ChatGPT - nqt

Bài 8: Angular Services và Dependency Injection

1. Giới thiệu về Angular Services

1.1. Angular Service là gì?

- **Service** trong Angular là một lớp (class) dùng để chứa logic xử lý dữ liệu và có thể được sử dụng lại ở nhiều component khác nhau.
- **Mục đích chính**:
  - Tách biệt logic nghiệp vụ ra khỏi component.
  - Tái sử dụng code một cách hiệu quả.
  - Hỗ trợ Dependency Injection (DI) giúp quản lý trạng thái dễ dàng hơn.

1.2. Khi nào nên sử dụng Service?

- Khi cần chia sẻ dữ liệu giữa nhiều component.
- Khi cần gọi API và xử lý dữ liệu trước khi gửi đến component.
- Khi cần thực hiện các tác vụ như log, lưu trữ dữ liệu, hoặc xử lý logic chung.

---

2. Tạo Service trong Angular

Angular cung cấp lệnh **Angular CLI** để tạo service một cách nhanh chóng:

```
sh
```

```
ng generate service services/logger
```

Lệnh này sẽ tạo hai file:

- `logger.service.ts`: Chứa code logic của service.
- `logger.service.spec.ts`: File test cho service.

2.1. Cấu trúc một Service

Dưới đây là một ví dụ về `LoggerService` dùng để log thông tin:

`logger.service.ts`

```
typescript
```

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

@Injectable({
  providedIn: 'root' // Đăng ký service ở cấp ứng dụng
})
export class LoggerService {
  log(message: string): void {
    console.log(`[LoggerService] ${message}`);
  }
}
```

Giải thích:

`@Injectable({ providedIn: 'root' })` giúp Angular tự động quản lý service này trong toàn bộ ứng dụng.

---

3. Sử dụng Service trong Component

3.1. Inject Service vào Component

Giả sử chúng ta có một component **AppComponent**, ta sẽ sử dụng `LoggerService` để log dữ liệu.

`app.component.ts`

```
typescript
```

```typescript
import { Component } from '@angular/core';
import { LoggerService } from './services/logger.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  constructor(private logger: LoggerService) {}

  logMessage(): void {
    this.logger.log('Đây là một log message từ AppComponent!');
  }
}
```

`app.component.html`

```
html
```

```html
<button (click)="logMessage()">Ghi Log</button>
```

Giải thích:

- Inject `LoggerService` vào constructor của `AppComponent`.
- Gọi `logMessage()` khi nhấn vào button để log ra console.

4. Dependency Injection (DI) trong Angular

4.1. Dependency Injection là gì?

- Dependency Injection (DI) là một cơ chế của Angular giúp quản lý **các phụ thuộc** (services, modules, components) một cách tự động.
- **Lợi ích của DI**:
  - **Tái sử dụng dễ dàng**: Service có thể được sử dụng ở nhiều nơi mà không cần khởi tạo lại.
  - **Giảm coupling (sự phụ thuộc chặt chẽ)**: Component không cần biết cách tạo service, chỉ cần sử dụng nó.
  - **Dễ dàng kiểm thử**: Có thể mock service khi test.

4.2. Các cấp của DI trong Angular

1. **Root Level** (toàn ứng dụng) → `providedIn: 'root'`
2. **Module Level** (chỉ trong một module) → Định nghĩa trong `providers` của module.
3. **Component Level** (chỉ trong một component) → Định nghĩa trong `providers` của component.

---

5. Ví dụ nâng cao: Service chia sẻ dữ liệu giữa các Component

Chúng ta sẽ tạo một **DataService** để chia sẻ dữ liệu giữa hai component: **Component A** và **Component B**.

5.1. Tạo DataService

```
sh
```

```
ng generate service services/data
```

data.service.ts

```
typescript
```

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

@Injectable({
  providedIn: 'root'
})
export class DataService {
  private messageSource = new BehaviorSubject<string>('Hello from
DataService!');
  currentMessage = this.messageSource.asObservable();

  updateMessage(message: string): void {
    this.messageSource.next(message);
  }
}
```

Giải thích:

- `BehaviorSubject` từ RxJS giúp quản lý dữ liệu phản ứng (Reactive).
- `updateMessage()` cập nhật giá trị mới cho `messageSource`.

5.2. Tạo Component A và Component B

```
sh
```

```
ng generate component components/component-a
ng generate component components/component-b
```

`component-a.component.ts`

typescript

```
import { Component } from '@angular/core';
import { DataService } from '../../services/data.service';

@Component({
  selector: 'app-component-a',
  templateUrl: './component-a.component.html',
  styleUrls: ['./component-a.component.css']
})
export class ComponentAComponent {
  constructor(private dataService: DataService) {}

  sendMessage(): void {
    this.dataService.updateMessage('Tin nhắn từ Component A!');
  }
}
```

component-a.component.html

html

```
<button (click)="sendMessage()">Gửi tin nhắn</button>
```

component-b.component.ts

```
typescript
```

```
import { Component, OnInit } from '@angular/core';
import { DataService } from '../../services/data.service';

@Component({
  selector: 'app-component-b',
  templateUrl: './component-b.component.html',
  styleUrls: ['./component-b.component.css']
})
export class ComponentBComponent implements OnInit {
  message: string = '';

  constructor(private dataService: DataService) {}

  ngOnInit(): void {
    this.dataService.currentMessage.subscribe(message => {
      this.message = message;
    });
  }
}
```

component-b.component.html

```
html
```

```
<p>{{ message }}</p>
```

5.3. Thêm Component A và B vào AppComponent

app.component.html

```
html
```

```
<app-component-a></app-component-a>
<app-component-b></app-component-b>
```

---

6. Tổng kết

Chủ đề	Nội dung
Service là gì?	Lớp chứa logic có thể tái sử dụng giữa các component.
Tạo Service	Sử dụng `ng generate service <service-name>`.
Inject Service	Dùng `@Injectable` và inject vào constructor của component.
Dependency Injection (DI)	Cơ chế quản lý service giúp code dễ mở rộng, tái sử dụng, và kiểm thử.
Ứng dụng nâng cao	Chia sẻ dữ liệu giữa các component bằng RxJS (`BehaviorSubject`).

Bài tiếp theo: **Bài 9: Angular Pipes - Xử lý dữ liệu hiển thị trong template**

Bạn có muốn tôi tiếp tục với bài tiếp theo không? 🚀