ClassSerializerInterceptor

August 18, 2026

ClassSerializerInterceptor

Kind: Service

Source: `packages/common/serializer/class-serializer.interceptor.ts`

Part of: Common

`ClassSerializerInterceptor` is a NestJS interceptor that transforms controller responses into plain JavaScript objects using `class-transformer`. It applies serialization rules such as `@Exclude()`, `@Expose()`, groups, versioning, and custom transformation options before the response is sent to the client. It can read serializer options from global configuration or route-level metadata.

Methods

Method	Signature	Returns	Description
`intercept`	`intercept(context: ExecutionContext, next: CallHandler)`	`Observable<any>`	
`serialize`	`serialize(response: PlainLiteralObject`	Array, options: ClassSerializerContextOptions)`	`PlainLiteralObject
`transformToPlain`	`transformToPlain(plainOrClass: any, options: ClassSerializerContextOptions)`	`PlainLiteralObject`	
`getContextOptions`	`getContextOptions(context: ExecutionContext)`	`ClassSerializerContextOptions	undefined`	

Dependencies

- `ClassSerializerInterceptorOptions` *(optional)*

Where it refuses work

- `ClassSerializerInterceptor` stops the work with an early return when `!isObject(response) || response instanceof StreamableFile`.

- `ClassSerializerInterceptor` stops the work with an early return when `!plainOrClass`.
- `ClassSerializerInterceptor` stops the work with an early return when `!options.type`.
- `ClassSerializerInterceptor` stops the work with an early return when `plainOrClass instanceof options.type`.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Interceptor as ClassSerializerInterceptor
  participant Reflector
  participant Transformer as class-transformer

  Client->>Interceptor: HTTP request
  Interceptor->>Reflector: Read route/class serializer options
  Interceptor->>Controller: next.handle()
  Controller-->>Interceptor: Response entity or entity array
  Interceptor->>Transformer: instanceToPlain() / classToPlain()
  Transformer-->>Interceptor: Serialized plain object
  Interceptor-->>Client: HTTP response
```

Usage

```
import {
  ClassSerializerInterceptor,
  Controller,
  Get,
  UseInterceptors,
} from '@nestjs/common';
import { Exclude, Expose } from 'class-transformer';

class UserResponseDto {
  @Expose()
  id: string;

  @Expose()
  email: string;

  @Exclude()
  password: string;
}

@Controller('users')
@UseInterceptors(ClassSerializerInterceptor)
export class UsersController {
  @Get('me')
  getCurrentUser(): UserResponseDto {
    return {
      id: 'user_123',
```

```
      email: 'developer@example.com',
      password: 'internal-secret',
    };
  }
}
```

AI Coding Instructions

- Return class instances or DTO instances with `class-transformer` decorators; plain objects may not apply `@Exclude()` and `@Expose()` metadata as expected.
- Apply `ClassSerializerInterceptor` globally, at the controller level, or per route depending on the desired serialization scope.
- Use `@SerializeOptions()` when a route requires custom groups, versioning, or other `ClassSerializerContextOptions`.
- Do not expose sensitive entity fields by default; explicitly exclude fields such as passwords, tokens, secrets, and internal identifiers.
- Preserve observable behavior in `intercept()` by transforming the value returned from `next.handle()` rather than manually subscribing.

Relationships

- DEPENDS_ON → `ClassSerializerInterceptorOptions`

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `AppController` — `sample/21-serializer/src/app.controller.ts`:10