ModuleFactory

August 18, 2026

ModuleFactory

Kind: Interface

Source: `packages/core/injector/compiler.ts`

Part of: Core

`ModuleFactory` describes the compiled representation of a module used by the injector/compiler pipeline. It links a module class (`type`) to its unique registration token and any runtime-provided `dynamicMetadata` required to construct or configure a dynamic module.

Properties

Property	Type
`type`	`Type<any>`
`token`	`string`
`dynamicMetadata`	`Partial<DynamicModule>`

Diagram

```
graph LR
  A[Module Class<br/>Type&lt;any&gt;] --> B[ModuleFactory]
  C[Dynamic Module Metadata<br/>Partial&lt;DynamicModule&gt;] --> B
  B --> D[token: string]
  B --> E[Injector / Compiler]
  E --> F[Resolved Module Instance]
```

Usage

```
import type { DynamicModule, Type } from '@nestjs/common';
```

```
interface ModuleFactory {
  type: Type<any>;
  token: string;
  dynamicMetadata: Partial<DynamicModule>;
}

class DatabaseModule {}

const databaseModuleFactory: ModuleFactory = {
  type: DatabaseModule,
  token: 'database-module-token',
  dynamicMetadata: {
    providers: [
      {
        provide: 'DATABASE_URL',
        useValue: process.env.DATABASE_URL,
      },
    ],
    exports: ['DATABASE_URL'],
  },
};

// Pass the compiled module definition to the injector/compiler flow.
registerCompiledModule(databaseModuleFactory);
```

AI Coding Instructions

- Preserve `token` stability: it should uniquely identify the module and its dynamic configuration within the container.
- Use `type` for the concrete module class; do not replace it with an instance or plain object.
- Treat `dynamicMetadata` as optional/partial metadata and guard against missing properties before reading arrays such as `providers`, `imports`, or `exports`.
- Ensure dynamic module metadata matches the `DynamicModule` contract so injector resolution can register providers and exports correctly.
- When generating factories, keep token creation and metadata normalization centralized in the compiler pipeline.

Relationships

- IMPORTS → `DynamicModule`
- IMPORTS → `ForwardReference`
- IMPORTS → `Type`