BadRequestException

August 18, 2026

BadRequestException

Kind: Class

Source: `packages/common/exceptions/bad-request.exception.ts`

Part of: Common

Defines an HTTP exception for *Bad Request* type errors.

`BadRequestException` represents an HTTP 400 error caused by invalid client input, malformed requests, or failed validation. It extends the framework's HTTP exception handling flow, allowing controllers and services to return consistent error responses that can be processed by global exception filters.

Extends: `HttpException`

Diagram

```
graph LR
  Client[Client Request] --> Controller[Controller or Service]
  Controller --> Validation{Input valid?}
  Validation -->|Yes| Handler[Continue request handling]
  Validation -->|No| Exception[BadRequestException]
  Exception --> Filter[HTTP Exception Filter]
  Filter --> Response[HTTP 400 Bad Request Response]
```

Usage

```
import { BadRequestException } from '@nestjs/common';

export class UsersService {
  async createUser(email: string) {
    const existingUser = await this.findByEmail(email);

    if (existingUser) {
      throw new BadRequestException(
```

```
        'A user with this email address already exists',
      );
    }

    return this.saveUser({ email });
  }

  private async findByEmail(email: string) {
    // Look up an existing user.
  }

  private async saveUser(user: { email: string }) {
    // Persist and return the new user.
  }
}
```

AI Coding Instructions

- Throw `BadRequestException` only for client-correctable request errors, such as invalid input, missing required data, or invalid request state.
- Provide clear, actionable error messages; avoid exposing internal implementation details, database errors, or sensitive data.
- Prefer validation pipes and DTO validation decorators for standard input validation; use this exception for custom business-rule validation.
- Do not use this exception for authentication, authorization, missing resources, or server failures; use the corresponding HTTP exception type instead.
- Ensure custom exception payloads remain compatible with the application's global exception filters and API error-response conventions.

Used by

10 references from 10 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (10)

- `RoutesResolver` — `packages/core/router/routes-resolver.ts` :35
- `ErrorsController` — `integration/hello-world/src/errors/errors.controller.ts` :3
- `ParseIntPipe` — `integration/inspector/src/common/pipes/parse-int.pipe.ts` :8
- `ParseIntPipe` — `sample/01-cats-app/src/common/pipes/parse-int.pipe.ts` :8
- `ValidationPipe` — `sample/01-cats-app/src/common/pipes/validation.pipe.ts` :11
- `ParseIntPipe` — `sample/10-fastify/src/common/pipes/parse-int.pipe.ts` :4

- `ValidationPipe` — `sample/10-fastify/src/common/pipes/validation.pipe.ts`:11
- `ParseIntPipe` — `sample/36-hmr-esm/src/common/pipes/parse-int.pipe.ts`:8

…and 2 more.