SmartRouter

August 17, 2026

SmartRouter

Kind: Class

Source: `src/router/smart-router/router.ts`

Part of: Router

`SmartRouter` registers routes with `add()` and resolves incoming paths with `match()`, which returns a `Result<T>`. It acts as the routing layer between route definitions and code that needs to select a matching handler or value.

Implements: `Router`

Methods

Method	Signature	Returns
`add`	`add(method: string, path: string, handler: T)`	`void`
`match`	`match(method: string, path: string)`	`Result<T>`

Properties

Property	Type
`name`	`string`
`#routers`	`Router<T>[]`
`#routes`	`[string, string, T][]`

Where it refuses work

- `SmartRouter` stops the work with `Error` when `!this.#routes`.
- `SmartRouter` stops the work with `Error` when `!this.#routes` — “Fatal error”.
- `SmartRouter` stops the work with `Error` when `i === len` — “Fatal error”.

- `SmartRouter` stops the work with `Error` when `this.#routes || this.#routers.length !== 1` — “No active router has been determined yet.”.

When something fails

- `SmartRouter` handles failure in 1 place: it lets it reach the caller in all 1.

Diagram

```
graph LR
  Routes[Route definitions] -->|add()| Router[SmartRouter]
  Request[Incoming path] -->|match()| Router
  Router --> Result[Result&amp;lt;T&amp;gt;]
```

Usage

```
import { SmartRouter } from './src/router/smart-router/router';

const router = new SmartRouter();

router.add('/users/:id', {
  handler: 'getUser',
});

const result = router.match('/users/ada');

// Handle the returned Result<T> according to its success or failure shape.
console.log(result);
```

AI Coding Instructions

- Register route patterns through `add()` before attempting to resolve paths with `match()`.
- Keep values passed to `add()` consistent so callers can handle the `Result<T>` returned by `match()`.
- Preserve route-pattern parsing and matching behavior when changing router internals.
- Handle both match and non-match outcomes from `match()` rather than assuming every path resolves.

How it works

`SmartRouter<T>` is a `Router<T>` implementation that receives an ordered array of candidate routers and initially stores added routes locally rather than adding them to those candidates immediately. Its initial `name` is `"SmartRouter"`. [src/router/smart-router/router.ts:4-10]

A `Router<T>` has a mutable `name`, an `add(method, path, handler)` method, and a `match(method, path)` method returning `Result<T>`. [src/router.ts:29-52]

Construction and route registration

- The constructor requires `{ routers: Router<T>[] }` and stores that array as its candidate-router list. It does not validate that the list is non-empty. [src/router/smart-router/router.ts:9-11]
- Before router selection, `add()` appends `[method, path, handler]` tuples to its private route list. [src/router/smart-router/router.ts:13-19]
- After selection, the private route list is set to `undefined`; any later `add()` call throws `Error` with `MESSAGE_MATCHER_IS_ALREADY_BUILT`, whose text is `"Can not add a route since the matcher is already built."`. [src/router/smart-router/router.ts:14-16] [src/router/smart-router/router.ts:46-48] [src/router.ts:19-22]

First match and candidate selection

The first call to `match(method, path)` selects one candidate router:

1. It iterates through candidate routers in their configured order. [src/router/smart-router/router.ts:26-35]
2. For each candidate, it calls that candidate's `add()` once for every locally stored route, then calls the candidate's `match(method, path)`. [src/router/smart-router/router.ts:35-38]
3. If this work throws `UnsupportedPathError`, it skips that candidate and tries the next one. Any other thrown value is rethrown. [src/router/smart-router/router.ts:39-44]
4. The first candidate that completes these operations is selected, even if its match result contains no handlers; selection is based on the absence of an `UnsupportedPathError`, not on match-result contents. [src/router/smart-router/router.ts:38-49]
5. It replaces its own `match` method with the selected router's bound `match` method, retains only that router, clears the buffered routes, changes `name` to `"SmartRouter + <selected router name>"`, and returns the first match result. [src/router/smart-router/router.ts:46-60]

For example, the default `Hono` constructor creates a `SmartRouter` with `RegExpRouter` first and `TrieRouter` second, unless `options.router` is supplied. [src/hono.ts:26-33]

`RegExpRouter` can throw `UnsupportedPathError` when insertion encounters its internal `PATH_ERROR`, which is the error type `SmartRouter` catches for candidate fallback. [src/router/reg-exp-router/router.ts:59-64] [src/router/smart-router/router.ts:39-42]

Errors and state restrictions

- Calling the original `match()` after route buffering has already been cleared would throw `Error('Fatal error')`; under normal selection, that method is replaced before subsequent calls. [src/router/smart-router/router.ts:21-24] [src/router/smart-router/router.ts:46-48]
- If every candidate throws `UnsupportedPathError`, `match()` throws `Error('Fatal error')`. [src/router/smart-router/router.ts:40-42] [src/router/smart-router/router.ts:52-55]
- If a candidate throws anything other than `UnsupportedPathError` while routes are being added or while matching, that value is rethrown. [src/router/smart-router/router.ts:34-44]
- Candidate routers that fail with `UnsupportedPathError` may already have received some buffered routes, because routes are added one at a time before the error is caught; `SmartRouter` does not reset such a candidate. [src/router/smart-router/router.ts:35-42]

Active router

`activeRouter` returns the selected router only after selection has reduced the candidate list to one router and cleared the buffered route list. Otherwise it throws `Error('No active router has been determined yet.')`. [src/router/smart-router/router.ts:63-69]

Application lifecycle

In `HonoBase`, route registration calls `this.router.add(method, path, [handler, routeMetadata])`, and dispatch calls `this.router.match(method, path)`. Thus, for the default router, route buffering ends when dispatch first performs a match. [src/hono-base.ts:386-397] [src/hono-base.ts:419-428]

Relationships

- IMPORTS → `MESSAGE_MATCHER_IS_ALREADY_BUILT`
- IMPORTS → `UnsupportedPathError`