ExecutionContext

August 17, 2026

ExecutionContext

Kind: Interface

Source: `packages/common/interfaces/features/execution-context.interface.ts`

Part of: Common

Interface describing details about the current request pipeline.

`ExecutionContext` describes the state and metadata available while a request moves through the application's execution pipeline. It provides a shared context for pipeline features, allowing them to read request-scoped information and coordinate processing without relying on global state.

Diagram

```
graph LR
  Request[Incoming Request] --> Pipeline[Request Pipeline]
  Pipeline --> Context[ExecutionContext]
  Context --> FeatureA[Feature / Middleware A]
  Context --> FeatureB[Feature / Middleware B]
  FeatureA --> Result[Response or Execution Result]
  FeatureB --> Result
```

Usage

```
import type { ExecutionContext } from '@common/interfaces/features/execution-context.interface'

function processFeature(context: ExecutionContext): void {
  // Read request-scoped details from the context and perform
  // feature-specific processing.
  console.log('Processing request with execution context:', context);
}

function runPipeline(context: ExecutionContext): void {
  processFeature(context);
```

```
  // Pass the same context to subsequent pipeline features so they
  // operate on the same request-scoped execution state.
  // validateFeature(context);
  // authorizationFeature(context);
}
```

AI Coding Instructions

- Treat `ExecutionContext` as request-scoped data; do not reuse one instance across unrelated requests.
- Pass the existing context through pipeline stages instead of creating disconnected context objects.
- Keep feature-specific logic in pipeline handlers rather than adding unrelated behavior to the interface.
- When extending the context, ensure new fields are available to every pipeline integration that depends on them.

Used by

48 references from 48 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (48)

- `AuthGuard` — `integration/graphql-code-first/src/common/guards/auth.guard.ts`:9
- `DataInterceptor` — `integration/graphql-code-first/src/common/interceptors/data.interceptor.ts`:10
- `CatsGuard` — `integration/graphql-schema-first/src/cats/cats.guard.ts`:4
- `Guard` — `integration/inspector/src/circular-hello/guards/request-scoped.guard.ts`:9
- `Interceptor` — `integration/inspector/src/circular-hello/interceptors/logging.interceptor.ts`:10
- `RolesGuard` — `integration/inspector/src/common/guards/roles.guard.ts`:4
- `TimeoutInterceptor` — `integration/inspector/src/common/interceptors/timeout.interceptor.ts`:10
- `LoggingInterceptor` — `integration/inspector/src/core/interceptors/logging.interceptor.ts`:10

…and 40 more.