RouteTree

August 18, 2026

RouteTree

Kind: Interface

Source: `packages/core/router/interfaces/routes.interface.ts`

Part of: Core

`RouteTree` defines a hierarchical route configuration for the router. Each node maps a URL `path` to a module and can contain nested route nodes or module types as `children`, allowing the application to compose route trees declaratively.

Properties

Property	Type
`path`	`string`
`module`	`Type<any>`
`children`	`(RouteTree

Diagram

```
graph LR
  Root["RouteTree"]
  Path["path: string"]
  Module["module: Type&lt;any&gt;"]
  Children["children: (RouteTree | Type&lt;any&gt;)[]"]

  Root --> Path
  Root --> Module
  Root --> Children

  Children --> NestedRoute["Nested RouteTree"]
  Children --> ChildModule["Child Module Type"]
```

Usage

```
import type { Type } from '@blaze/core';
import type { RouteTree } from '@blaze/core/router';

class AppModule {}
class UsersModule {}
class UserProfileModule {}

const routes: RouteTree = {
  path: '/',
  module: AppModule,
  children: [
    {
      path: '/users',
      module: UsersModule,
      children: [UserProfileModule],
    },
  ],
};
```

AI Coding Instructions

- Define `path` values as router-recognized URL segments and keep nested paths consistent with the parent route structure.
- Set `module` to the module responsible for handling the route; it must be a runtime class/type, not an instance.
- Use nested `RouteTree` objects when child routes need their own path and children configuration.
- Use `Type<any>` entries directly in `children` only for child modules that do not require explicit route metadata.
- Avoid placing arbitrary values in `children`; every entry must be either a valid `RouteTree` or a module type.

How it works

Structure

`RouteTree` is an exported TypeScript interface for one node in a module-route hierarchy. It has a required `path: string`, optional `module?: Type<any>`, and optional `children?: (RouteTree | Type<any>)[]`; `Routes` is an array of these nodes. [packages/core/router/interfaces/routes.interface.ts:3-9]

`Type<T>` is a constructible function type with a `new (...args: any[])` signature, so `module` and direct `children` entries are class/constructor references at the type level. [packages/common/interfaces/type.interface.ts:1-3]

Route-tree behavior

A node with both a truthy `module` and a truthy `path` becomes a flattened `{ module, path }` route entry. [packages/core/router/utils/flatten-route-paths.util.ts:10-13]

A node may omit `module` and act as a path grouping node for `children`; the flattener still descends into its children. [packages/core/router/utils/flatten-route-paths.util.ts:11-14] Nested object children whose `path` is a string have their paths rewritten to the normalized concatenation of the parent path and child path before recursive processing. [packages/core/router/utils/flatten-route-paths.util.ts:16-20,25] Tests show this produces accumulated paths such as `/parent/child/child2` for nested route objects. [packages/core/test/router/utils/flat-routes.spec.ts:44-68,104-116]

A direct constructor in `children` is flattened as a module at its parent node's path. [packages/core/router/utils/flatten-route-paths.util.ts:16-23] This shape is exercised with `children: [AuthModule, CatsModule, DogsModule]`, producing one route entry per module at `/v1`. [packages/core/test/router/utils/flat-routes.spec.ts:100-102,122-128]

Path normalization adds a leading slash, removes trailing slashes, collapses repeated slashes, and maps an absent or empty path to `/`. [packages/common/utils/shared.utils.ts:33-38] For example, a route configured as `path: '/module-path/'` is tested at `/module-path/test`. [integration/hello-world/e2e/router-module-middleware.spec.ts:49-54,62-65]

Consumption by `RouterModule`

`RouterModule.register()` accepts `Routes` and places that array in a provider under the `ROUTES` symbol. [packages/core/router/router-module.ts:9,29-38] On construction, `RouterModule` recursively clones route objects while retaining direct constructor entries, then initializes from the clone. [packages/core/router/router-module.ts:21-27,41-56]

Initialization flattens the cloned tree, normalizes each resulting path, stores it as reflection metadata on the target module under a key derived from `MODULE_PATH` and the container application ID, and records matching module references in a `WeakSet` associated with that container. [packages/core/router/router-module.ts:58-75,78-93] If a flattened module constructor is not found in the container, the cache update returns without adding it. [packages/core/router/router-module.ts:86-92]

The interface and registration method contain no explicit runtime validation or explicit error handling for route fields. [packages/core/router/interfaces/routes.interface.ts:3-7] During flattening, an object child without a string `path` is treated as a module constructor rather than as a nested route object. [packages/core/router/utils/flatten-route-paths.util.ts:16-23]

Relationships

- IMPORTS → `Type`