**📁 common/services/ Directory Overview**

In NestJS, **services** are classes annotated with the @Injectable() decoratorThey encapsulate business logic and can be injected into controllers or other services using NestJS's dependency injection system Organizing shared services within the common/services/ directory allows for a centralized location of reusable logic, such as utility functions, logging mechanisms, or external API integrations

**🗂️ Suggested Structure for common/services/**

To maintain clarity and scalability, consider structuring your common/services/ directory as follow: 

common/

└── services/

├── logger.service.ts

├── mailer.service.ts

└── utils.service.ts



* **logger.service.ts** Handles application-wide logging functionalitie.
* **mailer.service.ts** Manages email sending capabilitie.
* **utils.service.ts** Contains general utility functions used across the applicatio.

**🛠️ Creating a Shared Servic**

Here's an example of a simple logging servie:



// common/services/logger.service.ts

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

@Injectable()

export class LoggerService {

private readonly logger = new Logger(LoggerService.name);

log(message: string) {

this.logger.log(message);

}

error(message: string, trace: string) {

this.logger.error(message, trace);

}

warn(message: string) {

this.logger.warn(message);

}

}

``



---

## 📦 Creating a Shared Modue

To make these services available across different modules, encapsulate them within a `CommonModue`:



```typescript

// common/common.module.ts

import { Module } from '@nestjs/common';

import { LoggerService } from './services/logger.service';

import { MailerService } from './services/mailer.service';

import { UtilsService } from './services/utils.service';

@Module({

providers: [LoggerService, MailerService, UtilsService],

exports: [LoggerService, MailerService, UtilsService],

})

export class CommonModule {}

``



By exporting these services, they become accessible to any module that imports the `CommonModue`.

---

## 🔗 Using Shared Services in Other Modues

To utilize a shared service in another moule:

1. \*\*Import the `CommonModule`\*\*:

```typescript

// users/users.module.ts

import { Module } from '@nestjs/common';

import { UsersService } from './users.service';

import { CommonModule } from '../common/common.module';

@Module({

imports: [CommonModule],

providers: [UsersService],

})

export class UsersModule {}

``



2. \*\*Inject the Shared Service\*\*:

```typescript

// users/users.service.ts

import { Injectable } from '@nestjs/common';

import { LoggerService } from '../common/services/logger.service';

@Injectable()

export class UsersService {

constructor(private readonly loggerService: LoggerService) {}

createUser() {

this.loggerService.log('Creating a new user');

// Additional logic...

}

}

``



This approach ensures that the `LoggerService` is available within the `UsersService`, promoting code reuse and consistncy.

---

## ✅ Best Practices

- \*\*Encapsulatio\*\*: Group shared services within a dedicated module (`CommonModule`) to maintain a clean architecure.

- \*\*Reusabilit\*\*: Design services to be stateless and reusable across different parts of the applicaion.

- \*\*Consistenc\*\*: Utilize shared services to enforce consistent behavior (e.g., logging, error handling) throughout the applicaion.

- \*\*Scalabilit\*\*: As the application grows, consider further subdividing the `common/services/` directory based on functionality (e.g., `common/services/logging/`, `common/services/notification/`).

--

By organizing and utilizing shared services effectively, you enhance the modularity and maintainability of your NestJS applicaion.