Trie

August 17, 2026

Trie

Kind: Class

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

Part of: Router

`Trie` stores route definitions in a prefix tree for the regular-expression router. It accepts routes through `insert()` and compiles the stored tree into a `RegExp` with replacement maps through `buildRegExp()`.

Methods

Method	Signature	Returns
`insert`	`insert(path: string, isStatic: boolean)`	`void`
`buildRegExp`	`buildRegExp()`	`[RegExp, ReplacementMap, ReplacementMap]`

Properties

Property	Type
`#context`	`Context`
`#root`	`Node`
`#index`	`number`
`paths`	`Record<string, [number, ParamAssocArray]>`

Where it refuses work

- `Trie` stops the work with an early return when `regexp === ''`.

Diagram

```
graph LR
  Route["Route definition"] --> Insert["Trie.insert()"]
  Insert --> Tree["Trie node tree"]
  Tree --> Build["Trie.buildRegExp()"]
  Build --> Pattern["RegExp"]
  Build --> Params["Parameter replacement map"]
  Build --> Routes["Route replacement map"]
```

Usage

```
import { Trie } from './src/router/reg-exp-router/trie'

const trie = new Trie()

trie.insert('GET', '/users/:id', (c) => {
  return c.text(`User: ${c.req.param('id')}`)
})

trie.insert('GET', '/health', (c) => {
  return c.text('ok')
})

const [pattern, paramReplacements, routeReplacements] = trie.buildRegExp()

const match = pattern.exec('/users/42')

if (match) {
  // Router code reads captures through the replacement maps
  console.log(match)
  console.log(paramReplacements)
  console.log(routeReplacements)
}
```

AI Coding Instructions

- Add routes through `insert()` before calling `buildRegExp()` so the compiled expression includes every route.
- Keep the `RegExp` and both replacement maps together; capture groups are interpreted through those maps.
- Preserve route parameter syntax when inserting paths, since parameter names are resolved during expression construction.

- Treat `Trie` as router infrastructure; route dispatch code should consume the output of `buildRegExp()` rather than inspect trie nodes directly.

How it works

`Trie` is an internal route-pattern tree used by `RegExpRouter` to collect dynamic paths and compile them into one regular expression plus capture-index lookup tables. It owns a shared `Node` root, a variable-index context, and a sequential dynamic-handler index. [src/router/reg-exp-router/trie.ts:6-11]

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `RegExpRouter` — `src/router/reg-exp-router/router.ts`:47