# 🧩 @nready/nestjs-shared

**Shared utility library for NestJS and Node.js microservice projects.**
Open-source and commercial-friendly under the **MIT License**.


---

# @nready/nestjs-shared

```
_____   _____________________________________  __   ____________________
___  | / /__  __ \__  ____/__    |__  __ \ \/ /   ___  /___    |__  __ )
__   |/ /__  /_/ /_  __/  __  /| |_  / / /_  /    __  / __  /| |_  __  |
_  /|  / _  _, _/_  /___  _  ___ |  /_/ /_  /     _  /___  ___ |  /_/ /
/_/ |_/  /_/ |_| /_____/  /_/  |_/_____/ /_/      /_____/_/  |_/_____/
```

## 📘 Overview

`@nready/nestjs-shared` provides **lightweight, type-safe, and framework-agnostic utilities** for NestJS-based microservice architectures — built to simplify development, testing, and scaling.

> Designed for internal projects, but open for public and commercial use.

---

The lib `@nready/nestjs-shared` is used for Nest.js v10.x project with pre-built some utilities function:

## Usage

First install the library with command:

```
npm i @nready/nestjs-shared
```

## ✨ Features

- 🧩 Shared **decorators**, **mappers**, and **interceptors**
- 🔐 JWT, session, and permission helpers
- 🧠 Dynamic DTO mapping engine
- ⚙️ Utility functions for cross-service communication
- 🪶 Zero dependencies, fully TypeScript-based

### Message Queue

The pre-built Message Queue using RabbitMq, supported re-connect if RabbitMq is down, supported queue survives RabbitMQ restart, Message will persist until it is consumed

- `emit`: Use AMQP library to send persistent messages
- `overwriteQueue`: Overwrite an existed key with another value.

Install AMQP library in project:

```
npm i amqp-connection-manager
npm i amqplib
```

Config the `.env`:

```
RABBITMQ_URL=amqp://localhost:5672
RABBITMQ_QUEUE_NAME=USER
RABBITMQ_TTL=3600000
RABBITMQ_ACK=false
```

Register the `@nready/nestjs-shared` in `app.module.ts`.

```
import { MessagingModule } from '@nready/nestjs-shared';

@Module({
  controllers: [AppController],
  providers: [],
  imports: [
    MessagingModule,
    ...
```

Import in Service:

```
import { MessagingService } from '@nready/nestjs-shared';
...
@Injectable()
export class MyService {
  constructor(
    private rabbitClient: MessagingService,
  ) {}
  ...
  public async myFunction(): Promise {
    await this.rabbitClient.emit(key, value);
  }
```

## Redis

The pre-built Cache using Redis, supported re-connect if the Redis server is down and more.

- `setKey(key: string, value: string, ttlMinutes?: number)`
- `getKey(key: string)`
- `deleteKey(key: string)`
- `increaseValue(key: string)`
- `reset()`

Install ioredis lib in project:

```
npm i ioredis
```

Config `.env`:

```
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PREFIX=NRD
REDIS_TTL=5
```

Register the `@nready/nestjs-shared` in `app.module.ts`.

```
import { RedisModule } from '@nready/nestjs-shared';

@Module({
  controllers: [AppController],
  providers: [],
  imports: [
```

```
    RedisModule,
    ...
```

Import in Service:

```
import { RedisService } from '@nready/nestjs-shared';
...
@Injectable()
export class MyService {
  constructor(
    private readonly redisService: RedisService,
  ) {}
  ...
  public async myFunction(): Promise {
    await this.redisService.setKey(key, value);
  }
```

## Abstract Model

With pre-built properties that inheritance from Abstract Class, we will have common field and no need to duplicate these properties.

List of these properties: `createDate`, `effectDate`, `inactiveDate`, `dateLastMaint`, `version`, `editedBy`, `approvedBy`, `note`.

In `*.entity.ts`, inherit from abstract Model:

```
export class MyEntity extends AbstractEntity {
  // rest of properties
}
```

## Search and paging

Dto:

```
// Search
import { AbstractSearchDto } from '@nready/nestjs-shared';
export class MyEntitySearch extends AbstractSearchDto<MyEntity> {}

//
@Exclude()
export class MyEntityResponseDto extends MyEntity {
  @Expose()
  id: string;
  ...
```

In **Controller**:

```
@Get()
async get(@Query(new QueryTransformPipe()) searchModel: MyEntitySearch)
{
  const res = await this.service.query(searchModel);
  return res;
}
```

In **Service**:

There are 2 ways:

