AppDividend

Home › Angular ›

ANGULAR

Angular 7 Routing And Sub Routing Tutorial With Example

By **Krunal** Last updated **Dec 19, 2018**

Angular 7 Routing and Sub Routing Tutorial With Example is today's leading topic. If you want to navigate to different pages in your application, but you also want the application to be an **SPA** (Single Page Application), with no page reloading, then your app needs routing and angular makes it very easy. Routing means navigating between the pages. You have seen many websites with links that direct you to the new page. This can be achieved using routing.

We also see the sub routing or children routing for our components. That means, in our application, there is one root route and other routes are for their respective components. If we want to make our Angular application modular, then it is the best practice to assign the routes module-wise. We will take an example of how we can create root route and child routes in this angular 7 routing and sub routing tutorial with an example.

If you want to learn more about Angular, then check out this Angular 7 – The complete Guide course.

Content Overview [hide]

Angular 7 Routing and Sub Routing Tutorial

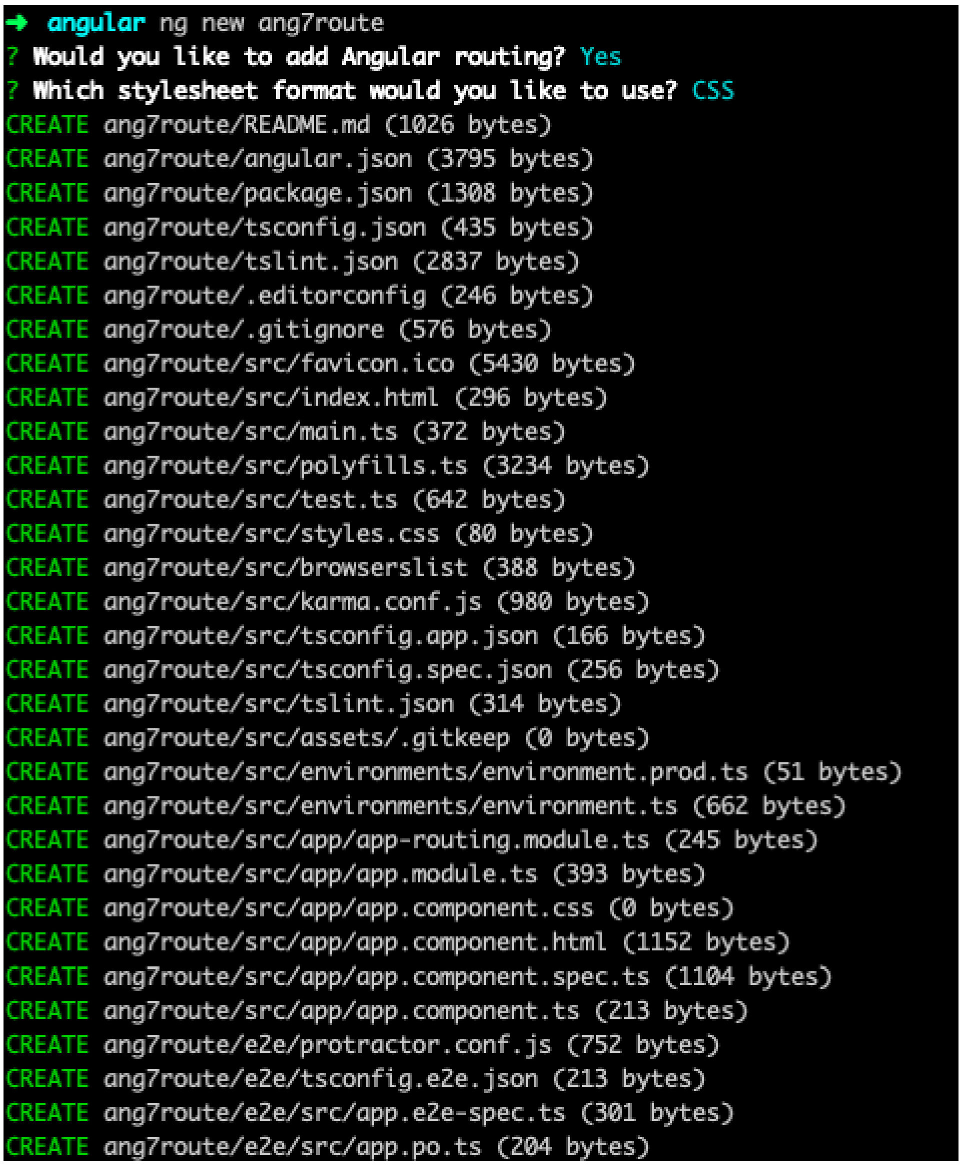
Now, the first thing is to create an Angular 7 project. Type the following command to create it. Please install or update Angular CLI, if you have not done already.

