timing

August 17, 2026

timing

Kind: Function

Source: `src/middleware/timing/timing.ts`

Part of: Middleware

Server-Timing Middleware for Hono.

`timing` creates Hono middleware that measures request timing and writes timing data to the `Server-Timing` response header. It also makes a metrics collector available through the request context so handlers can record named operations.

Signature

```
function timing(config: TimingOptions): MiddlewareHandler
```

Parameters

Name	Type
`config`	`TimingOptions`

Returns: `MiddlewareHandler`

Diagram

```
graph LR
  Request --> Timing["timing() middleware"]
  Timing --> Metrics["Context metrics collector"]
  Metrics --> Handler["Route handler"]
  Handler --> Response["Response"]
  Timing --> Header["Server-Timing header"]
  Header --> Response
```

Usage

```
import { Hono } from 'hono'
import { timing } from 'hono/timing'

const app = new Hono()

app.use('*', timing())

app.get('/reports', async (c) => {
  const metrics = c.get('metrics')

  metrics.start('database')
  const report = await loadReport()
  metrics.end('database')

  return c.json(report)
})

async function loadReport() {
  return { status: 'ok' }
}

export default app
```

AI Coding Instructions

- Register `timing()` before route handlers so it can measure the full request lifecycle.
- Access custom timing metrics with `c.get('metrics')` inside handlers or downstream middleware.
- Call `metrics.start(name)` and `metrics.end(name)` with the same metric name.
- Preserve `Server-Timing` headers when adding response headers manually.
- Enable cross-origin timing settings only when browser clients need access to timing data.