PipesContextCreator

August 17, 2026

PipesContextCreator

Kind: Class

Source: `packages/core/pipes/pipes-context-creator.ts`

Part of: Core

`PipesContextCreator` is a framework-level base class that resolves pipe metadata into executable `PipeTransform` instances for a handler or controller. It combines locally declared pipes with globally configured pipes, resolves request-scoped instances through the module container, and tracks the active module context during resolution.

Extends: `ContextCreator`

Methods

Method	Signature	Returns
`create`	`create(instance: Controller, callback: (...args: unknown[]) => unknown, moduleKey: string, contextId: undefined, inquirerId: string)`	`PipeTransform[]`
`createConcreteContext`	`createConcreteContext(metadata: T, contextId: undefined, inquirerId: string)`	`R`
`getPipeInstance`	`getPipeInstance(pipe: Function	PipeTransform, contextId: undefined, inquirerId: string)`
`getInstanceByMetatype`	`getInstanceByMetatype(metatype: Type<unknown>)`	`InstanceWrapper
`getGlobalMetadata`	`getGlobalMetadata(contextId: undefined, inquirerId: string)`	`T`
`setModuleContext`	`setModuleContext(context: string)`	`void`

Where it refuses work

- `PipesContextCreator` stops the work with an early return when `isEmpty(metadata)`.
- `PipesContextCreator` stops the work with an early return when `isObject`.
- `PipesContextCreator` stops the work with an early return when `!instanceWrapper`.
- `PipesContextCreator` stops the work with an early return when `!this.moduleContext`.
- `PipesContextCreator` stops the work with an early return when `!moduleRef`.
- `PipesContextCreator` stops the work with an early return when `!this.config`.

Diagram

```
graph LR
  A[Controller / Handler Metadata] --> B[PipesContextCreator.create]
  B --> C[Local Pipe Metadata]
  B --> D[Global Pipe Metadata]
  C --> E[createConcreteContext]
  D --> E
  E --> F[getPipeInstance]
  F --> G[Module Container]
  G --> H[InstanceWrapper]
  H --> I[PipeTransform instances]
```

Usage

```
import { Injectable, PipeTransform } from '@nestjs/common';
import { PipesContextCreator } from '@nestjs/core/pipes/pipes-context-creator';

@Injectable()
class TrimPipe implements PipeTransform {
  transform(value: unknown) {
    return typeof value === 'string' ? value.trim() : value;
  }
}

// PipesContextCreator is intended for framework adapters and internal
// integrations. Concrete implementations provide container/config access.
class CustomPipesContextCreator extends PipesContextCreator {
  protected pipesContainer = appContainer;
  protected config = applicationConfig;

  createConcreteContext(metadata: unknown[]): PipeTransform[] {
    return metadata
      .map(pipe => this.getPipeInstance(pipe))
      .filter((pipe): pipe is PipeTransform => pipe !== null);
  }
```

```
}

const pipesCreator = new CustomPipesContextCreator();

// Resolve pipes declared on a handler within its owning module.
const pipes = pipesCreator.create(
  UsersController,
  UsersController.prototype.create,
  'UsersModule',
);

// Example result: [TrimPipe instance, ...global pipes]
const transformedValue = await pipes[0].transform('  Ada  ', {});
```

AI Coding Instructions

- Treat `PipesContextCreator` as infrastructure code: extend it through a concrete context creator rather than instantiating it directly.
- Always set or pass the correct module context before resolving metatype-based pipes; pipe providers are resolved from the active module's injectable collection.
- Support both pipe instances and pipe classes: existing objects with a `transform()` method should be reused, while classes should be resolved through the container.
- Preserve request-scoped behavior by forwarding `contextId` and `inquirerId` when resolving pipe instances.
- Include global pipe metadata when building a concrete pipe context, including request-scoped global pipes when the context is non-static.

How it works

`PipesContextCreator` is a `ContextCreator` subclass that builds arrays of `PipeTransform` instances from global configuration and pipe metadata attached to a controller class or handler callback. It stores a `NestContainer` and optional `ApplicationConfig`; the constructor performs no other work. [packages/core/pipes/pipes-context-creator.ts:11-19]

- `create(instance, callback, moduleKey, contextId, inquirerId)` records `moduleKey` as its current module context, then delegates to the inherited `createContext` with the `PIPES_METADATA` key. [packages/core/pipes/pipes-context-creator.ts:21-35]
- The inherited operation reads global metadata, metadata on `instance`'s prototype constructor, and metadata on `callback`; it converts each group through

`createConcreteContext` and returns them in global, class, then method order. [packages/core/helpers/context-creator.ts:16-40]

