ContextUtils

August 18, 2026

ContextUtils

Kind: Class

Source: `packages/core/helpers/context-utils.ts`

Part of: Core

`ContextUtils` centralizes reflection and execution-context helpers used when NestJS resolves handler parameters. It reads callback metadata, aligns parameter metadata with reflected types, builds argument placeholders, and creates `ExecutionContextHost` instances for custom parameter factories.

Methods

Method	Signature	Returns
`mapParamType`	`mapParamType(key: string)`	`string`
`reflectCallbackParamtypes`	`reflectCallbackParamtypes(instance: Controller, methodName: string)`	`any[]`
`reflectCallbackMetadata`	`reflectCallbackMetadata(instance: Controller, methodName: string, metadataKey: string)`	`T`
`reflectPassthrough`	`reflectPassthrough(instance: Controller, methodName: string)`	`boolean`
`getArgumentsLength`	`getArgumentsLength(keys: string[], metadata: T)`	`number`
`createNullArray`	`createNullArray(length: number)`	`any[]`
`mergeParamsMetatypes`	`mergeParamsMetatypes(paramsProperties: ParamProperties[], paramtypes: any[])`	`(ParamProperties & { metatype?: any })[]`
`getCustomFactory`	`getCustomFactory(factory: (...args: unknown[]) => void, data: unknown,`	`(...args: unknown[]) => unknown`

Method	Signature	Returns
	`contextFactory: (args: unknown[]) => ExecutionContextHost)`	
`getContextFactory`	`getContextFactory(contextType: TContext, instance: object, callback: Function)`	`(args: unknown[]) => ExecutionContextHost`

Where it refuses work

- `ContextUtils` stops the work with an early return when `!paramtypes`.

Diagram

```
graph LR
  Handler[Controller/Resolver Handler] --> Reflection[Reflect Metadata]
  Reflection --> ContextUtils
  ContextUtils --> ParamTypes[Parameter Metatypes]
  ContextUtils --> Arguments[Argument Array]
  ContextUtils --> ContextFactory[ExecutionContextHost Factory]
  ContextFactory --> CustomFactory[Custom Parameter Factory]
  Arguments --> Handler
  CustomFactory --> Handler
```

Usage

```
import { ContextUtils } from '@nestjs/core/helpers/context-utils';
import { ExecutionContextHost } from '@nestjs/core/helpers/execution-context-host';

class UsersController {
  findOne(id: string) {
    return { id };
  }
}

const contextUtils = new ContextUtils();
const controller = new UsersController();

// Read TypeScript design:paramtypes metadata for the handler.
const paramTypes = contextUtils.reflectCallbackParamtypes(
  controller,
  'findOne',
);

// Create an execution-context factory for custom parameter decorators.
const createContext = contextUtils.getContextFactory('http');
```

```
const args = [{ params: { id: '42' } }, {}, () => undefined];
const context: ExecutionContextHost = createContext(args);

console.log(paramTypes); // Example: [String]
console.log(context.getType()); // "http"
```

AI Coding Instructions

- Use `reflectCallbackParamtypes()` and `reflectCallbackMetadata()` instead of reading `Reflect` metadata directly in parameter-resolution code.
- Keep parameter indexes aligned when using `getArgumentsLength()`, `createNullArray()`, and `mergeParamsMetatypes()`.
- Create contexts through `getContextFactory()` so the resulting `ExecutionContextHost` has the correct transport type.
- Wrap custom parameter decorators with `getCustomFactory()` to ensure they receive both decorator data and the generated execution context.
- Treat missing reflection metadata as valid; handlers may not have emitted design-time metadata.

Used by

3 references from 3 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (3)

- `RpcHandlerMetadata` — `packages/microservices/context/rpc-context-creator.ts`:35
- `WsHandlerMetadata` — `packages/websockets/context/ws-context-creator.ts`:34
- `MessageMappingProperties` — `packages/websockets/gateway-metadata-explorer.ts`:15