PipeTransform

August 18, 2026

PipeTransform

Kind: Interface

Source: `packages/common/interfaces/features/pipe-transform.interface.ts`

Part of: Common

Interface describing implementation of a pipe.

`PipeTransform` defines the contract for pipes that transform or validate incoming values before they reach a route handler or other consumer. Implementations receive the raw value and argument metadata, then return a transformed value or throw an exception when validation fails.

Diagram

```
graph LR
  A[Incoming request value] --> B[PipeTransform.transform]
  C[ArgumentMetadata] --> B
  B -->|Valid / transformed value| D[Route handler parameter]
  B -->|Validation error| E[Exception response]
```

Usage

```
import {
  ArgumentMetadata,
  BadRequestException,
  PipeTransform,
} from '@nestjs/common';

export class ParsePositiveIntPipe implements PipeTransform<string, number> {
  transform(value: string, metadata: ArgumentMetadata): number {
    const parsedValue = Number(value);

    if (!Number.isInteger(parsedValue) || parsedValue <= 0) {
      throw new BadRequestException(
```

```
        `${metadata.data ?? 'value'} must be a positive integer`,
      );
    }

    return parsedValue;
  }
}

// Example route usage:
// @Get(':id')
// findOne(@Param('id', ParsePositiveIntPipe) id: number) {
//   return this.usersService.findOne(id);
// }
```

AI Coding Instructions

- Implement `transform(value, metadata)` and return the value type expected by the consuming handler.
- Use `ArgumentMetadata` to tailor validation behavior or error messages based on the parameter source and name.
- Throw framework HTTP exceptions, such as `BadRequestException`, for invalid input instead of returning invalid or partially transformed values.
- Keep pipes focused on input transformation and validation; delegate business rules and persistence checks to services.
- Ensure asynchronous pipes return a `Promise` and are registered where the value is bound, such as route parameters, request bodies, or global pipe configuration.

Used by

30 references from 30 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (30)

- `ParamProperties` — `packages/core/router/router-execution-context.ts`:50
- `UserByIdPipe` — `integration/hello-world/src/hello/users/user-by-id.pipe.ts`:4
- `UserByIdPipe` — `integration/hello-world/src/host/users/user-by-id.pipe.ts`:4
- `UserByIdPipe` — `integration/hello-world/src/host-array/users/user-by-id.pipe.ts`:4
- `UserByIdPipe` — `integration/inspector/src/circular-hello/users/user-by-id.pipe.ts`:4
- `ParseIntPipe` — `integration/inspector/src/common/pipes/parse-int.pipe.ts`:8
- `UserByIdPipe` — `integration/scopes/src/circular-hello/users/user-by-id.pipe.ts`:4

- `UserByIdPipe` — `integration/scopes/src/circular-transient/users/user-by-id.pipe.ts`:8

…and 22 more.