



CORE

ParamProperties

Kind: Interface

VERSION
47

UPDATED
7 August 2026

DOCUMENT
reference-subsystem-packages-core/interface-paramproperties

CONTENTS

1	Properties	2	4	AI Coding Instructions	3
2	Diagram	2	5	Relationships	4
3	Usage	3			

Kind: Interface

Source: `packages/core/router/router-execution-context.ts`

Part of: [Core](#)

`ParamProperties` describes how a controller method parameter is resolved during route execution. It stores the parameter's position, route parameter type, optional metadata, transformation pipes, and an extractor function that reads the value from the request lifecycle.

1 Properties

Property	Type
<code>index</code>	<code>number</code>
<code>type</code>	<code>`RouteParamtypes</code>
<code>data</code>	<code>ParamData</code>
<code>pipes</code>	<code>PipeTransform[]</code>
<code>extractValue</code>	<code><TRequest, TResponse>(req: TRequest, res: TResponse, next: Function,)</code>
<code>e</code>	<code>=> any</code>

2 Diagram

```
graph LR
    Request[HTTP Request] --> Extract[extractValue(req, res, next)]
    Extract --> RawValue[Raw parameter value]
    RawValue --> Pipes[PipeTransform[]]
    Pipes --> ResolvedValue[Resolved method argument]
    Metadata[index, type, data] --> Extract
    ResolvedValue --> Handler[Controller handler parameter]
```

3 Usage

```
import { RouteParamtypes } from '@nestjs/common/enums/route-paramtypes.enum';
import type { PipeTransform } from '@nestjs/common/interfaces';
import type { ParamProperties } from '@nestjs/core/router/router-execution-context';

const parseUserIdPipe: PipeTransform = {
  transform(value: string) {
    const id = Number(value);

    if (Number.isNaN(id)) {
      throw new Error('User ID must be a number');
    }

    return id;
  },
};

const userIdParameter: ParamProperties = {
  index: 0,
  type: RouteParamtypes.PARAM,
  data: 'id',
  pipes: [parseUserIdPipe],

  extractValue: (req) => {
    const request = req as { params: Record<string, string> };
    return request.params.id;
  },
};

// Represents the first argument passed to a controller handler:
// getUser(id: number) { ... }
```

4 AI Coding Instructions

- Keep `index` aligned with the controller handler's parameter position; indexes are zero-based.
- Use `type` and `data` consistently: for `RouteParamtypes.PARAM`, `data` typically contains the route parameter name such as `"id"`.
- Ensure `extractValue` safely handles the expected request shape and returns the raw value before pipes are applied.
- Add `PipeTransform` instances for validation or conversion rather than embedding transformation logic in controller handlers.
- Treat `ParamProperties` as router execution metadata; preserve its field contract when extending parameter decorator or route argument resolution behavior.

5 Relationships

- IMPORTS → `CanActivate`
- IMPORTS → `ForbiddenException`
- IMPORTS → `HttpServer`
- IMPORTS → `ParamData`
- IMPORTS → `PipeTransform`
- IMPORTS → `RequestMethod`
- IMPORTS → `CUSTOM_ROUTE_ARGS_METADATA`
- IMPORTS → `HEADERS_METADATA`
- IMPORTS → `HTTP_CODE_METADATA`
- IMPORTS → `REDIRECT_METADATA`
- IMPORTS → `RENDER_METADATA`
- IMPORTS → `ROUTE_ARGS_METADATA`
- IMPORTS → `SSE_METADATA`
- IMPORTS → `RouteParamMetadata`
- IMPORTS → `RouteParamtypes`
- IMPORTS → `ContextType`
- IMPORTS → `Controller`
- IMPORTS → `isEmpty`
- IMPORTS → `isString`