PreparedRegExpRouter

August 18, 2026

PreparedRegExpRouter

Kind: Class

Source: `src/router/reg-exp-router/prepared-router.ts`

Part of: Router

`PreparedRegExpRouter<T>` collects route definitions and prepares regular-expression matchers for the routing layer. It separates path and wildcard handling while producing a `MatcherMap<T>` through `buildAllMatchers()`.

Implements: `Router`

Methods

Method	Signature	Returns
`#addWildcard`	`#addWildcard(method: string, handlerData: [T, ParamIndexMap])`	`void`
`#addPath`	`#addPath(method: string, path: string, handler: T, indexes: (number	string)[], map: ParamIndexMap
`add`	`add(method: string, path: string, handler: T)`	`void`
`buildAllMatchers`	`buildAllMatchers()`	`MatcherMap<T>`

Properties

Property	Type
`name`	`string`
`#matchers`	`MatcherMap<T>`
`#relocateMap`	`RelocateMap`

Property	Type
`match`	`typeof match<Router<T>, T>`

Where it refuses work

- `PreparedRegExpRouter` stops the work with `Error` when `!data`.

Diagram

```
graph LR
  A[add method path handler] --> B{Route pattern}
  B -->|Path| C[#addPath]
  B -->|Wildcard| D[#addWildcard]
  C --> E[Prepared route data]
  D --> E
  E --> F[buildAllMatchers]
  F --> G[MatcherMap<T>]
```

Usage

```
import { PreparedRegExpRouter } from './src/router/reg-exp-router/prepared-router'

type Handler = (request: Request) => Response

const router = new PreparedRegExpRouter<Handler>()

router.add('GET', '/articles/:slug', (request) => {
  return new Response(`Article request: ${request.url}`)
})

router.add('GET', '/assets/*', () => {
  return new Response('Asset request')
})

const matchers = router.buildAllMatchers()

// Pass `matchers` to the router component that resolves requests.
```

AI Coding Instructions

- Register routes through `add()`; keep `#addPath()` and `#addWildcard()` as internal preparation details.

- Preserve the generic `T` consistently between registered handlers and the resulting `MatcherMap<T>`.
- Handle wildcard paths through the existing wildcard flow rather than adding separate matcher logic at call sites.
- Build matchers after route registration and pass the resulting map to the request-matching layer.

How it works

- `PreparedRegExpRouter<T>` is a `Router<T>` implementation whose `name` is `"PreparedRegExpRouter"`. Its constructor accepts a prebuilt `MatcherMap<T>` and a relocation map, then stores both in private fields. [src/router/reg-exp-router/prepared-router.ts:9-17]
- It expects its initialization data to describe the paths that may later be registered. `buildInitParams({ paths })` builds that data by adding every path to a temporary `RegExpRouter` under the `ALL` method, exporting its compiled matchers, recording where each path's handlers and parameter-index map belong, and clearing those handler slots before returning `[matchers, relocateMap]`. [src/router/reg-exp-router/prepared-router.ts:95-153]
- `add(method, path, handler)` registers a handler into the already prepared matcher data. If the specified method has no matcher entry, it creates one by copying the `ALL` matcher's regular expression, handler lists, and static-route handler lists. [src/router/reg-exp-router/prepared-router.ts:47-59]
- For `'*'` and `'/*'`, `add` creates `[handler, {}]`; for a specific method it appends that pair to every existing dynamic and static handler list for that method. For `ALL`, it does the same for every matcher currently present. [src/router/reg-exp-router/prepared-router.ts:19-23] [src/router/reg-exp-router/prepared-router.ts:61-70]
- For every non-wildcard path, `add` requires an entry in the relocation map. If none exists, it throws `Error("Path <path> is not registered")`. [src/router/reg-exp-router/prepared-router.ts:73-76] The relocation data determines whether the handler is appended to a static-path list or one or more dynamic handler lists, and associates the handler with the recorded `ParamIndexMap`. [src/router/reg-exp-router/prepared-router.ts:25-45] With method `ALL`, this insertion is repeated for each current matcher; otherwise it is done only for the selected method. [src/router/reg-exp-router/prepared-router.ts:77-85]

- `match(method, path)` is the shared regular-expression matcher. It selects the matcher for `method`, falling back to the `ALL` matcher; it first checks an exact static-path entry, then tests the path against the matcher regular expression. [src/router/reg-exp-router/matcher.ts:10-24] A non-match returns `[[], emptyParam]`; a dynamic match selects a handler list from the first empty capture after index 0 and returns that list with the regular-expression match array. [src/router/reg-exp-router/matcher.ts:22-29] On its first call, this function replaces the instance's `match` property with the bound matching closure. [src/router/reg-exp-router/matcher.ts:10-12] [src/router/reg-exp-router/matcher.ts:31-32] `PreparedRegExpRouter` assigns this function as its `match` member. [src/router/reg-exp-router/prepared-router.ts:92]
- A match result contains handler and parameter-index-map pairs plus a parameter stash, as represented by `Result<T>`; static routes use an empty parameter stash. [src/router.ts:67-98] [src/router/reg-exp-router/matcher.ts:17-20] Parameter-index maps are records from parameter names to numeric capture positions. [src/router.ts:54-58]
- `buildInitParams` omits wildcard paths from the relocation map, because wildcard registration is handled directly by `add`. [src/router/reg-exp-router/prepared-router.ts:61-70] [src/router/reg-exp-router/prepared-router.ts:111-115] It also merges parameter maps and records distinct dynamic or static handler locations for each non-wildcard path. [src/router/reg-exp-router/prepared-router.ts:116-143]
- `serializeInitParams` converts constructor parameters to a JavaScript array-expression string. It serializes `RegExp` values through `toString()`, removes the marker quoting, adjusts doubled backslashes, and appends JSON for the relocation map. [src/router/reg-exp-router/prepared-router.ts:156-165]

Relationships

- IMPORTS → `METHOD_NAME_ALL`
- IMPORTS → `match`
- IMPORTS → `emptyParam`
- IMPORTS → `RegExpRouter`