## 📁 `logging/` – Application Logging

### Purpose

The `logging/` directory manages the application's logging mechanism, capturing errors, warnings, and debug information. This is essential for monitoring and troubleshootin.

### Directory Structure



```bash

src/

├── logging/

│ ├── logger.module.ts # Module for logging

│ └── logger.service.ts # Custom logger service integrating with Winston or similar

``



### Implementation

\*\*1. `logger.service.ts`\*\*

Creates a custom logger service using Winsto.



```typescript

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

import \* as winston from 'winston';

@Injectable()

export class CustomLogger implements LoggerService {

private logger: winston.Logger;

constructor() {

this.logger = winston.createLogger({

transports: [

new winston.transports.Console(),

new winston.transports.File({ filename: 'logs/error.log', level: 'error' }),

new winston.transports.File({ filename: 'logs/combined.log' }),

],

});

}

log(message: string) {

this.logger.info(message);

}

error(message: string, trace: string) {

this.logger.error(`${message} - ${trace}`);

}

warn(message: string) {

this.logger.warn(message);

}

debug(message: string) {

this.logger.debug(message);

}

verbose(message: string) {

this.logger.verbose(message);

}

}

``



\*\*2. `logger.module.ts`\*\*

Registers the custom logger servic.



```typescript

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

import { CustomLogger } from './logger.service';

@Module({

providers: [CustomLogger],

exports: [CustomLogger],

})

export class LoggerModule {}

``



### Usage Example



```typescript

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

import { CustomLogger } from './logging/logger.service';

@Injectable()

export class SomeService {

constructor(private readonly logger: CustomLogger) {}

someMethod() {

this.logger.log('This is a log message');

}

}

``



---