# ScopeOptions

August 18, 2026

## ScopeOptions

**Kind:** Interface

**Source:** `packages/common/interfaces/scope-options.interface.ts`

**Part of:** Common

`ScopeOptions` configures how a scoped dependency or context should behave within the application. It pairs a `Scope` value with a `durable` flag that determines whether the scoped instance or state should be retained beyond its normal lifecycle.

### Properties

| Property | Type |
| --- | --- |
| `scope` | `Scope` |
| `durable` | `boolean` |

### Diagram

```
graph LR
  A[ScopeOptions] --> B[scope: Scope]
  A --> C[durable: boolean]
  B --> D[Defines lifecycle boundary]
  C --> E[Controls persistence across lifecycle events]
```

### Usage

```
import type { ScopeOptions } from '@common/interfaces/scope-options.interface';
import { Scope } from '@common/enums/scope.enum';

const requestScopeOptions: ScopeOptions = {
  scope: Scope.REQUEST,
  durable: false,
```

```
};

const durableScopeOptions: ScopeOptions = {
  scope: Scope.DEFAULT,
  durable: true,
};

function configureScope(options: ScopeOptions): void {
  if (options.durable) {
    console.log(`Keeping ${options.scope} scope state available.`);
    return;
  }

  console.log(`Cleaning up ${options.scope} scope state after use.`);
}

configureScope(requestScopeOptions);
configureScope(durableScopeOptions);
```

## AI Coding Instructions

- Always provide both `scope` and `durable`; neither field should be omitted when constructing `ScopeOptions`.
- Use the project's existing `Scope` enum or type rather than hard-coded string values.
- Set `durable` to `true` only when state must survive the normal scope lifecycle, as it may affect cleanup and memory usage.
- Keep lifecycle behavior consistent with the consuming dependency injection, request handling, or context-management code.

## How it works

`ScopeOptions` is a public TypeScript interface for specifying the injection lifetime of a provider or controller. It contains two optional properties: `scope` and `durable`. [packages/common/interfaces/scope-options.interface.ts:21-37]

- `scope?: Scope` selects a lifetime from the accompanying `Scope` enum. The enum defines:
  - `DEFAULT`: a provider may be shared by multiple classes; its lifetime is tied to the application lifecycle, and the source states that all providers are instantiated after application bootstrap. [packages/common/interfaces/scope-options.interface.ts:4-10]

    - `TRANSIENT`: a new private provider instance is instantiated for every use. [packages/common/interfaces/scope-options.interface.ts:11-14]
    - `REQUEST`: a new instance is instantiated for each request-processing pipeline. [packages/common/interfaces/scope-options.interface.ts:15-18]
- `durable?: boolean` flags a provider as durable. The interface comments state that this is for use with a custom context-id factory strategy to construct lazy DI subtrees, and only with `scope: Scope.REQUEST`. [packages/common/interfaces/scope-options.interface.ts:31-37]

`InjectableOptions` is an alias for `ScopeOptions`, so `@Injectable()` accepts these options. The decorator writes the supplied options as reflection metadata under `SCOPE_OPTIONS_METADATA` on the decorated class. [packages/common/decorators/core/injectable.decorator.ts:13,43-47]

`ControllerOptions` extends `ScopeOptions`, and `@Controller()` writes an object containing its `scope` and `durable` values to the same metadata key. [packages/common/decorators/core/controller.decorator.ts:16,151-176]

At runtime, core reads that metadata to obtain a class's scope and durable flag. [packages/core/helpers/get-class-scope.ts:5-8] [packages/core/helpers/is-durable.ts:4-7] When registering a class provider, the module stores both values in its `InstanceWrapper`. [packages/core/injector/module.ts:266-278] For a request-scoped wrapper, an omitted `durable` value is treated as `false`; a supplied value becomes the dependency tree's durable state. [packages/core/injector/instance-wrapper.ts:245-255]

The interface and the decorator implementations shown do not validate option values or throw errors. [packages/common/interfaces/scope-options.interface.ts:26-37] [packages/common/decorators/core/injectable.decorator.ts:43-47] [packages/common/decorators/core/controller.decorator.ts:151-177]