HTTP adapter

Occasionally, you may want to access the underlying HTTP server, either within the Nest application context or from the outside.

Basically, every native (platform-specific) HTTP server/library instance is wrapped in an **adapter**. The adapter is registered as a globally available provider that might be plucked from the application context as well as injected into other providers easily.

Outside strategy

In order to get the `HttpAdapter` from the outside of the application context, you can call `getHttpAdapter()` method.

```js
const app = await NestFactory.create(ApplicationModule);
const httpAdapter = app.getHttpAdapter();
```

In-context strategy

In order to get the `HttpAdapterHost` from within the application context, you can inject it in the same way as any other existing provider (let's say, through `constructor`).

```js
export class CatsService {
  constructor(private readonly adapterHost: HttpAdapterHost) {}
}
```

> **HINT**
> The `HttpAdapterHost` is imported from the `@nestjs/core` package.

Adapter host

So far, we have learnt how to get the `HttpAdapterHost` . However, it's still not an actual `HttpAdapter` . In order to get the `HttpAdapter` , simply access the `httpAdapter` property.

```
const adapterHost = app.get(HttpAdapterHost);
const httpAdapter = adapterHost.httpAdapter;
```

The `httpAdapter` is a real instance of the HTTP adapter used by the framework underneath. It can be either `ExpressAdapter` or `FastifyAdapter` (both classes extend `AbstractHttpAdapter`).

Every adapter exposes several useful methods to interact with the HTTP server. Nonetheless, if you want to access the library reference directly, call `getInstance()` method.

```
const instance = httpAdapter.getInstance();
```

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