ValueProvider

August 18, 2026

ValueProvider

Kind: Interface

Source: `packages/common/interfaces/modules/provider.interface.ts`

Part of: Common

Interface defining a *Value* type provider.

For example:

```
const connectionProvider = {
  provide: 'CONNECTION',
  useValue: connection,
};
```

`ValueProvider` defines a dependency injection provider that registers an existing value under an `InjectionToken`. Use it when a dependency is already created—such as a configuration object, database connection, or third-party client—and should be shared through the container without factory logic.

Properties

Property	Type
`provide`	`InjectionToken`
`useValue`	`T`
`inject`	`never`

Diagram

```
graph LR
  A[Existing value] --> B[ValueProvider]
  B -->|provide: InjectionToken| C[Dependency Injection Container]
```

```
  C -->|resolve token| D[Consumer]
  B -->|useValue: T| A
```

Usage

```ts
import type { ValueProvider } from '@nestjs/common';

const connection = createDatabaseConnection();

const connectionProvider: ValueProvider = {
  provide: 'CONNECTION',
  useValue: connection,
};

// Consumers can inject the registered value using the same token.
class UserRepository {
  constructor(
    @Inject('CONNECTION')
    private readonly connection: DatabaseConnection,
  ) {}
}
```

AI Coding Instructions

- Use `useValue` for pre-existing objects, constants, configuration, mocks, or initialized third-party clients.
- Ensure the `provide` token exactly matches the token used by consuming classes or factories.
- Do not add `inject` dependencies to a value provider; `useValue` is supplied directly and `inject` is `never`.
- Prefer symbols or exported token constants over repeated string literals for shared application dependencies.
- Use `ValueProvider` in tests to replace real dependencies with deterministic mock objects.

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)

- `isClassProvider` — `packages/core/injector/helpers/provider-classifier.ts`:9
- `InternalCoreModule` — `packages/core/injector/internal-core-module/internal-core-module.ts`:16
- `Module` — `packages/core/injector/module.ts`:44
- `DependenciesScanner` — `packages/core/scanner.ts`:75