HTTP adapter

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

Every native (platform-specific) HTTP server/library (e.g., Express and Fastify) instance is wrapped in an **adapter**. The adapter is registered as a globally available provider that can be retrieved from the application context, as well as injected into other providers.

Outside application context strategy

To get a reference to the `HttpAdapter` from outside of the application context, call the `getHttpAdapter()` method.

```
@@filename()
const app = await NestFactory.create(AppModule);
const httpAdapter = app.getHttpAdapter();
```

In-context strategy

To get a reference to the `HttpAdapterHost` from within the application context, inject it using the same technique as any other existing provider (e.g., using constructor injection).

```
@@filename()
export class CatsService {
  constructor(private adapterHost: HttpAdapterHost) {}
}
@@switch
@Dependencies(HttpAdapterHost)
export class CatsService {
  constructor(adapterHost) {
    this.adapterHost = adapterHost;
  }
}
```

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

The `HttpAdapterHost` is **not** an actual `HttpAdapter`. To get the actual `HttpAdapter` instance, simply access the `httpAdapter` property.

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

The `httpAdapter` is the actual instance of the HTTP adapter used by the underlying framework. It is an instance of either `ExpressAdapter` or `FastifyAdapter` (both classes extend `AbstractHttpAdapter`).

The adapter object exposes several useful methods to interact with the HTTP server. However, if you want to access the library instance (e.g., the Express instance) directly, call the `getInstance()` method.

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