- We can use `AbstractSearchService` all call the method `paginate(model)`:

```
import { AbstractSearchService, SearchResultDto } from '@nready/nestjs-
shared';
import { plainToInstance } from 'class-transformer';

@Injectable()
export class MyService extends AbstractSearchService<MyEntity,
MyEntitySearch> {
  constructor(
    @InjectRepository(MyEntitySearch) private readonly repository:
Repository<MyEntitySearch>
  ) {
    super(repository);
  }

  public async query(model: MyEntitySearch): Promise<any> {
    const res = await this.paginate(model);
    const data = plainToInstance(MyEntityResponseDto, res.data, {
excludeExtraneousValues: true });
    return new SearchResultDto<MyEntityResponseDto>(res.pageMeta, data);
  }
  ...
```

- or call directly the method `paginateRepository(repo: Repository<T>, model: S, defaultOrder?: FindOptionsOrder<T>)`:

```
import { paginateRepository, SearchResultDto } from '@nready/nestjs-
shared';
import { plainToInstance } from 'class-transformer';

@Injectable()
export class MyService {
  constructor(
    @InjectRepository(MyEntitySearch) private readonly repository:
```

```
Repository<MyEntitySearch>
  ) {}

  public async query(model: MyEntitySearch): Promise<any> {
    const res = await paginateRepository<MyEntity, MyEntitySearch>(
      this.repository as any,
      model,
    );
    const data = plainToInstance(MyEntityResponseDto, res.data, {
excludeExtraneousValues: true });
    return new SearchResultDto<MyEntityResponseDto>(res.pageMeta, data);
  }
  ...
```

Curl with paging and search by field name:

```
curl --location '/?page=0&take=5&someFieldName=blahblah' \
--header 'Content-Type: application/json'
```

## Middlewares

**ApplicationMiddleware**

With this Middleware, it is restricted to access until get authenticated.

**TransformInterceptor**

All the response will have **Http Status Code 200** and Wrapped.

In `app.module.ts`:

```
import { ApplicationMiddleware } from '@nready/nestjs-shared';
...
export class AppModule implements NestModule {
  configure(consumer: MiddlewareConsumer) {
    consumer
      .apply(ApplicationMiddleware)
      .forRoutes({ path: '(*)', method: RequestMethod.ALL });
  }
}
```

An example:

```
{
    "timestamp": "2025-06-03T09:41:39.699Z",
    "statusCode": 400,
```

```
    "message": [
        "email must be shorter than or equal to 50 characters",
        "Email has invalid format",
        "Email is mandatory"
    ],
    "error": "Bad Request"
}
```

### Utils Service

- UtilsService
  - .generateHash(password: string)
  - .validateHash(password: string, hash: string)
  - .generateRandomInteger(min: number, max: number)
  - .generateRandomString(length: number)
  - .getAge(d1: Date, d2?: Date)
  - .capitalizeName(name: string)
  - .encodeString(text: string)
  - .mergeObject(A: any, B: any)
  - .cleanNullObject(obj: any)
  - .getLocaleDate(isoString: string, timeZone?: string)
  - .isNullOrUndefined(value: any)
  - .isFutureDate(value: any)
  - .isActiveByDate(effectDate: Date, inactiveDate: Date)
  - .base64Encode(text: string)
  - .base64Decode(text: string)
  - .randomString()
  - .transformEndOfDate(date: Date | string)

(tobe deprecated)

- extractKey(path: string)
- transformEndOfDate(date: Date | string)
- randomString(length = 60)
- isDate(value: any)
- getTodayFormatYYYYDDMM()
- hasCommonItemInArrays(arr1: [], arr2: [])
- convertToAscii(str: string)
- cleanNullObject(obj: any)
- isNullable(value: any)
- isFutureDate(value: any)
- isActiveByDate(effectDate: Date, inactiveDate: Date)

## Development

Check Published Package:

```
npm pack
```

use in project with zip:

```
npm install ../@nready/nestjs-shared-0.0.1.tgz
```

or using locally:

```
# package.json
"@nready/nestjs-shared": "file:../libs/nestjs-shared",
```

### Use in project

```
npm i @nready/nestjs-shared
```

### Publish

```
npm publish --access public
```

## License


Licensed under the MIT License
.
Copyright © 2025 Le Quoc Nam

This library is free for both personal and commercial use.
No warranty provided; use at your own discretion.

💡 Built and maintained by Le Quoc Nam, <leqnam@live.com (nam@nready.net)>
https://nready.net/

Open. Simple. Reusable.