Angular TS – Services and Dependency Injection(DI)

[**✅ Task 5: Services and Dependency Injection** 1](#_Toc204790524)

[**1️⃣ Purpose** 1](#_Toc204790525)

[**2️⃣ Theory** 1](#_Toc204790526)

[**3️⃣ Prerequisites** 1](#_Toc204790527)

[**4️⃣ Code Example (Step-by-Step)** 1](#_Toc204790528)

[**5️⃣ Project Structure Snapshot** 3](#_Toc204790529)

[**✅ 6️⃣ Summary** 3](#_Toc204790530)

**✅ Task 5: Services and Dependency Injection**

**1️⃣ Purpose**

To understand how to:

* Create and use **Angular services**.
* Inject services using **Angular's Dependency Injection (DI)** system in **Angular 18 standalone setup**.

**2️⃣ Theory**

| **Concept** | **Description** |
| --- | --- |
| Service | A reusable TypeScript class to handle business logic, data fetching, or state sharing. |
| Dependency Injection (DI) | Angular’s way of providing required services automatically to components. |
| @Injectable() | Decorator to make a class injectable. |
| providedIn: 'root' | Registers the service in the root injector (singleton pattern). |

**3️⃣ Prerequisites**

* Angular 18 with standalone component structure.
* Prior routing and component setup.

**4️⃣ Code Example (Step-by-Step)**

**✅ a) Create the Service**

ng generate service services/student

**✅ b) student.service.ts**

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

@Injectable({

providedIn: 'root'

})

export class StudentService {

private students: string[] = ['Kiran', 'Arjun', 'Divya'];

getStudents(): string[] {

return this.students;

}

addStudent(name: string) {

this.students.push(name);

}

}

**✅ c) Create Component to Use the Service**

ng generate component components/student-list --standalone

**✅ d) student-list.component.ts**

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

import { CommonModule } from '@angular/common';

import { StudentService } from '../../services/student.service';

@Component({

selector: 'app-student-list',

standalone: true,

imports: [CommonModule],

template: `

<h2>Student List</h2>

<ul>

<li \*ngFor="let student of students">{{ student }}</li>

</ul>

<input [(ngModel)]="newStudent" placeholder="Enter name">

<button (click)="addStudent()">Add Student</button>

`

})

export class StudentListComponent {

students: string[] = [];

newStudent = '';

constructor(private studentService: StudentService) {}

ngOnInit() {

this.students = this.studentService.getStudents();

}

addStudent() {

if (this.newStudent.trim()) {

this.studentService.addStudent(this.newStudent.trim());

this.students = this.studentService.getStudents(); // Refresh list

this.newStudent = '';

}

}

}

**✅ e) Add to app.routes.ts**

import { StudentListComponent } from './components/student-list/student-list.component';

export const appRoutes: Routes = [

// ... existing routes

{ path: 'student-list', component: StudentListComponent }

];

**5️⃣ Project Structure Snapshot**

src/

├── app/

│ ├── components/

│ │ └── student-list/

│ │ └── student-list.component.ts ✅

│ ├── services/

│ │ └── student.service.ts ✅

│ ├── app.routes.ts ✅

│ └── app.config.ts ✅

└── main.ts ✅

**✅ 6️⃣ Summary**

* Services allow **code reusability** and centralized business logic.
* Angular injects services using **constructor injection**.
* providedIn: 'root' makes the service a **singleton**.
* Used ngModel for 2-way binding and interacted with the service.