* Navigation is one of the important aspects in a web application. Even though a single page application (SPA) does not have multiple page concepts, it does moves from one view to another view. Providing clear and understandable navigation elements decides the success of an application.
* Angular provides extensive set of navigation feature to accommodate simple scenario to complex scenario. The process of defining navigation element and the corresponding view is called **Routing**. Angular provides a separate module, **RouterModule** to set up the navigation in the Angular application.

**Steps to perform page routing:**

1. **Import [RouterModule](https://angular.io/api/router/RouterModule) and**[**Routes**](https://angular.io/api/router/Routes)**into your routing module in app.module.ts file**
2. **Define your routes in your**[**Routes**](https://angular.io/api/router/Routes)**array:**

* Each route in this array is a JavaScript object that contains two properties. The first property, path, defines the URL path for the route. The second property, component, defines the component Angular should use for the corresponding path.

1. Configures the imports and exports arrays for @NgModule(). In the RouterModule.forRoot() method you pass the routes array. This method creates and configures an NgModule with all the router providers and directives.
2. Update your component template to include <[router-outlet](https://angular.io/api/router/RouterOutlet)>. This element informs Angular to update the application view with the component for the selected route.

**app.module.ts**

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

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

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

import { AppComponent } from './app.component';

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

import { UsersComponent } from './users/users.component';

import { ServersComponent } from './servers/servers.component';

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

import { EditServerComponent } from './servers/edit-server/edit-server.component';

import { ServerComponent } from './servers/server/server.component';

import { FormsModule } from '@angular/forms';

import { ServersService } from './servers/servers.service';

const appRoutes: Routes=[

  {path:'',component:HomeComponent},

  {path:'users',component:UserComponent},

  {path:'servers',component:ServersComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    HomeComponent,

    UsersComponent,

    ServersComponent,

    UserComponent,

    EditServerComponent,

    ServerComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

    RouterModule.forRoot(appRoutes)

  ],

  providers: [ServersService],

  bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.html**

<div class="container">

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <ul class="nav nav-tabs">

        <li role="presentation" class="active"><a href="#">Home</a></li>

        <li role="presentation"><a href="#">Servers</a></li>

        <li role="presentation"><a href="#">Users</a></li>

      </ul>

    </div>

  </div>

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <router-outlet></router-outlet>

  </div>

</div>

**Route order:**

* The order of routes is important because the [Router](https://angular.io/api/router/Router) uses a first-match wins strategy when matching routes, so more specific routes should be placed above less specific routes.
* List routes with a static path first, followed by an empty path route, which matches the default route. The [wildcard route](https://angular.io/guide/router#setting-up-wildcard-routes) comes last because it matches every URL and the [Router](https://angular.io/api/router/Router) selects it only if no other routes match first.

**Navigating with Router Links:**

* To create a link to navigate to a given route, routerLink is used in Angular routing application rather than href. If you use href then clicking a link results in a page reload and that’s something you don't want in your single page apps. Using RouterLink directive to create a link to route doesn't cause full page reload.

<div class="container">

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <ul class="nav nav-tabs">

        <li role="presentation" class="active"><a routerLink="/">Home</a></li>

        <li role="presentation"><a routerLink="/servers">Servers</a></li>

        <li role="presentation"><a [routerLink]="['/users']">Users</a></li>

      </ul>

    </div>

  </div>

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <router-outlet></router-outlet>

  </div>

</div>

**Styling Active Router Links:**

* **routerLinkActive="active"** ensures that the currently selected menu option is highlighted.
* **[rouerLinkActive]** tracks whether the linked route of an element is currently active, and allows you to specify one or more CSS classes to add to the element when the linked route is active.
* To add the classes only when the URL matches the link exactly, **add the option exact: true**

<div class="container">

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <ul class="nav nav-tabs">

        <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}"><a routerLink="/">Home</a></li>

        <li role="presentation" routerLinkActive="active"><a routerLink="/servers">Servers</a></li>

        <li role="presentation" routerLinkActive="active"><a [routerLink]="['/users']">Users</a></li>

      </ul>

    </div>

  </div>

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <router-outlet></router-outlet>

  </div>

</div>

**Navigate to a route programmatically:**

* You may have a scenario where you would like to navigate to a route based on an event like click of a button, clicking a link.

**home.component.html**

<h4>Welcome to Server Manager 4.0</h4>

<p>Manage your Servers and Users.</p>

<button class="btn btn-primary" (click)="onLoadServers()">Load Servers</button>

**home.component.ts**

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

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  constructor(private router:Router) { }

  ngOnInit() {

  }

  onLoadServers()

  {

    this.router.navigate(['/servers']);

  }

}

**Relative Path:** In router.navigate() method you can also pass a second argument of type navigationExtras. One of the NavigationExtras is relativeTo which specifies a root URI to use for relative navigation.

