Angular Routing between modules

5 Comments / 7 minutes of reading / March 9, 2023

← Angular Modules | Angular Tutorial | Angular Folder Structure →

In this tutorial, we will learn how to setup routing between multiple feature modules. In the previous tutorial on Angular Modules, we learnt how to create the multiple feature modules in a application. The Modules are core of any Angular apps. Our App will going to contain several such modules each implementing a specific feature.

If you are new to routing, then suggest you to read the following articles

- Routing and Navigation in Angular
- Location Strategies in Angular Router
- Passing Parameters to Route
- Child Routes / Nested Routes
- Passing Optional (Query) Parameters to a route
- Navigation between Routes
- Angular Route Guards

Table of Contents

Routing : A recap

Let us briefly how routing is configured in the root module of the application.

Defining the routes

The routes are defined in a constant as shown below

```
1
2 const appRoutes={ path: 'home',   component: HomeComponent }
3
```

Where `path` is the URL segment and `component` is the component to be loaded. This route tells angular to render `HomeComponent` when the user navigates to the URL "/home"

How to Configure the Angular Routes

Register the Routes

Next, we register the routes with the `RouterModule` in our `AppModule` as shown below

```
imports: [RouterModule.forRoot(routes)],
```

Display the Component

Next, we will use the `RouterLink` directive to bind the click event to `Route`

```
<li><a [routerLink]="['home']">Home</a></li>
```

We, then display the component using the `router-outlet` directive as shown below.

```
<router-outlet></router-outlet>
```

Routing in Feature Module

The Routing in Feature Module or sub module follows the same pattern except for how we register the routes with the `RouterModule`

In the Root module we will use the `forRoot` (`RouterModule.forRoot(routes)`) method , while in feature modules we will use the `forChild` method (

RouterModule.forChild(routes))

forRoot vs forChild

The RouterModule contans several components. it also includes the several Services.

The services provided in the Root Module or in any of the *eagerly loaded feature modules* are app-scoped. i.e they are available for injection in every component in the app.

This rule does not apply to the *lazy loaded modules*. The lazy loaded modules gets their own injector and providers. The services provided in the lazy loaded modules are available only in the lazy loaded module only.

Now, consider the case where `RouterModule` is imported in a lazy loaded module. This will create the separate instance of the Router service in the lazy loaded module. This will have untended bugs as there should only a single instance of the Router service in the app.

We need to register the services only in the `AppModule` and not in the *Lazy loaded module*. The `forRoot` method imports `RouterModule` and registers all its services. Hence it is used in the *root module*. The `forChild` method imports `RouterModule` but does not registers its services. Hence it should be all other modules.

Example App

Create an Angular App using Angular CLI command.

```
1
2 ng new --routing  --style css ModuleDemo
```

```
3
```

Run the app to verify everything is ok

Creating the Feature Module

Lets create a feature module named `AdminModule`. First, Create a folder called `admin` under `app` folder.

user.component.ts

```
1
2  import { Component } from '@angular/core';
3
4  @Component({
5    template: '<h1>User Component</h1>',
6  })
7  export class UserComponent {
8    title = '';
9  }
10
```

rights.component.ts

```
1
2  import { Component } from '@angular/core';
```

```
3
4  @Component({
5    template: '<h1>Rights Component</h1>',
6  })
7  export class RightsComponent {
8    title = '';
9  }
10
```

dashboard.component.ts

```
1
2  import { Component } from '@angular/core';
3
4  @Component({
5    template: '<h1>Dashboard Component</h1>',
6  })
7  export class DashboardComponent {
8    title = '';
9  }
10
```

Routing Module

The next step is to create the routing module for the above components.

```
1
2  import { NgModule } from '@angular/core';
3  import { Routes, RouterModule } from '@angular/router';
4
5  import { UserComponent } from './user.component';
6  import { RightsComponent } from './rights.component';
7  import { DashboardComponent } from './dashboard.component';
8
9  const routes: Routes = [
10     {   path: 'user',   component: UserComponent   },
11     {   path: 'rights',   component: RightsComponent   },
12     {   path: 'dashboard',   component: DashboardComponent   },
13   ];
14
15 @NgModule({
16   imports: [RouterModule.forChild(routes)],
17   exports: [RouterModule]
```

```
})
export class AdminRoutingModule { }

```

The routes are defined as

```

const routes: Routes = [
   {  path: 'user',  component: UserComponent  },
   {  path: 'rights',  component: RightsComponent  },
   {  path: 'dashboard',  component: DashboardComponent  },
 ];

```

and `RouterModule` is imported with the `RouterModule.forChild(routes)`, which registers the routes with the router but does not create the router service

