Raw body

One of the most common use-case for having access to the raw request body is performing webhook signature verifications. Usually to perform webhook signature validations the unserialized request body is required to calculate an HMAC hash.

> warning **Warning** This feature can be used only if the built-in global body parser middleware is enabled, ie., you must not pass `bodyParser: false` when creating the app.

Use with Express

First enable the option when creating your Nest Express application:

```
import { NestFactory } from '@nestjs/core';
import type { NestExpressApplication } from '@nestjs/platform-express';
import { AppModule } from './app.module';

// in the "bootstrap" function
const app = await NestFactory.create<NestExpressApplication>(AppModule, {
  rawBody: true,
});
await app.listen(3000);
```

To access the raw request body in a controller, a convenience interface `RawBodyRequest` is provided to expose a `rawBody` field on the request: use the interface `RawBodyRequest` type:

```
import { Controller, Post, RawBodyRequest, Req } from '@nestjs/common';
import { Request } from 'express';

@Controller('cats')
class CatsController {
  @Post()
  create(@Req() req: RawBodyRequest<Request>) {
    const raw = req.rawBody; // returns a `Buffer`.
  }
}
```

Registering a different parser

By default, only `json` and `urlencoded` parsers are registered. If you want to register a different parser on the fly, you will need to do so explicitly.

For example, to register a `text` parser, you can use the following code:

```
app.useBodyParser('text');
```

> warning **Warning** Ensure that you are providing the correct application type to the `NestFactory.create` call. For Express applications, the correct type is `NestExpressApplication`. Otherwise the `.useBodyParser` method will not be found.

Body parser size limit

If your application needs to parse a body larger than the default `100kb` of Express, use the following:

```
app.useBodyParser('json', { limit: '10mb' });
```

The `.useBodyParser` method will respect the `rawBody` option that is passed in the application options.

Use with Fastify

First enable the option when creating your Nest Fastify application:

```
import { NestFactory } from '@nestjs/core';
import {
  FastifyAdapter,
  NestFastifyApplication,
} from '@nestjs/platform-fastify';
import { AppModule } from './app.module';

// in the "bootstrap" function
const app = await NestFactory.create<NestFastifyApplication>(
  AppModule,
  new FastifyAdapter(),
  {
    rawBody: true,
  },
);
await app.listen(3000);
```

To access the raw request body in a controller, a convenience interface `RawBodyRequest` is provided to expose a `rawBody` field on the request: use the interface `RawBodyRequest` type:

```
import { Controller, Post, RawBodyRequest, Req } from '@nestjs/common';
import { FastifyRequest } from 'fastify';

@Controller('cats')
class CatsController {
  @Post()
  create(@Req() req: RawBodyRequest<FastifyRequest>) {
    const raw = req.rawBody; // returns a `Buffer`.
  }
}
```

Registering a different parser

By default, only `application/json` and `application/x-www-form-urlencoded` parsers are registered. If you want to register a different parser on the fly, you will need to do so explicitly.

For example, to register a `text/plain` parser, you can use the following code:

```
app.useBodyParser('text/plain');
```

> warning **Warning** Ensure that you are providing the correct application type to the `NestFactory.create` call. For Fastify applications, the correct type is `NestFastifyApplication`. Otherwise the `.useBodyParser` method will not be found.

Body parser size limit

If your application needs to parse a body larger than the default 1MiB of Fastify, use the following:

```
const bodyLimit = 10_485_760; // 10MiB
app.useBodyParser('application/json', { bodyLimit });
```

The `.useBodyParser` method will respect the `rawBody` option that is passed in the application options.