



PLATFORM EXPRESS

ExpressAdapter

Kind: Class

VERSION
47

UPDATED
7 August 2026

DOCUMENT

reference-subsystem-packages-platform-express/class-expressadapter

CONTENTS

1	Methods	2	5	Usage	5
2	Where it refuses work	4	6	AI Coding Instructions	6
3	When something fails	4	7	How it works	6
4	Diagram	5	8	Relationships	7

Kind: Class

Source: `packages/platform-express/adapters/express-adapter.ts`

Part of: Platform Express

`ExpressAdapter` connects the platform-agnostic HTTP adapter contract to an Express application. It manages request and response hooks, response helpers, rendering and redirects, and Express-level error or not-found handlers.

Extends: `AbstractHttpAdapter`

1 Methods

Method	Signature	Returns
<code>setOnRequestHook</code>	<code>`setOnRequestHook(onRequestHook: (req: express.Request, res: express.Response, done: () => void,) => Promise</code>	<code>void)`</code>
<code>setOnResponseHook</code>	<code>`setOnResponseHook(onResponseHook: (req: express.Request, res: express.Response,) => Promise</code>	<code>void)`</code>
<code>reply</code>	<code>reply(response: any, body: any, statusCode: number)</code>	<code>void</code>
<code>status</code>	<code>status(response: any, statusCode: number)</code>	<code>void</code>
<code>end</code>	<code>end(response: any, message: string)</code>	<code>void</code>
<code>render</code>	<code>render(response: any, view: string, options: any)</code>	<code>void</code>
<code>redirect</code>	<code>redirect(response: any, statusCode: number, url: string)</code>	<code>void</code>
<code>setErrorHandler</code>	<code>setErrorHandler(handler: Function, prefix: string)</code>	<code>void</code>
<code>setNotFoundHandler</code>	<code>setNotFoundHandler(handler: Function, prefix: string)</code>	<code>void</code>
<code>isHeadersSent</code>	<code>isHeadersSent(response: any)</code>	<code>boolean</code>
<code>getHeader</code>	<code>getHeader(response: any, name: string)</code>	<code>void</code>
<code>setHeader</code>	<code>setHeader(response: any, name: string, value: string)</code>	<code>void</code>
<code>appendHeader</code>	<code>appendHeader(response: any, name: string, value: string)</code>	<code>void</code>

<code>normalizePath</code>	<code>normalizePath(path: string)</code>	<code>string</code>
<code>listen</code>	<code>`listen(port: string</code>	<code>number, callback: () => void)`</code>
<code>listen</code>	<code>`listen(port: string</code>	<code>number, hostname: string, callback: () => void)`</code>
<code>listen</code>	<code>listen(port: any, args: any[])</code>	<code>Server</code>
<code>close</code>	<code>close()</code>	<code>void</code>
<code>set</code>	<code>set(args: any[])</code>	<code>void</code>
<code>enable</code>	<code>enable(args: any[])</code>	<code>void</code>
<code>disable</code>	<code>disable(args: any[])</code>	<code>void</code>
<code>engine</code>	<code>engine(args: any[])</code>	<code>void</code>
<code>useStaticAssets</code>	<code>useStaticAssets(path: string, options: ServeStaticOptions)</code>	<code>void</code>
<code>setBaseViewsDir</code>	<code>`setBaseViewsDir(path: string</code>	<code>string[])`</code>
<code>setViewEngine</code>	<code>setViewEngine(engine: string)</code>	<code>void</code>
<code>getRequestHostname</code>	<code>getRequestHostname(request: any)</code>	<code>string</code>
<code>getRequestMethod</code>	<code>getRequestMethod(request: any)</code>	<code>string</code>
<code>getRequestUrl</code>	<code>getRequestUrl(request: any)</code>	<code>string</code>
<code>enableCors</code>	<code>`enableCors(options: CorsOptions</code>	<code>CorsOptionsDelegate)`</code>
<code>createMiddlewareFactory</code>	<code>createMiddlewareFactory(requestMethod: RequestMethod)</code>	<code>(path: string, callback: Function) => any</code>
<code>initHttpServer</code>	<code>initHttpServer(options: NestApplicationOptions)</code>	<code>void</code>

<code>registerParserMiddleware</code>	<code>registerParserMiddleware(prefix: string, rawBody: boolean)</code>	<code>void</code>
<code>useBodyParser</code>	<code>useBodyParser(type: NestExpressBodyParserType, rawBody: boolean, options: Omit<Options, 'verify'>)</code>	<code>this</code>
<code>setLocal</code>	<code>setLocal(key: string, value: any)</code>	<code>void</code>
<code>getType</code>	<code>getType()</code>	<code>string</code>
<code>applyVersionFilter</code>	<code>applyVersionFilter(handler: Function, version: VersionValue, versioningOptions: VersioningOptions)</code>	<code>Versioned Route</code>