- `@UsePipes` writes pipe classes or instances into `PIPES_METADATA` on either the controller target or a method descriptor's callback, which is the metadata consumed by `create`. [packages/common/decorators/core/use-pipes.decorator.ts:29-47]

Concrete-pipe resolution

- `createConcreteContext(metadata, contextId, inquirerId)` returns an empty array when `metadata` is empty or absent. [packages/core/pipes/pipes-context-creator.ts:38-45]
- Otherwise, it discards falsy entries and entries that have neither a `name` nor `transform` property, resolves each remaining item with `getPipeInstance`, then retains only resolved values whose `transform` member exists and is a function. [packages/core/pipes/pipes-context-creator.ts:46-50]
- `getPipeInstance` returns an input object directly when it has a truthy `transform` property. For a non-object pipe reference, it looks up an `InstanceWrapper` by metatype; if none is found, it returns `null`. [packages/core/pipes/pipes-context-creator.ts:53-65]
- For a found wrapper, it retrieves the instance for a context ID and optional inquirer ID, returning the host's `instance`. Before lookup, the context ID is replaced by a parent context when the supplied context ID has `getParent`; the parent lookup receives the wrapper token and dependency-tree durability flag. [packages/core/pipes/pipes-context-creator.ts:66-70] [packages/core/helpers/context-creator.ts:55-65]
- Metatype lookup requires a current module context and a matching module in `container.getModules()`; missing context or module returns `undefined`. A matching module is queried through `moduleRef.injectables.get(metatype)`. [packages/core/pipes/pipes-context-creator.ts:73-85]
- `setModuleContext(context)` changes the stored module context used by later class-based pipe lookups. [packages/core/pipes/pipes-context-creator.ts:114-116]

Global pipes and scoped contexts

- Without an `ApplicationConfig`, `getGlobalMetadata` returns an empty array. [packages/core/pipes/pipes-context-creator.ts:87-93]
- With configuration, a static context with no inquirer ID returns the configured global pipe array directly. [packages/core/pipes/pipes-context-creator.ts:94-97]

- For another context or when an inquirer ID is present, it obtains configured request-scoped global pipe wrappers, gets each wrapper's contextual instance with the adjusted context ID and inquirer ID, drops missing hosts, and appends their instances after the ordinary global pipes. [packages/core/pipes/pipes-context-creator.ts:98-111]
- `ApplicationConfig` stores ordinary global pipes separately from global request-pipe wrappers; its accessors return those respective arrays. [packages/core/application-config.ts:16-21] [packages/core/application-config.ts:76-78] [packages/core/application-config.ts:114-120]

Execution boundary

- This class assembles pipe instances; it does not call `transform`. `PipesConsumer.applyPipes` invokes each selected pipe's `transform` in array order, passing the prior result and argument metadata. [packages/core/pipes/pipes-consumer.ts:19-29]
- In the external-context path, `ExternalContextCreator` constructs this class with the container and application config, calls `create` while creating a handler context, and combines handler-level pipes with parameter-specific pipes before passing them to the pipe consumer. [packages/core/helpers/external-context-creator.ts:56-88] [packages/core/helpers/external-context-creator.ts:109-120] [packages/core/helpers/external-context-creator.ts:292-329]
- For parameter-specific metadata in that path, `exchangeKeysForValues` first sets this creator's module context and then calls `createConcreteContext` for each parameter's pipe collection. [packages/core/helpers/external-context-creator.ts:255-289]

Visible validation and failure behavior

- The class has no explicit `throw` statements or error translation. Its visible rejection behavior is omission: empty metadata yields no pipes, unresolved class references yield `null` and are filtered out, and values without a callable `transform` are filtered out. [packages/core/pipes/pipes-context-creator.ts:43-50] [packages/core/pipes/pipes-context-creator.ts:62-65]
- Its visible mutable side effect is assignment of `moduleContext` by `create` or `setModuleContext`. [packages/core/pipes/pipes-context-creator.ts:28] [packages/core/pipes/pipes-context-creator.ts:114-116]

Relationships

- IMPORTS → `PIPES_METADATA`

- IMPORTS → `Controller`
- IMPORTS → `PipeTransform`
- IMPORTS → `Type`
- IMPORTS → `isEmpty`
- IMPORTS → `isFunction`

Used by

4 references from 4 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (4)

- `RpcHandlerMetadata` — `packages/microservices/context/rpc-context-creator.ts`:35
- `MicroservicesModule` — `packages/microservices/microservices-module.ts`:23
- `WsHandlerMetadata` — `packages/websockets/context/ws-context-creator.ts`:34
- `SocketModule` — `packages/websockets/socket-module.ts`:27