**Making initial path relative means specifying / is mandatory while specifying next path.**

**home.component.ts**

<div class="container">

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <ul class="nav nav-tabs">

        <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}"><a routerLink="">Home</a></li>

        <li role="presentation" routerLinkActive="active"><a routerLink="servers">Servers</a></li>

        <li role="presentation" routerLinkActive="active"><a [routerLink]="['users']">Users</a></li>

      </ul>

    </div>

  </div>

  <div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <router-outlet></router-outlet>

  </div>

</div>

**server.component.html**

<div class="row">

  <div class="col-xs-12 col-sm-4">

    <div class="list-group">

      <a

        href="#"

        class="list-group-item"

        \*ngFor="let server of servers">

        {{ server.name }}

      </a>

    </div>

  </div>

  <div class="col-xs-12 col-sm-4">

    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>

    <app-edit-server></app-edit-server>

    <hr>

    <app-server></app-server>

  </div>

</div>

**server.component.ts**

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

import { ServersService } from './servers.service';

import {Router,ActivatedRoute} from '@angular/router'

@Component({

  selector: 'app-servers',

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

  styleUrls: ['./servers.component.css']

})

export class ServersComponent implements OnInit {

  public servers: {id: number, name: string, status: string}[] = [];

  constructor(private serversService: ServersService,private router:Router,private route:ActivatedRoute) { }

  ngOnInit() {

    this.servers = this.serversService.getServers();

  }

  onReload()

  {

    this.router.navigate(['/servers'],{relativeTo:this.route});

  }

}

**Passing Parameters to Routes & Fetching Route Parameters:**

**app.module.ts**

const appRoutes: Routes=[

  {path:'',component:HomeComponent},

  {path:'users',component:UserComponent},

  {path:'users/:id/:name',component:UserComponent},

  {path:'servers',component:ServersComponent}

]

**user.component.ts**

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

import {ActivatedRoute} from '@angular/router';

@Component({

  selector: 'app-user',

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

  styleUrls: ['./user.component.css']

})

export class UserComponent implements OnInit {

  user!: {id: number, name: string};

  constructor(private route:ActivatedRoute)

  {}

  ngOnInit(): void {

    this.user={

      id:this.route.snapshot.params['id'],

      name:this.route.snapshot.params['name']

    };

  }

}

**user.component.html**

<p>User with ID {{user.id}} loaded.</p>

<p>User name is {{user.name}}</p>

**Fetching Route Parameters Reactively:**

**user.component.ts**

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

import {ActivatedRoute,Params} from'@angular/router';

@Component({

  selector: 'app-user',

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

  styleUrls: ['./user.component.css']

})

export class UserComponent implements OnInit {

  user!: {id: number, name: string};

  constructor(private route:ActivatedRoute)

  {}

  ngOnInit(): void {

    this.user={

      id:this.route.snapshot.params['id'],

      name:this.route.snapshot.params['name']

    };

    this.route.params.subscribe(

      (params:Params)=>{

        this.user.id=params['id'];

        this.user.name=params['name'];

      }

    )

  }

}

**Passing Query Parameters & Fragments**:

**servers.component.html**

<div class="row">

  <div class="col-xs-12 col-sm-4">

    <div class="list-group">

      <a

      [routerLink]="['/servers',5,'edit']"

      [queryParams]="{allowEdit:'1'}"

      fragment="loading"

        href="#"

        class="list-group-item"

        \*ngFor="let server of servers">

        {{ server.name }}

      </a>

    </div>

  </div>

  <div class="col-xs-12 col-sm-4">

    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>

    <app-edit-server></app-edit-server>

    <hr>

    <app-server></app-server>

  </div>

</div>

**app.module.ts**

const appRoutes: Routes=[

  {path:'',component:HomeComponent},

  {path:'users',component:UserComponent},

  {path:'users/:id/:name',component:UserComponent},

  {path:'servers',component:ServersComponent},

  {path:'servers/:id/edit',component:ServersComponent}

]

[**home.component.html**](http://home.component.html)

<h4>Welcome to Server Manager 4.0</h4>

<p>Manage your Servers and Users.</p>

<button class="btn btn-primary" (click)="onLoadServer(1)">Load Server 1</button>

[**home.component.ts**](http://home.component.ts)

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

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  constructor(private router:Router) { }

ngOnInit() {

  }

  onLoadServer(id:number)

  {

    this.router.navigate(['/servers',id,'edit'],{queryParams:{allowEdit:'1'},fragment:'laoding'});

  }

}

**Retrieving query parameters & Fragments:**

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

import { ActivatedRoute } from '@angular/router';

import { ServersService } from '../servers.service';

@Component({

  selector: 'app-edit-server',

  templateUrl: './edit-server.component.html',

  styleUrls: ['./edit-server.component.css']

})

export class EditServerComponent implements OnInit {

  server!: {id: number, name: string, status: string};

  serverName = '';

  serverStatus = '';

constructor(private serversService: ServersService,private route:ActivatedRoute) { }

ngOnInit() {

    console.log(this.route.snapshot.queryParams);

    console.log(this.route.snapshot.fragment);

    this.server = this.serversService.getServer(1)!;

    this.serverName = this.server.name;

    this.serverStatus = this.server.status;

