InstanceLoader

August 17, 2026

InstanceLoader

Kind: Class

Source: `packages/core/injector/instance-loader.ts`

Part of: Core

`InstanceLoader` coordinates the creation of provider, controller, and injectable instances for modules registered in the dependency injection container. During application bootstrap, it delegates dependency resolution to the injector and can report initialization progress through a configured logger.

Methods

Method	Signature	Returns
`setLogger`	`setLogger(logger: Logger)`	`void`
`createInstancesOfDependencies`	`createInstancesOfDependencies(modules: Map<string, Module>)`	`void`

When something fails

- `InstanceLoader` handles failure in 1 place: it lets it reach the caller in all 1.

Diagram

```
graph LR
  A[Application Bootstrap] --> B[InstanceLoader]
  B --> C[Container Modules]
  B --> D[Injector]
  B --> E[Logger]
  C --> F[Providers]
  C --> G[Injectables]
  C --> H[Controllers]
  D --> F
```

```
  D --> G
  D --> H
```

Usage

```
import { InstanceLoader } from '@nestjs/core/injector/instance-loader';
import { Injector } from '@nestjs/core/injector/injector';

// Typically created internally by the Nest application bootstrap process.
const instanceLoader = new InstanceLoader(
  container,
  new Injector(),
  graphInspector,
);

instanceLoader.setLogger(logger);

// Resolves and creates providers, injectables, and controllers
// for all modules in the container.
await instanceLoader.createInstancesOfDependencies();
```

AI Coding Instructions

- Treat `InstanceLoader` as a bootstrap-level internal service; application code should normally rely on Nest's standard application factory instead of constructing it directly.
- Ensure modules are registered in the container before calling `createInstancesOfDependencies()`.
- Preserve the dependency creation order: providers and injectables must be available before controllers are instantiated.
- Use `setLogger()` to integrate bootstrap progress reporting without coupling instance creation logic to a specific logger implementation.
- Keep dependency resolution delegated to `Injector`; avoid adding direct constructor-resolution logic to `InstanceLoader`.

Relationships

- IMPORTS → `Logger`
- IMPORTS → `LoggerService`
- IMPORTS → `Controller`
- IMPORTS → `Injectable`

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `TestingInstanceLoader` — `packages/testing/testing-instance-loader.ts`:6