timeout

August 17, 2026

timeout

Kind: Function

Source: `src/middleware/timeout/index.ts`

Part of: Middleware

Timeout Middleware for Hono.

`timeout` creates Hono middleware that limits how long downstream request handling may run. Apply it to an app or route path to return a timeout response when the configured duration expires before the next handler finishes.

Signature

```
function timeout(duration: number, exception: HTTPExceptionFunction | HTTPException): Middlewar
```

Parameters

Name	Type
`duration`	`number`
`exception`	`HTTPExceptionFunction

Returns: `MiddlewareHandler`

Diagram

```
graph LR
  Request[Incoming request] --> Middleware[timeout middleware]
  Middleware --> Next[Downstream middleware or handler]
  Middleware --> Timer[Timeout timer]
  Next --> Race[Wait for completion]
  Timer --> Race
```

```
  Race -->|Handler completes| Response[Normal response]
  Race -->|Duration expires| TimeoutResponse[Timeout response]
```

Usage

```ts
import { Hono } from 'hono'
import { timeout } from 'hono/timeout'

const app = new Hono()

const requestTimeout = Number(process.env.REQUEST_TIMEOUT_MS)

app.use('/api/*', timeout(requestTimeout))

app.get('/api/data', async (c) => {
  const response = await fetch('https://example.com/data')
  return c.json(await response.json())
})

export default app
```

AI Coding Instructions

- Apply `timeout` before the middleware and handlers whose execution time it should limit.
- Pass a duration in milliseconds from configuration rather than hard-coding route-specific values.
- Keep downstream handlers compatible with early timeout responses; a timed-out client request does not stop external work already started by the handler.
- Scope the middleware with a route pattern when only selected endpoints need timeout handling.