Angular Services

Efficient Code Structure & Service Sharing

Mayooran
Senior Tech Lead, Unicom SD

Unicom TIC
19/10/2024

Prerequisites

- Basic understanding of Angular Framework [Component, Module]
- HTML & CSS
- JavaScript/TypeScript Knowledge
- Angular Directives and Data Binding
- Working with Templates
- Familiarity with Angular CLI
- Angular Pipes
- Angular Routing
- Component Communication

Learning Objectives

- Introduction to Angular Services
- Creating Angular Services
- What is Dependency Injection (DI)
- Injecting a Service into a Component

Introduction to Angular Services

- **What is an Angular Service?**
  - A service is a class in Angular that provides reusable logic and data.
  - Used to share data and encapsulate logic between components.
- **Why Use Services?**
  - Centralizes logic (reduces code duplication).
  - Enables **code reusability and maintainability**.

Service Use Cases

- **Common Examples of Services**
  - **HTTP requests**: Fetching data from APIs.
  - **State management**: Storing user or app state.
  - **Utility functions**: Logic that is used across components, like date/time utilities

Creating a Service in Angular

Generating a Service:

```
ng generate service service-name
```

Structure of a Service

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

@Injectable({
  providedIn: 'root'
})
export class ExampleService {
  constructor() { }

  getData() {
    return 'Sample Data';
  }
}
```

What is Dependency Injection (DI)?

Definition:

- DI is a design pattern that allows objects (dependencies) to be provided rather than created by the class itself.

How it Helps:

- Reduces tight coupling between components.
- Improves testability by making dependencies easily configurable.

Injecting a Service into a Component

1. Import the Service:

```
import { ExampleService } from './example.service’;
```

2. Inject in Constructor:

```
constructor(private exampleService: ExampleService) {
}

ngOnInit() {
      console.log(this.exampleService.getData());
}
```

Benefits of DI in Angular

- **Loose Coupling:** Components don't need to know how dependencies are created.
- **Reusability:** Services can be reused across multiple components or modules.
- **Testability:** Dependencies can be mocked or substituted easily during testing.

Any Questions?