Step 1: Install Angular 7 Project

Type the following command to create it.

```
ng new ang7route
```

Remember, you need to add the app routing by saying yes to the prompt when you are creating a new project like this. Here I have allowed adding Angular routing.



```
➜ angular ng new ang7route
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE ang7route/README.md (1026 bytes)
CREATE ang7route/angular.json (3795 bytes)
CREATE ang7route/package.json (1308 bytes)
CREATE ang7route/tsconfig.json (435 bytes)
CREATE ang7route/tslint.json (2837 bytes)
CREATE ang7route/.editorconfig (246 bytes)
CREATE ang7route/.gitignore (576 bytes)
CREATE ang7route/src/favicon.ico (5430 bytes)
CREATE ang7route/src/index.html (296 bytes)
CREATE ang7route/src/main.ts (372 bytes)
CREATE ang7route/src/polyfills.ts (3234 bytes)
CREATE ang7route/src/test.ts (642 bytes)
CREATE ang7route/src/styles.css (80 bytes)
CREATE ang7route/src/browserslist (388 bytes)
CREATE ang7route/src/karma.conf.js (980 bytes)
CREATE ang7route/src/tsconfig.app.json (166 bytes)
CREATE ang7route/src/tsconfig.spec.json (256 bytes)
CREATE ang7route/src/tslint.json (314 bytes)
CREATE ang7route/src/assets/.gitkeep (0 bytes)
CREATE ang7route/src/environments/environment.prod.ts (51 bytes)
CREATE ang7route/src/environments/environment.ts (662 bytes)
CREATE ang7route/src/app/app-routing.module.ts (245 bytes)
CREATE ang7route/src/app/app.module.ts (393 bytes)
CREATE ang7route/src/app/app.component.css (0 bytes)
CREATE ang7route/src/app/app.component.html (1152 bytes)
CREATE ang7route/src/app/app.component.spec.ts (1104 bytes)
CREATE ang7route/src/app/app.component.ts (213 bytes)
CREATE ang7route/e2e/protractor.conf.js (752 bytes)
CREATE ang7route/e2e/tsconfig.e2e.json (213 bytes)
CREATE ang7route/e2e/src/app.e2e-spec.ts (301 bytes)
CREATE ang7route/e2e/src/app.po.ts (204 bytes)
```

Now, install the bootstrap css framework.

```
npm install bootstrap --save
```

Add the Bootstrap file inside the **angular.json** file.

```
"styles": [
     "src/styles.css",
     "./node_modules/bootstrap/dist/css/bootstrap.min.css"
],
```

Next step is to create one header component. So type the following command.

```
ng g c header --spec=false
```

We will create a navigation bar inside that component. So, write the following code inside the **header.component.html** file.

```
<!-- header.component.html -->

<nav class="navbar navbar-expand-lg navbar-light" style="background-color: #e3f2fd;">
    <a class="navbar-brand brand-custom" href="#">Angular 7 Routing Example</a>
    <div class="collapse navbar-collapse" id="navbarText">
        <ul class="navbar-nav ml-auto">
            <li class="nav-item">
                <a class="nav-link login-custom" href="#">Students <span class="sr-only">
(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link register-custom" href="#">Home</a>
            </li>
        </ul>
    </div>
</nav>
```

Now, finally replace the **app.component.html** code with the following code.

```
<!-- app.component.html -->

<div>
    <app-header></app-header>
</div>
```

Save the file and start the angular development server.

```
ng serve --open
```

You will see the navigation bar with three nav items.

So here, one item is **Home**, and one is **Students.**

That means, our application has one root route for home and others are sub-routes like for students module.

At the time of creating the project, we have created one routing module called **app-routing.module.ts.** So we will define the Root routes inside that file.

Step 2: Add Root Routes

First, create a home component by the following command.

```
ng g c home --spec=false
```

Now, add that component inside the **app-routing.module.ts** file.

```
// app-routing.component.ts

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

import { HomeComponent } from './home/home.component';

const routes: Routes = [
    {
        path: 'home',
        component: HomeComponent
    }
];

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

So, here we have defined the root routes for our angular application. Now add the **router-outlet** inside the **app.component.html** file to display the content of the home component.

```
<!-- app.component.html -->

<div>
    <app-header></app-header>
    <div class="container">
        <router-outlet></router-outlet>
    </div>
</div>
```

Also, add the navigation link inside the **header.component.html** file.

```
<!-- header.component.html -->

<li class="nav-item">
     <a class="nav-link register-custom" routerLink="/home">Home</a>
</li>
```

Save the file and go to the browser and click the **Home** link. You can see that we can see the content of the **home.component.html** file. So, we have taken care of the Root routes. Now, it is time to create a student module and also define the sub-routes of the student module.

Step 3: Create a student module and components.

The first step is to create a module called the student. So let us create using the following command.

