ExceptionsHandler

August 18, 2026

ExceptionsHandler

Kind: Class

Source: `packages/core/exceptions/exceptions-handler.ts`

Part of: Core

`ExceptionsHandler` coordinates exception processing for a request or execution context. It first attempts to match and invoke registered custom exception filters, then falls back to the base exception filter when no custom filter handles the error.

Extends: `BaseExceptionFilter`

Methods

Method	Signature	Returns
`next`	`next(exception: Error	HttpException, ctx: ArgumentsHost)`
`setCustomFilters`	`setCustomFilters(filters: ExceptionFilterMetadata[])`	`void`
`invokeCustomFilters`	`invokeCustomFilters(exception: T, ctx: ArgumentsHost)`	`boolean`

Where it refuses work

- `ExceptionsHandler` stops the work with `InvalidExceptionFilterException` when `!Array.isArray(filters)`.
- `ExceptionsHandler` stops the work with an early return when `this.invokeCustomFilters(exception, ctx)`.
- `ExceptionsHandler` stops the work with an early return when `isEmpty(this.filters)`.

Diagram

```
graph LR
  A[Exception thrown] --> B[ExceptionsHandler.next]
  B --> C{Custom filter matches?}
  C -->|Yes| D[invokeCustomFilters]
  D --> E[Custom filter response]
  C -->|No| F[BaseExceptionFilter.catch]
  F --> G[Default error response]
```

Usage

```
import { ArgumentsHost, Catch, ExceptionFilter } from '@nestjs/common';
import { HttpAdapterHost } from '@nestjs/core';
import { ExceptionsHandler } from '@nestjs/core/exceptions/exceptions-handler';

class DomainError extends Error {}

@Catch(DomainError)
class DomainErrorFilter implements ExceptionFilter {
  catch(exception: DomainError, host: ArgumentsHost) {
    const response = host.switchToHttp().getResponse();

    response.status(400).json({
      statusCode: 400,
      message: exception.message,
    });
  }
}

const { httpAdapter } = app.get(HttpAdapterHost);
const exceptionsHandler = new ExceptionsHandler(httpAdapter);

const filter = new DomainErrorFilter();

exceptionsHandler.setCustomFilters([
  {
    exceptionMetatypes: [DomainError],
    func: filter.catch.bind(filter),
  },
]);

// Typically called internally by Nest when an exception occurs.
exceptionsHandler.next(new DomainError('Invalid domain input'), argumentsHost);
```

AI Coding Instructions

- Register custom filters through `setCustomFilters()` using `exceptionMetatypes` and a correctly bound `func` callback.

- Call `next()` to preserve the normal handling flow; it invokes matching custom filters before using the default exception response behavior.
- Ensure custom filter callbacks preserve their `this` context, such as with `filter.catch.bind(filter)`.
- Return control from a custom filter only after writing an appropriate response for the active transport context.
- Treat `ExceptionsHandler` as framework infrastructure; prefer Nest's `@UseFilters()` and `@Catch()` APIs for most application-level exception handling.

How it works

- `ExceptionsHandler` is an exported HTTP exception handler that extends `BaseExceptionFilter`. It keeps a private array of custom `ExceptionFilterMetadata`, initially empty. [packages/core/exceptions/exceptions-handler.ts:9-10]
- `next(exception, ctx)` first attempts custom-filter handling. If a matching custom filter is found, it returns without invoking the inherited fallback; otherwise, it calls `BaseExceptionFilter.catch(exception, ctx)`. [packages/core/exceptions/exceptions-handler.ts:12-17]
- The fallback treats `HttpException` instances differently from other values: it obtains the exception response, sends it through the HTTP adapter when headers have not already been sent, or ends the response otherwise. [packages/core/exceptions/base-exception-filter.ts:26-47] For non-`HttpException` values, it sends either an error object's `statusCode` and `message`, or a 500 response with the unknown-exception message; it logs non-`IntrinsicException` values. [packages/core/exceptions/base-exception-filter.ts:50-75]
- `setCustomFilters(filters)` requires `filters` to be an array. A non-array value throws `InvalidExceptionFilterException`; an array replaces the handler's current filter array. [packages/core/exceptions/exceptions-handler.ts:19-24] That exception extends `RuntimeException` and has the message `Invalid exception filters (@UseFilters()).`. [packages/core/errors/exceptions/invalid-exception-filter.exception.ts:4-7] [packages/core/errors/messages.ts:262]
- `invokeCustomFilters(exception, ctx)` returns `false` when its filter array has no elements. [packages/core/exceptions/exceptions-handler.ts:26-32] Otherwise, it selects the first metadata entry whose `exceptionMetatypes` array is empty or contains a type for which `exception instanceof ExceptionMetaType` is true. [packages/common/utils/select-exception-filter-metadata.util.ts:3-13] When

selected, it calls that entry's `func(exception, ctx)` and returns `true`; when none match, it returns `false`. [packages/core/exceptions/exceptions-handler.ts:34-36]

- Each filter metadata entry contains a `func` callback typed as an exception filter's `catch` method and an `exceptionMetatypes` array. [packages/common/interfaces/exceptions/exception-filter-metadata.interface.ts:4-7] Filter contexts create these entries by binding an instantiated filter's `catch` method and reading its reflected caught-exception metadata. [packages/core/exceptions/base-exception-filter-context.ts:26-36] [packages/core/exceptions/base-exception-filter-context.ts:73-77]
- In HTTP route setup, `RouterExceptionFilters.create` constructs `ExceptionsHandler` with the HTTP server adapter. If filters exist, it reverses their order before passing them to `setCustomFilters`. [packages/core/router/router-exception-filters.ts:23-44] `RouterProxy` calls `next` after a route callback or exception-layer callback throws, passing an `ExecutionContextHost` containing request, response, and `next`. [packages/core/router/router-proxy.ts:20-26] [packages/core/router/router-proxy.ts:45-51]

Relationships

- IMPORTS → `HttpException`
- IMPORTS → `ExceptionFilterMetadata`
- IMPORTS → `ArgumentsHost`
- IMPORTS → `selectExceptionFilterMetadata`
- IMPORTS → `isEmpty`