Routing in Angular

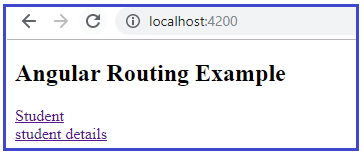
Why Routing in Angular Application?

We access our application through one URL such as http://localhost:4200 and our application is not aware of any other URLs such as http://localhost:4200/Login

Most web applications need to support different URLs to navigate different pages of the application. This is where angular routing comes into picture.

Understanding Angular Routing with an example:

Let us understand the Routing in Angular Application with an example. We want to create a page as shown below.



Configuring Routes in Angular Application:

Once you created the routing module, then you need to configure the path and their respective component in the AppRoutingModule as shown in the below image. As you can see, here we have created two paths i.e. studentLink and studentdetailsLink (you need to use the path properties to set the path. You can give any meaningful name here) and also set the respective components using the component property (Here you need to provide the component class name).

**import** **{** NgModule **}** from '@angular/core';

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

**import** **{** StudentComponent **}** from './student/student.component';

**import** **{** StudentdetailComponent **}** from './studentdetail/studentdetail.component';

const routes: Routes = **[**

**{**

path:'studentLink', component:StudentComponent

**}**,

**{**

path:'studentdetailsLink', component: StudentdetailComponent

**}**

**]**;

@NgModule**({**

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

exports: **[**RouterModule**]**

**})**

**export** **class** AppRoutingModule **{** **}**

Note: While generating the link you need to use the string studentLink and studentdetailsLink. Let us see how to use these routing path to generate link and navigates.

Generating to Links:

In order to generate links, open app.component.html file and then copy and paste the following code in it.

**<h2>**Angular Routing Example**</h2>**

**<a** [routerLink] = "['/studentLink']" **>**Student**</a>** **<br/>**

**<a** [routerLink] = "['/studentdetailsLink']" **>**student details**</a>**

**<div>**

**<router-outlet></router-outlet>**

**</div>**

Router-outlet:

The Router Outlet is a dynamic component that the router uses to display views based on router navigation. In our example, whenever the user clicks on the Student Link, then it will display student component view in the router-outlet div. So, the role of <router-outlet> is to mark where the router displays the view. This is the location where angular will insert the component.

The <router-outlet> tells the router where to display routed view. The RouterOutlet is one of the router directives that become available to the AppComponent because AppModule imports AppRoutingModule which exported RouterModule.

Router Link:

With the help of routerLink directive, you can link to routes of your application right from the HTML Template. You just need to add the directive to an HTML Element. When the user clicks on that element, angular navigates to that specified location.

The routerLink is the selector for the RouterLink directive that turns user clicks into router navigations. You can assign a string to the Router link. This directive generates the link based on the route path.

The above is client side.

**Router Link: Server side**

Sometimes it is also required to set the route dynamically based on some condition and that can be done at server side. For your application to work with server side rendering, the element hosting directive has to be a link (anchor) element.

It is also possible to navigate to a route from code. To do so, we need angular router and this need to be done in your typescript file. The syntax is given below.

Router Link: Server side

Once we have the router, then the navigation is quite simple. Just call the navigate function or Router. This function takes an array. The first element of the array defines the route we want to navigate. The second is optional and allows us to pass a route parameter. The syntax is given below.

Understanding Angular Routing with an example

First modify the **app.component.ts** file as shown below.

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

**import** **{**Router**}** from '@angular/router';

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

**constructor(private** router : Router**){}**

GetStudent**()**

**{**

this.router.navigate**([**'/studentLink'**])**;

**}**

GetStudentDetails**()**

**{**

this.router.navigate**([**'/studentdetailsLink'**])**;

**}**

**}**

Then modify **app.component.html** file as shown below.

**<h2>**Angular Routing Example**</h2>**

**<button** (click)="GetStudent()"**>**Student**</button>**

**<button** (click)="GetStudentDetails()"**>**GetStudentDetails**</button>**

**<div>**

**<router-outlet></router-outlet>**

**</div>**

##### ****Redirecting Routes in Angular:****

When the application start, it navigates to the empty route by default. We can configure the router to redirect to a named route by default. So, a redirect route translates the initial relative URL (”) to the desired default path. For example, if may want to redirect to Login page or registration page by default when the application start. Then you need to configure the redirectTo as shown below.

