ExceptionsZone

August 18, 2026

ExceptionsZone

Kind: Class

Source: `packages/core/errors/exceptions-zone.ts`

Part of: Core

`ExceptionsZone` executes synchronous or asynchronous bootstrap work within a centralized exception boundary. It delegates caught errors to Nest's exception handling and logging infrastructure, then runs a teardown callback—by default terminating the process.

Methods

Method	Signature	Returns
`run`	`run(callback: () => void, teardown: (err: any) => void, autoFlushLogs: boolean)`	`void`
`asyncRun`	`asyncRun(callback: () => Promise<void>, teardown: (err: any) => void, autoFlushLogs: boolean)`	`void`

When something fails

- `ExceptionsZone` handles failure in 2 places: it logs it and continues in all 2.

Diagram

```
graph LR
  A[Application bootstrap callback] --> B{ExceptionsZone.run / asyncRun}
  B -->|Success| C[Continue execution]
  B -->|Throws or rejects| D[ExceptionHandler]
  D --> E[Flush logs]
  E --> F[Teardown callback]
  F --> G[Exit process by default]
```

Usage

```
import { ExceptionsZone } from '@nestjs/core/errors/exceptions-zone';

async function bootstrap() {
  // Create and start the application here.
  console.log('Application started');
}

ExceptionsZone.asyncRun(
  bootstrap,
  error => {
    console.error('Bootstrap failed:', error);
    process.exitCode = 1;
  },
);
```

AI Coding Instructions

- Use `run()` for synchronous callbacks and `asyncRun()` for callbacks that return a `Promise`.
- Provide a custom teardown callback when callers need cleanup, test-friendly behavior, or custom process-exit handling.
- Do not catch and silently ignore errors inside the callback; allow unrecoverable bootstrap errors to reach `ExceptionsZone`.
- Preserve the exception handler and log-flushing flow when modifying this class, as it ensures startup failures are reported consistently.
- Treat this as core bootstrap infrastructure; application-level request exceptions should use Nest exception filters instead.

Relationships

- IMPORTS → `Logger`