**✅ helper/ Directory: Purpose & Philosophy**

The helper/ (or helpers/) directory contains **reusable utility functions, transformers, formatters, and filters** that don’t belong to any specific domain logic but are **used across multiple modules**.

They help:

* Avoid code duplication ❌🧱
* Improve modularity 🧩
* Make unit testing easier ✅
* Boost developer productivity 🚀

**📁 Suggested Structure for helper/ Directory**

src/

├── helper/

│ ├── date.helper.ts

│ ├── string.helper.ts

│ ├── number.helper.ts

│ ├── validation.helper.ts

│ ├── object.helper.ts

│ ├── file.helper.ts

│ ├── random.helper.ts

│ ├── http-exception.filter.ts # Global error formatting

│ ├── transform-response.interceptor.ts # Wraps responses uniformly

│ └── index.ts # Export all helpers centrally

**🛠️ 1. date.helper.ts**

// Common date formatting and manipulation functions

import \* as moment from 'moment';

/\*\*

\* Returns current date in ISO format (or custom format)

\*/

export const getCurrentDate = (format = 'YYYY-MM-DD'): string => {

return moment().format(format);

};

/\*\*

\* Converts a given date string into readable format

\*/

export const formatDate = (date: string, format = 'MMMM Do YYYY, h:mm:ss a'): string => {

return moment(date).format(format);

};

/\*\*

\* Check if a date is expired (used for token, coupons etc.)

\*/

export const isExpired = (targetDate: string | Date): boolean => {

return moment().isAfter(moment(targetDate));

};

**✂️ 2. string.helper.ts**

// Useful string manipulation helpers

/\*\*

\* Capitalize first letter of a string

\*/

export const capitalize = (value: string): string => {

return value.charAt(0).toUpperCase() + value.slice(1);

};

/\*\*

\* Converts camelCase to snake\_case

\*/

export const camelToSnake = (str: string): string =>

str.replace(/[A-Z]/g, letter => `\_${letter.toLowerCase()}`);

/\*\*

\* Slugify a string (used in URLs)

\*/

export const slugify = (value: string): string => {

return value

.toLowerCase()

.replace(/[^\w\s]/g, '')

.trim()

.replace(/\s+/g, '-');

};

**🔢 3. number.helper.ts**

/\*\*

\* Format large numbers with commas (1,000,000)

\*/

export const formatWithCommas = (value: number): string => {

return value.toLocaleString();

};

/\*\*

\* Round number to given decimal places

\*/

export const roundTo = (num: number, decimal = 2): number => {

return parseFloat(num.toFixed(decimal));

};

**🛡️ 4. validation.helper.ts**

/\*\*

\* Simple email format validator (use Joi or class-validator for stricter checks)

\*/

export const isValidEmail = (email: string): boolean => {

const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

return re.test(email);

};

/\*\*

\* Validate password strength

\*/

export const isStrongPassword = (password: string): boolean => {

return /(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[\W\_]).{8,}/.test(password);

};

**🧱 5. object.helper.ts**

/\*\*

\* Removes undefined/null keys from an object

\*/

export const cleanObject = <T extends object>(obj: T): Partial<T> => {

return Object.entries(obj).reduce((acc, [key, value]) => {

if (value !== undefined && value !== null) acc[key] = value;

return acc;

}, {} as Partial<T>);

};

**📁 6. file.helper.ts**

import \* as fs from 'fs';

import \* as path from 'path';

/\*\*

\* Check if file exists

\*/

export const fileExists = (filePath: string): boolean => {

return fs.existsSync(filePath);

};

/\*\*

\* Generate unique filename

\*/

export const generateFileName = (originalName: string): string => {

const ext = path.extname(originalName);

const base = path.basename(originalName, ext);

return `${base}-${Date.now()}${ext}`;

};

**🎲 7. random.helper.ts**

/\*\*

\* Generate a random numeric OTP

\*/

export const generateOTP = (length = 6): string => {

return Array.from({ length }, () => Math.floor(Math.random() \* 10)).join('');

};

/\*\*

\* Generate random alphanumeric string (for token/slug)

\*/

export const generateSlug = (length = 8): string => {

const chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';

return Array.from({ length }, () => chars[Math.floor(Math.random() \* chars.length)]).join('');

};

**❌ 8. http-exception.filter.ts (Global Error Filter)**

import {

ArgumentsHost,

Catch,

ExceptionFilter,

HttpException,

HttpStatus,

} from '@nestjs/common';

import { Request, Response } from 'express';

@Catch()

export class AllExceptionsFilter implements ExceptionFilter {

catch(exception: unknown, host: ArgumentsHost) {

const ctx = host.switchToHttp();

const response = ctx.getResponse<Response>();

const request = ctx.getRequest<Request>();

const status =

exception instanceof HttpException

? exception.getStatus()

: HttpStatus.INTERNAL\_SERVER\_ERROR;

const message =

exception instanceof HttpException

? exception.getResponse()

: 'Internal Server Error';

response.status(status).json({

statusCode: status,

timestamp: new Date().toISOString(),

path: request.url,

message,

});

}

}

**🎁 9. transform-response.interceptor.ts (Optional)**

import {

Injectable,

NestInterceptor,

ExecutionContext,

CallHandler,

} from '@nestjs/common';

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

/\*\*

\* Automatically wraps all successful responses in a consistent structure.

\*/

@Injectable()

export class TransformResponseInterceptor implements NestInterceptor {

intercept(context: ExecutionContext, next: CallHandler): Observable<any> {

return next.handle().pipe(

map(data => ({

success: true,

data,

timestamp: new Date().toISOString(),

})),

);

}

}

**📦 10. index.ts (Optional)**

// Central export for all helpers if needed

export \* from './date.helper';

export \* from './string.helper';

export \* from './number.helper';

export \* from './validation.helper';

export \* from './object.helper';

export \* from './file.helper';

export \* from './random.helper';

**✅ Benefits of This Helper Layer**

| **Category** | **Purpose** | **Example Use** |
| --- | --- | --- |
| date.helper.ts | Consistent date formatting | Logs, Reports |
| string.helper.ts | Slug generation, string sanitization | URL, Tags, Labels |
| number.helper.ts | Comma format, rounding | Financial Display |
| file.helper.ts | File name management, validation | Upload Service |
| http-exception.filter.ts | Global error formatting | API Standardization |
| random.helper.ts | OTP, ID generation | Auth, Ref Links |