 'Java Script' },
6. { id: 4, name: 'Ajax Programming' },
7. { id: 5, name: 'jQuery' },
8. { id: 6, name: 'Mastering Node.js' },
9. { id: 7, name: 'Angular JS 1.x' },
10. { id: 8, name: 'ng-book 2' },
11. { id: 9, name: 'Backbone JS' },
12. { id: 10, name: 'Yeoman' }
13. ];

4. Create a service called **BookService** under the book folder using the following CLI command

1. D:\MyApp\src\app\book>ng generate service book

5. Add the following code in **book.service.ts**

1. import { Injectable } from '@angular/core';
2. import { BOOKS } from './books-data';
3. @Injectable({
4. providedIn: 'root'
5. })
6. export class BookService {
7. getBooks() {
8. return BOOKS;
9. }
10. }

6. Add the following code in the **book.component.ts** file

1. import { Component, OnInit } from '@angular/core';
2. import { Book } from './book';
3. import { BookService } from './book.service';
4. @Component({
5. selector: 'app-book',
6. templateUrl: './book.component.html',
7. styleUrls: ['./book.component.css']
8. })
9. export class BookComponent implements OnInit {
10. books!: Book[];
11. constructor(private bookService: BookService) { }
12. getBooks() {
13. this.books = this.bookService.getBooks();
14. }
15. ngOnInit() {
16. this.getBooks();
17. }
18. }

7. Write the below-given code in**book.component.html**

1. <h2>My Books</h2>
2. <ul class="books">
3. <li \*ngFor="let book of books">
4. <span class="badge">{{book.id}}</span> {{book.name}}
5. </li>
6. </ul>

8. Add the following code in**book.component.css**  which has styles for books

1. .books {
2. margin: 0 0 2em 0;
3. list-style-type: none;
4. padding: 0;
5. width: 13em;
6. }
7. .books li {
8. cursor: pointer;
9. position: relative;
10. left: 0;
11. background-color: *#eee;*
12. margin: 0.5em;
13. padding: 0.3em 0;
14. height: 1.5em;
15. border-radius: 4px;
16. }
17. .books li:hover {
18. color: *#607d8b;*
19. background-color: *#ddd;*
20. left: 0.1em;
21. }
22. .books .badge {
23. display: inline-block;
24. font-size: small;
25. color: white;
26. padding: 0.8em 0.7em 0 0.7em;
27. background-color: *#607d8b;*
28. line-height: 0.5em;
29. position: relative;
30. left: -1px;
31. top: -4px;
32. height: 1.8em;
33. margin-right: 0.8em;
34. border-radius: 4px 0 0 4px;
35. }

9. Add the following code in app.component.html

1. <app-book></app-book>

10. Save the files and check the output in the browser