ExternalExceptionsHandler

August 18, 2026

ExternalExceptionsHandler

Kind: Class

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

Part of: Core

`ExternalExceptionsHandler` routes exceptions raised outside the standard request pipeline, such as errors from external contexts or adapters. It first attempts to handle an exception with registered custom filters, then falls back to Nest's default external exception handling behavior when no matching filter is available.

Extends: `ExternalExceptionFilter`

Methods

Method	Signature	Returns
`next`	`next(exception: Error, host: ArgumentsHost)`	`Promise<any>`
`setCustomFilters`	`setCustomFilters(filters: ExceptionFilterMetadata[])`	`void`
`invokeCustomFilters`	`invokeCustomFilters(exception: T, host: ArgumentsHost)`	`Promise

Where it refuses work

- `ExternalExceptionsHandler` stops the work with `InvalidExceptionFilterException` when `!Array.isArray(filters)`.
- `ExternalExceptionsHandler` stops the work with an early return when `result`.
- `ExternalExceptionsHandler` stops the work with an early return when `isEmpty(this.filters)`.

Diagram

```
graph LR
  A[Exception raised] --> B[ExternalExceptionsHandler.next]
  B --> C[invokeCustomFilters]
  C --> D{Matching custom filter?}
  D -->|Yes| E[Execute filter function]
  D -->|No| F[Delegate to default handler]
  F --> G[ExternalExceptionFilter.catch]
```

Usage

```
import { ExternalExceptionsHandler } from '@nestjs/core/exceptions/external-exceptions-handler'

const exceptionsHandler = new ExternalExceptionsHandler();

exceptionsHandler.setCustomFilters([
  {
    exceptionMetatypes: [Error],
    func: async (exception, host) => {
      console.error('External operation failed:', exception.message);

      // Handle the error for the current execution context.
      const response = host.switchToHttp().getResponse();
      response.status(500).json({
        message: 'An external operation failed',
      });
    },
  },
]);

// Typically invoked internally by Nest when an external-context callback fails.
await exceptionsHandler.next(new Error('Service unavailable'), host);
```

AI Coding Instructions

- Register custom filters through `setCustomFilters()` before calling `next()`; an empty filter list preserves default behavior.
- Ensure each custom filter declares the exception types it handles and provides a `func(exception, host)` callback.
- Let `next()` remain the primary entry point so unmatched exceptions correctly fall back to `ExternalExceptionFilter`.
- Return or await asynchronous filter work from `func`; `invokeCustomFilters()` supports promise-based handlers.

- Avoid bypassing this handler in external-context integrations, since doing so skips custom filter selection and fallback handling.

How it works

`ExternalExceptionsHandler`

`ExternalExceptionsHandler` is an exception handler class that extends `ExternalExceptionFilter` and keeps an initially empty private list of `ExceptionFilterMetadata` entries. packages/core/exceptions/external-exceptions-handler.ts:8-9

Relationships

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