HttpCode

August 17, 2026

HttpCode

Kind: Function

Source: `packages/common/decorators/http/http-code.decorator.ts`

Part of: Common

Request method Decorator. Defines the HTTP response status code. Overrides default status code for the decorated request method.

`HttpCode()` is a method decorator that explicitly sets the HTTP status code returned by a request handler. It overrides the framework's default response status for the decorated controller method, such as returning `204 No Content` after a successful update.

Signature

```
function HttpCode(statusCode: number): MethodDecorator
```

Parameters

Name	Type
`statusCode`	`number`

Returns: `MethodDecorator`

Diagram

```
graph LR
  A[Incoming HTTP Request] --> B[Controller Route Handler]
  B --> C[@HttpCode(status)]
```

```
  C --> D[Response Status Metadata]
  D --> E[HTTP Response with Configured Status Code]
```

Usage

```
import { Controller, Post, HttpCode, HttpStatus } from '@nestjs/common';

@Controller('sessions')
export class SessionsController {
  @Post('logout')
  @HttpCode(HttpStatus.NO_CONTENT)
  logout(): void {
    // End the current session.
  }
}
```

AI Coding Instructions

- Apply `@HttpCode()` directly to controller request-handler methods, alongside route decorators such as `@Get()`, `@Post()`, or `@Delete()`.
- Use `HttpStatus` constants instead of hard-coded numeric status codes when possible for readability.
- Choose a status code that matches the endpoint behavior; for example, use `204` when no response body is returned.
- Remember that `@HttpCode()` overrides the default status associated with the HTTP request method.

Used by

12 references from 12 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (12)

- `AppController` — `integration/microservices/src/app.controller.ts`:20
- `GrpcController` — `integration/microservices/src/grpc/grpc.controller.ts`:21
- `AdvancedGrpcController` — `integration/microservices/src/grpc-advanced/advanced.grpc.controller.ts`:14
- `KafkaController` — `integration/microservices/src/kafka/kafka.controller.ts`:15

- `KafkaConcurrentController` — `integration/microservices/src/kafka-concurrent/kafka-concurrent.controller.ts`:24
- `KafkaConcurrentMessagesController` — `integration/microservices/src/kafka-concurrent/kafka-concurrent.messages.controller.ts`:6
- `MqttController` — `integration/microservices/src/mqtt/mqtt.controller.ts`:16
- `NatsController` — `integration/microservices/src/nats/nats.controller.ts`:19

…and 4 more.