FastifyAdapter

August 17, 2026

FastifyAdapter

Kind: Class

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

Part of: Platform Fastify

`FastifyAdapter` bridges the framework's HTTP abstractions to a Fastify server instance. It initializes the Fastify application, registers request/response lifecycle hooks, exposes common HTTP route methods, and starts the server through overloaded `listen()` methods.

Extends: `AbstractHttpAdapter`

Methods

Method	Signature	Returns
`setOnRequestHook`	`setOnRequestHook(hook: ( request: TRequest, reply: TReply, done: (err?: Error) => void, ) => void`	Promise)`
`setOnResponseHook`	`setOnResponseHook(hook: ( request: TRequest, reply: TReply, done: (err?: Error) => void, ) => void`	Promise)`
`init`	`init()`	`void`
`listen`	`listen(port: string`	number, callback: () => void)`
`listen`	`listen(port: string`	number, hostname: string, callback: () => void)`
`listen`	`listen(listenOptions: string`	number
`get`	`get(args: any[])`	`void`
`post`	`post(args: any[])`	`void`

Method	Signature	Returns
`head`	`head(args: any[])`	`void`
`delete`	`delete(args: any[])`	`void`
`put`	`put(args: any[])`	`void`
`patch`	`patch(args: any[])`	`void`
`options`	`options(args: any[])`	`void`
`search`	`search(args: any[])`	`void`
`propfind`	`propfind(args: any[])`	`void`
`proppatch`	`proppatch(args: any[])`	`void`
`mkcol`	`mkcol(args: any[])`	`void`
`copy`	`copy(args: any[])`	`void`
`move`	`move(args: any[])`	`void`
`lock`	`lock(args: any[])`	`void`
`unlock`	`unlock(args: any[])`	`void`
`applyVersionFilter`	`applyVersionFilter(handler: Function, version: VersionValue, versioningOptions: VersioningOptions)`	`VersionedRoute<TRequest, TReply>`
`reply`	`reply(response: TRawResponse	TReply, body: any, statusCode: number)`
`status`	`status(response: TRawResponse	TReply, statusCode: number)`
`end`	`end(response: TReply, message: string)`	`void`
`render`	`render(response: TReply & { view: Function }, view: string, options: any)`	`void`
`redirect`	`redirect(response: TReply, statusCode: number, url: string)`	`void`
`setErrorHandler`	`setErrorHandler(handler: Parameters<TInstance['setErrorHandler']>[0])`	`void`
`setNotFoundHandler`	`setNotFoundHandler(handler: Function)`	`void`
`getHttpServer`	`getHttpServer()`	`T`
`getInstance`	`getInstance()`	`T`

Method	Signature	Returns
`register`	`register(plugin: TRegister['0'], opts: TRegister['1'])`	`void`
`inject`	`inject()`	`LightMyRequestChain`
`inject`	`inject(opts: InjectOptions`	string)`
`inject`	`inject(opts: InjectOptions`	string)`
`close`	`close()`	`void`
`initHttpServer`	`initHttpServer()`	`void`
`useStaticAssets`	`useStaticAssets(options: FastifyStaticOptions)`	`void`
`setViewEngine`	`setViewEngine(options: FastifyViewOptions`	string)`
`isHeadersSent`	`isHeadersSent(response: TReply)`	`boolean`
`getHeader`	`getHeader(response: any, name: string)`	`void`
`setHeader`	`setHeader(response: TReply, name: string, value: string)`	`void`
`appendHeader`	`appendHeader(response: any, name: string, value: string)`	`void`
`getRequestHostname`	`getRequestHostname(request: TRequest)`	`string`
`getRequestMethod`	`getRequestMethod(request: TRequest)`	`string`
`getRequestUrl`	`getRequestUrl(request: TRequest)`	`string`
`getRequestUrl`	`getRequestUrl(request: TRawRequest)`	`string`
`getRequestUrl`	`getRequestUrl(request: TRequest & TRawRequest)`	`string`
`enableCors`	`enableCors(options: FastifyCorsOptions)`	`void`
`registerParserMiddleware`	`registerParserMiddleware(prefix: string, rawBody: boolean)`	`void`

Properties

Property	Type
`logger`	`any`
`instance`	`TInstance`

Property	Type
`_pathPrefix`	`string`

Where it refuses work

- `FastifyAdapter` stops the work with `Error` when `!isString(value) && !Array.isArray(value)` — “Version constraint should be a string or an array of strings.”.
- `FastifyAdapter` stops the work with an early return when `Array.isArray(version)`.
- `FastifyAdapter` stops the work with an early return when `this.versioningOptions?.type === VersioningType.CUSTOM`.
- `FastifyAdapter` stops the work with an early return when `this.isMiddieRegistered`.
- `FastifyAdapter` stops the work with an early return when `err.code !== 'ERR_SERVER_NOT_RUNNING'`.
- `FastifyAdapter` stops the work with an early return when `this._isParserRegistered`.

When something fails

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

Diagram

```
graph LR
  App[Application] --> Adapter[FastifyAdapter]
  Adapter --> Hooks[Request / Response Hooks]
  Adapter --> Routes[GET / POST / HEAD / DELETE Routes]
  Adapter --> Fastify[Fastify Server]
  Fastify --> Client[HTTP Clients]
```

Usage

```
import { FastifyAdapter } from '@platform-fastify/adapters/fastify-adapter';

const adapter = new FastifyAdapter();

adapter.setOnRequestHook(async (request, reply) => {
  request.log.info({ url: request.url }, 'Incoming request');

  // Example authentication guard
  if (request.headers['x-api-key'] !== process.env.API_KEY) {
    return reply.code(401).send({ message: 'Unauthorized' });
```

```
  }
});

adapter.setOnResponseHook(async (request, reply) => {
  request.log.info(
    { url: request.url, statusCode: reply.statusCode },
    'Response sent',
  );
});

adapter.get('/health', async () => {
  return { status: 'ok' };
});

adapter.post('/users', async (request, reply) => {
  const user = request.body;

  reply.code(201);
  return { user };
});

await adapter.init();
adapter.listen(3000);
```

AI Coding Instructions

- Register lifecycle hooks with `setOnRequestHook()` and `setOnResponseHook()` before calling `init()` or `listen()`.
- Use the adapter's route helpers (`get`, `post`, `head`, and `delete`) instead of registering routes directly on the underlying Fastify instance when working within the abstraction.
- Ensure handlers return serializable response values or explicitly send a response through Fastify's reply object.
- Call `init()` before starting the server when initialization is not handled automatically by the surrounding application bootstrap.
- Preserve Fastify request/reply semantics in hooks and route handlers, including returning early after sending an error response.

Relationships

- IMPORTS → `HttpStatus`
- IMPORTS → `Logger`
- IMPORTS → `RawBodyRequest`
- IMPORTS → `RequestMethod`

- IMPORTS → `StreamableFile`
- IMPORTS → `VERSION_NEUTRAL`
- IMPORTS → `VersioningOptions`
- IMPORTS → `VersioningType`
- IMPORTS → `VersionValue`
- IMPORTS → `loadPackage`
- IMPORTS → `isString`
- IMPORTS → `isUndefined`
- IMPORTS → `AbstractHttpAdapter`
- IMPORTS → `LegacyRouteConverter`