  }

onUpdateServer() {

    this.serversService.updateServer(this.server.id, {name: this.serverName, status: this.serverStatus});

  }

}

**Setting up Child routes:**

**app.module.ts**

const appRoutes: Routes=[

  {path:'',component:HomeComponent},

  {path:'users',component:UsersComponent,children:[{path:':id/:name',component:UserComponent}]},

  {path:'servers',component:ServersComponent,children:[

    {path:':id',component:ServerComponent},

    {path:':id/edit',component:EditServerComponent}

  ]}

]

**users.component.html**

<div class="row">

  <div class="col-xs-12 col-sm-4">

    <div class="list-group">

      <a

        href="#"

        class="list-group-item"

        \*ngFor="let user of users">

        {{ user.name }}

      </a>

    </div>

  </div>

  <div class="col-xs-12 col-sm-4">

    <router-outlet></router-outlet>

  </div>

</div>

**servers.component.html**

<div class="row">

  <div class="col-xs-12 col-sm-4">

    <div class="list-group">

      <a

      [routerLink]="['/servers',5,'edit']"

      [queryParams]="{allowEdit:'1'}"

      fragment="loading"

        href="#"

        class="list-group-item"

        \*ngFor="let server of servers">

        {{ server.name }}

      </a>

    </div>

  </div>

  <div class="col-xs-12 col-sm-4">

    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>

   <router-outlet></router-outlet>

  </div>

</div>

**QueryParamsHandling:**

How to handle query parameters in a router link. One of:

* "merge" : Merge new parameters with current parameters.
* "preserve" : Preserve current parameters.
* "" : Replace current parameters with new parameters. This is the default behavior.

type [QueryParamsHandling](https://angular.io/api/router/QueryParamsHandling) = 'merge' | 'preserve' | '';

**server.component.ts**

export class ServerComponent implements OnInit {

  server!: {id: number, name: string, status: string};

  constructor(private serversService: ServersService,private route:ActivatedRoute,private router:Router) { }

  ngOnInit() {

    this.server = this.serversService.getServer(1)!;

  }

 onEdit()

 {

  this.router.navigate(['edit'],{relativeTo:this.route,queryParamsHandling:'merge'});

 }

}

**Wild card route in Angular:**

For setting up a wild card route in Angular you use ‘\*\*’ (double asterisk). This wild card route catches all routes that are not configured with in the route definition.

**{ path: '\*\*', component: COMPONENT\_NAME}**

Do ensure that the wildcard route is the last one in the route definition ordering as wild card route matches every URL.

This order of routes is important because the Router uses a first-match wins strategy. When matching routes if wildcard route, which matches every URL, is at the top then the component paired with wild route will be the one always called. Follow the order of-

* List routes with a static path first.
* Empty path route (with a possible redirection).
* The wildcard route comes last.

const appRoutes: Routes=[

  {path:'',component:HomeComponent},

  {path:'users',component:UsersComponent,children:[{path:':id/:name',component:UserComponent}]},

  {path:'servers',component:ServersComponent,children:[

    {path:':id',component:ServerComponent},

    {path:':id/edit',component:EditServerComponent}

  ]},

  {path:'not-found',component:PageNotFoundComponent},

  {path:'\*\*',redirectTo:'not-found'}

]

We can redirect to a fixed page on encountering wild card route.

**Outsourcing the Route Configuration:**

**app-routing.module.ts**

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

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

import { ServerComponent } from './servers/server/server.component';

import { ServersComponent } from './servers/servers.component';

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

import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

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

import { UsersComponent } from './users/users.component';

import { EditServerComponent } from './servers/edit-server/edit-server.component';

const appRoutes: Routes=[

    {path:'',component:HomeComponent},

    {path:'users',component:UsersComponent,children:[{path:':id/:name',component:UserComponent}]},

    {path:'servers',component:ServersComponent,children:[

      {path:':id',component:ServerComponent},

      {path:':id/edit',component:EditServerComponent}

    ]},

    {path:'not-found',component:PageNotFoundComponent},

    {path:'\*\*',redirectTo:'not-found'}

  ]

  @NgModule({

    imports: [

      RouterModule.forRoot(appRoutes)

    ],

    exports:[RouterModule]

})

export class AppRoutingModule

{

}

**app.module.ts**

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

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

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

import { AppComponent } from './app.component';

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

import { UsersComponent } from './users/users.component';

import { ServersComponent } from './servers/servers.component';

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

import { EditServerComponent } from './servers/edit-server/edit-server.component';

import { ServerComponent } from './servers/server/server.component';

import { FormsModule } from '@angular/forms';

import { ServersService } from './servers/servers.service';

import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

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

@NgModule({

  declarations: [

    AppComponent,

    HomeComponent,

    UsersComponent,

    ServersComponent,

    UserComponent,

    EditServerComponent,

    ServerComponent,

    PageNotFoundComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

    AppRoutingModule

  ],

  providers: [ServersService],

  bootstrap: [AppComponent]

})

export class AppModule { }