Lifecycle Events

Every application element has a lifecycle managed by Nest. Nest offers **lifecycle hooks** that provide visibility into key life moments and the ability to act when they occur.

Lifecycle sequence

After creating a injectable/controller by calling its constructor, Nest calls the lifecycle hook methods in the following sequence at specific moments:

`OnModuleInit`	Called once the host module has been initialized
`OnApplicationBootstrap`	Called once the application has fully started and is bootstrapped
`OnModuleDestroy`	Cleanup just before Nest destroys the host module (`app.close()` method has been evaluated)
`OnApplicationShutdown`	Responds to the system signals (when application gets shutdown by e.g. `SIGTERM`)

Usage

Each lifecycle hook is represented by interface. Interfaces are technically optional because they do not exist anyway after TypeScript compilation. Nonetheless, it's a good practice to use them in order to benefit from strong typing and editor tooling.

JS

```
import { Injectable, OnModuleInit } from '@nestjs/common';

@Injectable()
export class UsersService implements OnModuleInit {
  onModuleInit() {
    console.log(`The module has been initialized.`);
  }
}
```

Additionally, both `OnModuleInit` and `OnApplicationBootstrap` hooks allow you to defer the application initialization process (return a `Promise` or mark the method as `async`).

```
JS

async onModuleInit(): Promise<void> {
  await this.fetch();
}
```

OnApplicationShutdown

The `OnApplicationShutdown` responds to the system signals (when application gets shutdown by e.g. `SIGTERM`). Use this hook to gracefully shutdown a Nest application. This feature is often used with **Kubernetes**, **Heroku** or similar services.

To use this hook you must activate a listener which listens to shutdown signals.

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  // Starts listening to shutdown hooks
  app.enableShutdownHooks();
  await app.listen(3000);
}
bootstrap();
```

If the application receives a signal it will call the `onApplicationShutdown` function of your `Injectable` with the corresponding signal as first parameter. If your function does return a promise, it will not shutdown your Nest application until the promise is resolved or rejected.

```
JS

@Injectable()
class UsersService implements OnApplicationShutdown {
  onApplicationShutdown(signal: string) {
    console.log(signal); // e.g. "SIGINT"
  }
}
```

Support us

Nest is an MIT-licensed open source project. It can grow thanks to the support by these awesome people. If you'd like to join them, please read more here.

Principal Sponsor

Sponsors / Partners