OverrideByFactoryOptions

August 18, 2026

OverrideByFactoryOptions

Kind: Interface

Source: `packages/testing/interfaces/override-by-factory-options.interface.ts`

Part of: Testing

`OverrideByFactoryOptions` configures a testing override that creates a replacement provider through a factory function. The `inject` array declares the dependencies passed to the factory, allowing test modules to compose mocks or custom implementations from other registered providers.

Properties

Property	Type
`factory`	`(...args: any[]) => any`
`inject`	`any[]`

Diagram

```
graph LR
  A[Test Module] --> B[OverrideByFactoryOptions]
  B --> C[factory(...args)]
  D[inject tokens] --> E[Resolved dependencies]
  E --> C
  C --> F[Replacement provider instance]
```

Usage

```
import { Test } from '@nestjs/testing';

const mockUsersService = {
  findAll: jest.fn().mockResolvedValue([]),
```

```
};

const moduleRef = await Test.createTestingModule({
  providers: [UsersService, ConfigService],
})
  .overrideProvider(UsersService)
  .useFactory({
    inject: [ConfigService],
    factory: (configService: ConfigService) => ({
      ...mockUsersService,
      findAll: jest.fn().mockResolvedValue([
        { id: 1, environment: configService.get('NODE_ENV') },
      ]),
    }),
  })
  .compile();
```

AI Coding Instructions

- Use `factory` when an override needs to be built dynamically from other providers instead of returning a fixed mock.
- Include every factory dependency in `inject` and keep the array order aligned with the factory function parameters.
- Return the complete replacement provider implementation expected by the code under test.
- Prefer stable mock behavior in factories; avoid unnecessary external state or side effects.
- Ensure injected dependency tokens are available in the testing module before compiling it.