```
ng g module student
```

So, it will create a folder inside the app folder called the **student**, and inside that folder, it will create a **student.module.ts** file.

Next step is to create the three angular components related to student module. So let us do that.

```
ng g c student/student --spec=false
ng g c student/student-list --spec=false
ng g c student/student-detail --spec=false
```

It will create the three folders inside the **src** >> **app** >> **student** folder.

Now, all these four components are already imported inside the **student.module.ts** file.

```
// student.module.ts

import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { StudentComponent } from './student/student.component';
import { StudentListComponent } from './student-list/student-list.component';
import { StudentDetailComponent } from './student-detail/student-detail.component';

@NgModule({
  declarations: [StudentComponent, StudentListComponent, StudentDetailComponent],
  imports: [
    CommonModule
  ]
})
export class StudentModule { }
```

Now, we do not need to import all these components inside the **app.module.ts** file.

Instead, we need to import this **student.module.ts** file inside the **app.module.ts** file.

```
// app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { StudentModule } from './student/student.module';

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';

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

So, all of our student components are registered to the angular application.

Step 4: Create Student route.

Now, inside the **src >> app >> student** folder, we can create a routing file called **student-routing.module.ts** and add the following code inside it.

```
// student-routing.module.ts

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

import { StudentComponent } from './student/student.component';
import { StudentListComponent } from './student-list/student-list.component';
import { StudentDetailComponent } from './student-detail/student-detail.component';

const routes: Routes = [
    {
        path: 'student',
        component: StudentComponent,
        children: [
            {
                path: 'list',
                component: StudentListComponent
            },
            {
                path: 'detail',
                component: StudentDetailComponent
            }
        ]
    }
];


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

So, here we have defined the sub-routing for the student module. The main path is a **/student** and its children are **/student/list** and **/student/detail.**

So that means, we have defined the subroutes for the student module. Now, the only thing remaining is to register this routing module to the **student.module.ts** file.

Remember, both **student.module.ts,** and **student-routing.module.ts** files are different. You can see this structure as same as our root angular project structure like **app.module.ts** and **app-routing.module.ts.**

```
// student.module.ts

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

import { StudentRoutingModule } from './student-routing.module';

import { StudentComponent } from './student/student.component';
import { StudentListComponent } from './student-list/student-list.component';
import { StudentDetailComponent } from './student-detail/student-detail.component';

@NgModule({
  declarations: [StudentComponent, StudentListComponent, StudentDetailComponent],
  imports: [
    CommonModule,
    StudentRoutingModule
  ]
})
export class StudentModule { }
```

Now, we need to display the routes. So add the following code inside the **student.component.html** file.

```
<!-- student.component.html -->

<div class="container">
    <router-outlet></router-outlet>
</div>
```

This **router-outlet** will only show the component related to the **student** module. So it is different from the root routing's router-outlet which is still in the place inside the **app.component.html** file.

Also, now add the router link inside the **header.component.html** file.

```
<!-- header.component.html -->

<nav class="navbar navbar-expand-lg navbar-light" style="background-color: #e3f2fd;">
    <a class="navbar-brand brand-custom" href="#">Angular 7 Routing Example</a>
    <div class="collapse navbar-collapse" id="navbarText">
        <ul class="navbar-nav ml-auto">
            <li class="nav-item">
                <a class="nav-link login-custom" routerLink="/student/list">Students <spa
n class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link register-custom" routerLink="/home">Home</a>
            </li>
        </ul>
    </div>
</nav>
```

Save the file and to the browser and navigate to the **http://localhost:4200/student/list**

You can see that it is rendering the correct component. Now, go to the **http://localhost:4200/student/detail**

It will also show the right component, and now our student module is working.

You can still go to the **http://localhost:4200/home**, and it will render the correct component which is HomeComponent.

This is how you can organize your Angular Project module wise with the root and children routing.

Summary of Angular Routing

1. You added the Angular router to navigate among different components.
2. You turned the AppComponent into a navigation shell with `<a>` links and a `<router-outlet>`.
3. You configured the router in an **AppRoutingModule**.
4. You configured the router in the **StudentRoutingModule.**
5. You defined simple routes, a redirect route.
6. You used the `routerLink` directive in anchor elements.

Finally, **Angular 7 Routing and Sub Routing Tutorial With Example** is over. Thanks for taking.

GITHUB CODE

Angular Angular 7

Krunal - 575 Posts - 176 Comments

Krunal Lathiya is From India, and he is an Information Technology Engineer. By profession, he is the latest web and mobile technology adapter, freelance developer, Machine Learning, Artificial Intelligence enthusiast, and primary Author of this blog.

2 COMMENTS

Vishnu Says *3 months ago*

thank you

Dgrfg Says *2 months ago*

with parameters?

This site uses Akismet to reduce spam. Learn how your comment data is processed.