Root Module

admin.module.ts

```

import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

import { AdminRoutingModule } from './admin.routing.module';

import { UserComponent } from './user.component';
```

```
import { RightsComponent } from './rights.component';
import { DashboardComponent } from './dashboard.component';

@NgModule({
  declarations: [UserComponent,RightsComponent,DashboardComponent],
  imports: [
    CommonModule,
    AdminRoutingModule,
  ],
  providers: [],
})
export class AdminModule { }

```

app.module.ts

Finally, we need to import the `AdminModule` in the `AppModule`

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { AdminModule} from './admin/admin.module';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    AdminModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

app-routing.module.ts

The AppRoutingModule contains the routing information for the AppModule , which in the example is empty. The Routs are registered with the forRoot method, which registers the routes and also registers the routing related services.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Finally, the AppRoutingModule is imported in the AppModule

Add the following CSS styles to app.component.css

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333333;
}

li {  float: left; }

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 16px;
    text-decoration: none;
}

li a:hover { background-color: #111111; }
```

```
21
```

Child Routes

In the above example , the URL looks like /user , /dashboard & /rights . You can change the URL to admin/dashboard , admin/user , admin/rights by using the following routes

```
1
2  const routes: Routes = [
3     {   path: 'admin',   component: DashboardComponent,
4        children :[
5           { path: 'dashboard', component: DashboardComponent},
6           { path: 'user', component: UserComponent},
7           { path: 'rights', component: RightsComponent},
8        ]
9     },
10 ];
11
```

In the above example both /admin & /admin/dashboard points to the DashboardComponent

You need to make appropriate changes in the app.component.html

```
1
2  <ul>
3    <li>
```

```
    <a routerLink="/admin/dashboard">Dashboard</a>
  </li>
  <li>
     <a routerLink="/admin/user">User</a>
  </li>
  <li>
    <a routerLink="/admin/rights">Rights</a>
  </li>
</ul>

<router-outlet></router-outlet>

```

Menu in Feature Module

Change the routes to

```

const routes: Routes = [
   {  path: 'admin', component: DashboardComponent,
      children :[
         { path: 'user', component: UserComponent},
         { path: 'rights', component: RightsComponent},
      ]
   },
];

```

Add the Menu in DashboardComponent

```

import { Component } from '@angular/core';

@Component({
 template: `<h1>Dashboard Component</h1>

 <ul>
   <li><a routerLink="user">User</a></li>
   <li> <a routerLink="rights">Rights</a></li>
 </ul>
 <router-outlet></router-outlet>
 `,
})
```

```
export class DashboardComponent {
  title = '';
}

```

And remove it from app.component.html

```

<ul>
  <li>
    <a routerLink="/admin">Admin</a>
  </li>
</ul>

<router-outlet></router-outlet>

```

Wild card route

The (**) wild card route is matches by every URL and *must be placed last.*. This is used to display the *Page not found* error message, when the URL does not match any routes.

```

{ path: '**', component: NotFoundComponent }

```

Open the AppModule and add a new component not-found.component.ts

```

import { Component } from '@angular/core';

@Component({
  template: '<h1>Not Found</h1>',
})
export class NotFoundComponent {
  title = '';
}
```

```
10
```

Add the AppRoutingModule to include the NotFoundComponent

```
1
2  import { NgModule } from '@angular/core';
3  import { Routes, RouterModule } from '@angular/router';
4
5  import { NotFoundComponent } from './not-found.component';
6
7  const routes: Routes = [
8    { path: '**', component: NotFoundComponent }
9  ];
10
11 @NgModule({
12   declarations:[NotFoundComponent],
13   imports: [RouterModule.forRoot(routes)],
14   exports: [RouterModule]
15 })
16 export class AppRoutingModule { }
17
```

And AppModule , move the AppRoutingModule to last in the imports. This will ensure that the wild card route is always at the last place.

```
1
2  import { BrowserModule } from '@angular/platform-browser';
3  import { NgModule } from '@angular/core';
4
5  import { AppRoutingModule } from './app-routing.module';
6  import { AppComponent } from './app.component';
7
8  import { AdminModule} from './admin/admin.module';
9
10 @NgModule({
11   declarations: [
12     AppComponent
13   ],
14   imports: [
15     BrowserModule,
16     AdminModule,
17
```

```
    AppRoutingModule,   //this must be called last
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

It may happen that you may have other routes defined in the **AppRoutingModule** and might not be able to import it last. In that case create another routing module and place only the wild card route in it and import it in the AppModule.

Summary

In this module, we learnt how to define routes in feature modules. We also looked at how to add child routes and wild card routes.

← Angular Modules | Angular Tutorial | Angular Folder Structure →

Related Posts