Angular 4 - Could not resolve submodule for routing

Asked 1 year, 10 months ago Active 3 months ago Viewed 34k times

29

3

I'm building a webapp with Angular 4. I have a top-level routing module and a separate routing module for each submodule (e.g. HomeModule).

This is my top-level routing configuration:

```
export const ROUTES: Routes = [
  {path: '', loadChildren: './home#HomeModule'},
  {path: '**', component: NotFoundComponent},
];
```

When I run `ng server`, I get a strange error, that module `home` was not found. The app does not work in the browser.

The strange part is the following: When a file is changed and webpack recompiles the project, everything works just fine and the routing works.
The error does only appear when I'm running `ng serve`.

This is the error I get when I'm running `ng serve`, not when the project is recompiled because of a file change:

```
ERROR in Error: Could not resolve module ./home relative to
/path/to/my/project/src/app/app.module.ts
    at StaticSymbolResolver.getSymbolByModule
(/path/to/my/project/node_modules/@angular/compiler/bundles/compiler.umd.js:31884:30)
    at StaticReflector.resolveExternalReference
(/path/to/my/project/node_modules/@angular/compiler/bundles/compiler.umd.js:30350:62)
    at parseLazyRoute
(/path/to/my/project/node_modules/@angular/compiler/bundles/compiler.umd.js:28616:55)
    at listLazyRoutes
(/path/to/my/project/node_modules/@angular/compiler/bundles/compiler.umd.js:28578:36)
    at visitLazyRoute
(/path/to/my/project/node_modules/@angular/compiler/bundles/compiler.umd.js:29995:47)
    at AotCompiler.listLazyRoutes
(/path/to/my/project/node_modules/@angular/compiler/bundles/compiler.umd.js:29963:20)
    at AngularCompilerProgram.listLazyRoutes
(/path/to/my/project/node_modules/@angular/compiler-
```

```
    at AngularCompilerPlugin._getLazyRoutesFromNgtools
(/path/to/my/project/node_modules/@ngtools/webpack/src/angular_compiler_plugin.js:247:66)

    at Promise.resolve.then.then
(/path/to/my/project/node_modules/@ngtools/webpack/src/angular_compiler_plugin.js:538:50)

    at <anonymous>
    at process._tickCallback (internal/process/next_tick.js:188:7)
```

Thanks in advance.

angular angular-ui-router

edited May 3 at 14:14
Kishan Patel
568 4 16

asked Dec 1 '17 at 21:19
Martin Fink
471 1 10 17

Lazy Loading route configuration: *The address is the AdminModule file location (**relative to the app root**), followed by a # separator, followed by the name of the exported module class, AdminModule.* – Kirk Larkin Dec 1 '17 at 21:58

7 Answers

78

I was using the absolute path convention: `app/home#HomeModule` and it **wasn't** working.

I then tried the **relative** path convention: `./home#HomeModule` and it worked.

> ... in the CLI, the paths to lazy routes should be relative from the file you're in

Source

I followed this Tutorial and it used the absolute path convention and it worked.

UPDATE:

As Friedrich mentioned, to make it work using an Absolute Path, update `src/tsconfig.app.json` as follows:

```
{
  ...,
  "compilerOptions": {
    ...,
    baseUrl: "./"
  }
}
```

edited Sep 28 '18 at 22:00

answered Dec 29 '17 at 22:12

spottedmahn
6,040 3 44 81

2 I append ./ to my module name and it works for me thanks – SURENDRANATH SONAWANE Apr 28 '18 at 20:59

15 @spottedmahn: The path `app/home#HomeModule` **is** working, but you have to add `baseUrl: "./"` in the `compilerOptions` in `tsconfig.app.json` (see stackoverflow.com/a/50606826/5816097). – Friedrich May 30 '18 at 14:12

1 Friedrich had provided the correct solution. – ZZZ Jul 8 '18 at 12:16

Thanks, my problem was needing to omit the `*.ts` extension – dook Feb 12 at 19:48

13

I had the same issue and none of the answers above worked for me,the final solution i found was using the absolute path of the module and module class name after the '#'.

```
export const ROUTES: Routes = [
   {path: '', loadChildren: 'src/app/pathToYourModule/home.module#HomeModule'},
   {path: '**', component: NotFoundComponent}
 ];
```

I started the path from 'src' and dont forget to remove the '.ts' from the module path.

answered Sep 2 '18 at 16:40

1 Yes thank you! only thing that fixed it – Smokey Dawson Jan 9 at 2:14

Thank you for the last statement! Apparently, had forgot to remove .ts from the path! – Nishkarsh Jun 14 at 11:51

8

You have to remove the .ts suffix from the reference to your module file: {path: '', loadChildren: 'app/pathToYourModule/home.module#HomeModule'},

answered Apr 24 '18 at 12:27

Michael Benoit

86 1 3

4

I have faced a similar issue and resolved by **removing the extension** from the module name and adding a **relative path**.

Code which was throwing an error:

```
const routes: Routes = [
    {
        path: 'settings',
        loadChildren: 'app/setting/setting.module.ts#SettingsModule'
    }
];
```

Working code:

```
const routes: Routes = [
    {
        path: 'settings',
        loadChildren: './setting/setting.module#SettingsModule'
    }
];
```

Hope it helps!

Note: On Angular version - 7.0.0

Vikash Kumar Choudhary
191 1 4

2

Try using the absolute path of your module with the module file name like so:

```
export const ROUTES: Routes = [
  {path: '', loadChildren: 'app/pathToYourModule/home.module.ts#HomeModule'},
  {path: '**', component: NotFoundComponent},
];
```

answered Dec 1 '17 at 21:50

Carlos Rincones
192 3 12

1

For those who are not able to find a solution, simply remove ".ts" extension from the loadChildren line.

```
 {path: '', loadChildren: './home.ts#HomeModule'},
```

to

```
{path: '', loadChildren: './home#HomeModule'},
```

answered Jul 4 at 11:43

Sagar Khatri
190 2 8

0

I came across this issue, for me I simply forgot to add `.module` to the file name.

I had this:

```
  {
    path: 'page-not-found',
    loadChildren: './modules/page-not-found/page-not-found#PageNotFoundModule' // <-
page-not-found missing .module
  },
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: '**',
    redirectTo: 'page-not-found'
  }
];
```

Should be:

```
const routes: Routes = [
  /* ... */
  {
    path: 'page-not-found',
    loadChildren: './modules/page-not-found/page-not-found.module#PageNotFoundModule'
  },
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: '**',
    redirectTo: 'page-not-found'
  }
];
```

answered Jun 24 at 19:12

Wayne

66 1 7