# Next

August 18, 2026

## Next

**Kind:** Constant

**Source:** `packages/common/decorators/http/route-params.decorator.ts`

**Part of:** Common

Route handler parameter decorator. Extracts reference to the `Next` function from the underlying platform and populates the decorated parameter with the value of `Next`.

`Next` is a route handler parameter decorator that injects the underlying platform's `next` callback into a controller method parameter. It is primarily used in Express-based applications to delegate control to the next middleware or error handler in the request pipeline.

### Definition

```
() => ParameterDecorator
```

### Value

```
createRouteParamDecorator(
  RouteParamtypes.NEXT,
)
```

### Diagram

```
graph LR
  A[Incoming HTTP Request] --> B[Middleware Pipeline]
  B --> C[Controller Route Handler]
  C --> D[@Next() Decorated Parameter]
```

```
  D --> E[next Function]
  E --> F[Next Middleware or Error Handler]
```

## Usage

```
import { Controller, Get, Next } from '@nestjs/common';
import type { NextFunction } from 'express';

@Controller('users')
export class UsersController {
  @Get()
  findAll(@Next() next: NextFunction) {
    try {
      // Handle the request or delegate to later middleware.
      next();
    } catch (error) {
      next(error);
    }
  }
}
```

## AI Coding Instructions

- Use `@Next()` only on route-handler parameters where direct access to the platform middleware flow is required.
- Type the injected value as `NextFunction` when using the Express platform.
- Call `next(error)` to delegate unexpected errors to registered error-handling middleware.
- Prefer NestJS guards, interceptors, pipes, and exception filters for framework-native request handling when possible.
- Ensure the selected HTTP adapter supports a `next` callback before relying on this decorator.