Security

In this chapter you will learn some techniques that will allow you to increase the security of your applications.

Helmet

Helmet can help protect your app from some well-known web vulnerabilities by setting HTTP headers appropriately. Generally, Helmet is just a collection of 12 smaller middleware functions that set security-related HTTP headers (read **more**). Firstly, install the required package:

```
$ npm i --save helmet
```

Once the installation is completed, apply it as a global middleware.

```
import * as helmet from 'helmet';
// somewhere in your initialization file
app.use(helmet());
```

CORS

Cross-origin resource sharing (CORS) is a mechanism that allows resources to be requested from another domain. Under the hood, Nest makes use of **cors** package, that provides a bunch of options that you may customize based on your requirements. In order to enable CORS, you have to call `enableCors()` method.

```
const app = await NestFactory.create(ApplicationModule);
app.enableCors();
await app.listen(3000);
```

Also, you can pass a configuration object as a parameter of this function. The available properties are exhaustively described in the official **cors** repository. A different way is to use a Nest options object:

```
const app = await NestFactory.create(ApplicationModule, { cors: true });
await app.listen(3000);
```

Instead of passing a boolean value, you can use a cors configuration object as well (read **more**).

CSRF

Cross-site request forgery (known as CSRF or XSRF) is a type of malicious exploit of a website where **unauthorized** commands are transmitted from a user that the web application trusts. To mitigate this kind of attacks you can use the **csurf** package. Firstly, install the required package:

```
$ npm i --save csurf
```

Once the installation is completed, apply it as a global middleware.

```
import * as csurf from 'csurf';
// somewhere in your initialization file
app.use(csurf());
```

Rate limiting

To protect your applications from brute-force attacks, you have to implement some kind of rate-limiting. Luckily, there is a bunch of various middleware available on the NPM already. One of them is **express-rate-limit**.

```
$ npm i --save express-rate-limit
```

Once the installation is completed, apply it as a global middleware.

```
import * as rateLimit from 'express-rate-limit';
// somewhere in your initialization file
app.use(
  rateLimit({
    windowMs: 15 * 60 * 1000, // 15 minutes
    max: 100, // limit each IP to 100 requests per windowMs
  }),
);
```

> **HINT**
> If you work with `FastifyAdapter`, consider using **fastify-rate-limit** instead.

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