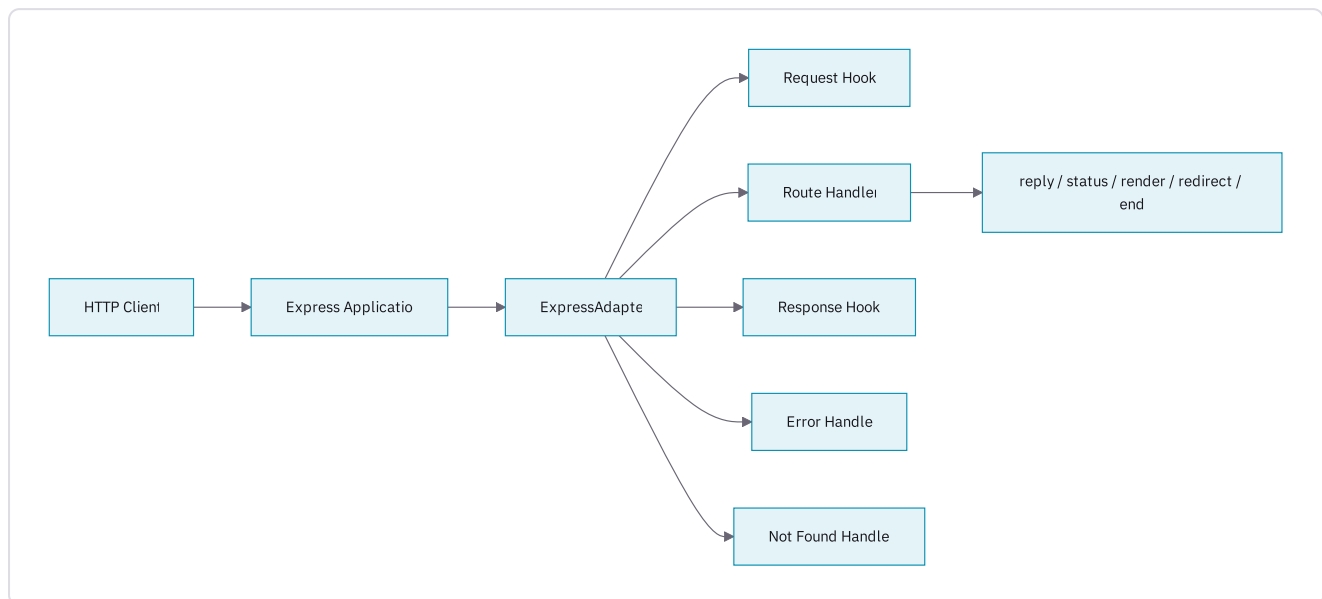
2 Where it refuses work

- `ExpressAdapter` stops the work with `InternalServerErrorException` when `!next` — “HTTP adapter does not support filtering on version”.
- `ExpressAdapter` stops the work with an early return when `version.includes(VERSION_NEUTRAL)`, in 2 places.
- `ExpressAdapter` stops the work with an early return when `isNil(body)`.
- `ExpressAdapter` stops the work with an early return when `!this.httpServer`.
- `ExpressAdapter` stops the work with an early return when `options && options.prefix`.
- `ExpressAdapter` stops the work with an early return when `Array.isArray(extractedVersion) && version.filter(v => extractedVersion.includes(v as str...`.

3 When something fails

- `ExpressAdapter` handles failure in 2 places: it lets it reach the caller in all 2.

4 Diagram



5 Usage

```
import express from 'express';
import { ExpressAdapter } from '@nestjs/platform-express';

const server = express();
const adapter = new ExpressAdapter(server);

adapter.setOnRequestHook((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
});

adapter.setOnResponseHook((req, res, next) => {
  res.on('finish', () => {
    console.log(`Response completed with ${res.statusCode}`);
  });
  next();
});

server.get('/health', (_req, res) => {
  adapter.reply(res, { status: 'ok' }, 200);
});

server.get('/dashboard', (_req, res) => {
  adapter.render(res, 'dashboard', { title: 'Dashboard' });
});

adapter.setNotFoundHandler((req, res) => {
```

```
adapter.reply(res, { message: `Route not found: ${req.url}` }, 404);
});

adapter.setErrorHandler((error, _req, res, _next) => {
  console.error(error);

  if (!adapter.isHeadersSent(res)) {
    adapter.reply(res, { message: 'Internal server error' }, 500);
  }
});

server.listen(3000);
```

6 AI Coding Instructions

- Use adapter response helpers such as `reply()`, `status()`, `redirect()`, and `end()` instead of directly coupling shared platform code to Express response APIs.
- Register request and response hooks early, before routes or framework initialization, so they apply consistently to all requests.
- Check `isHeadersSent(response)` before writing an error response to prevent duplicate headers or response-write errors.
- Ensure custom error and not-found handlers terminate the response and preserve Express middleware signatures.
- Keep Express-specific middleware and view-engine configuration at the platform boundary; application logic should remain adapter-agnostic.

7 How it works

`ExpressAdapter` is a public class that extends Nest's `AbstractHttpAdapter` with an Express application instance and an HTTP or HTTPS Node server type. Its constructor accepts an optional application instance; otherwise it creates one with `express()`. [packages/platform-express/adapters/express-adapter.ts:48-53] [packages/platform-express/adapters/express-adapter.ts:67-82]

8 Relationships

- IMPORTS → `HttpStatus`
- IMPORTS → `InternalServerErrorException`
- IMPORTS → `Logger`
- IMPORTS → `RequestMethod`
- IMPORTS → `StreamableFile`
- IMPORTS → `VERSION_NEUTRAL`
- IMPORTS → `VersioningOptions`
- IMPORTS → `VersioningType`
- IMPORTS → `VersionValue`
- IMPORTS → `CorsOptions`
- IMPORTS → `CorsOptionsDelegate`
- IMPORTS → `NestApplicationOptions`
- IMPORTS → `isFunction`
- IMPORTS → `isNil`
- IMPORTS → `isObject`
- IMPORTS → `isString`
- IMPORTS → `isUndefined`
- IMPORTS → `AbstractHttpAdapter`
- IMPORTS → `RouterMethodFactory`
- IMPORTS → `LegacyRouteConverter`