ExpressAdapter

August 17, 2026

ExpressAdapter

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`

Methods

Method	Signature	Returns
`setOnRequestHook`	`setOnRequestHook(onRequestHook: ( req: express.Request, res: express.Response, done: () => void, ) => Promise`	`void)`
`setOnResponseHook`	`setOnResponseHook(onResponseHook: ( req: express.Request, res: express.Response, ) => Promise`	`void)`
`reply`	`reply(response: any, body: any, statusCode: number)`	`void`
`status`	`status(response: any, statusCode: number)`	`void`
`end`	`end(response: any, message: string)`	`void`
`render`	`render(response: any, view: string, options: any)`	`void`
`redirect`	`redirect(response: any, statusCode: number, url: string)`	`void`

Method	Signature	Returns
`setErrorHandler`	`setErrorHandler(handler: Function, prefix: string)`	`void`
`setNotFoundHandler`	`setNotFoundHandler(handler: Function, prefix: string)`	`void`
`isHeadersSent`	`isHeadersSent(response: any)`	`boolean`
`getHeader`	`getHeader(response: any, name: string)`	`void`
`setHeader`	`setHeader(response: any, name: string, value: string)`	`void`
`appendHeader`	`appendHeader(response: any, name: string, value: string)`	`void`
`normalizePath`	`normalizePath(path: string)`	`string`
`listen`	`listen(port: string	number, callback: () => void)`	
`listen`	`listen(port: string	number, hostname: string, callback: () => void)`	
`listen`	`listen(port: any, args: any[])`	`Server`
`close`	`close()`	`void`
`set`	`set(args: any[])`	`void`
`enable`	`enable(args: any[])`	`void`
`disable`	`disable(args: any[])`	`void`
`engine`	`engine(args: any[])`	`void`
`useStaticAssets`	`useStaticAssets(path: string, options: ServeStaticOptions)`	`void`
`setBaseViewsDir`	`setBaseViewsDir(path: string | string[])`	
`setViewEngine`	`setViewEngine(engine: string)`	`void`
`getRequestHostname`	`getRequestHostname(request: any)`	`string`
`getRequestMethod`	`getRequestMethod(request: any)`	`string`
`getRequestUrl`	`getRequestUrl(request: any)`	`string`
`enableCors`	`enableCors(options: CorsOptions | CorsOptionsDelegate)`	

Method	Signature	Returns
`createMiddlewareFactory`	`createMiddlewareFactory(requestMethod: RequestMethod)`	`(path: string, callback: Function) => any`
`initHttpServer`	`initHttpServer(options: NestApplicationOptions)`	`void`
`registerParserMiddleware`	`registerParserMiddleware(prefix: string, rawBody: boolean)`	`void`
`useBodyParser`	`useBodyParser(type: NestExpressBodyParserType, rawBody: boolean, options: Omit<Options, 'verify'>)`	`this`
`setLocal`	`setLocal(key: string, value: any)`	`void`
`getType`	`getType()`	`string`
`applyVersionFilter`	`applyVersionFilter(handler: Function, version: VersionValue, versioningOptions: VersioningOptions)`	`VersionedRoute`

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…`.

When something fails

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

Diagram

```
graph LR
  Client[HTTP Client] --> Express[Express Application]
  Express --> Adapter[ExpressAdapter]
  Adapter --> RequestHook[Request Hook]
  Adapter --> Route[Route Handler]
  Route --> ResponseHelpers[reply / status / render / redirect / end]
  Adapter --> ResponseHook[Response Hook]
  Adapter --> ErrorHandler[Error Handler]
  Adapter --> NotFoundHandler[Not Found Handler]
```

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.status(res, 404);
  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);
```

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.

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]

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`