Craate components

Produng generate modulects

Abouts

Productdetails

Home

Contact us

ng generate module module name

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

**import { CommonModule } from '@angular/common';**

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

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

**import { ContactUsComponent } from '../contact-us/contact-us.component';**

**import { AboutComponent } from '../about/about.component';**

**import { ProductsComponent } from '../products/products.component';**

**import { ProductDetailsComponent } from '../product-details/product-details.component';**

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

**//import { ProdcutsComponent } from '../prodcuts/prodcuts.component';**

**const routes=[**

**{path:'home',component:HomeComponent},**

**{path:'', redirectTo:'/home',pathMatch:'full'},**

**{path:'ContactUs',component:ContactUsComponent},**

**{path:'About',component:AboutComponent},**

**{path:'Products',component:ProductsComponent},**

**{path:'ProductDetails',component:ProductDetailsComponent},**

**{path:'\*\*',component:PageNotFoundComponent},**

**];**

**@NgModule({**

**declarations: [**

**],**

**imports: [**

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

**],**

**exports: [RouterModule]**

**})**

**export class AppRoutingModule { }**

**App.module.ts**

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

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

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

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

**import { AboutComponent } from './about/about.component';**

**import { ContactUsComponent } from './contact-us/contact-us.component';**

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

**import { ProductsComponent } from './products/products.component';**

**import { ProductDetailsComponent } from './product-details/product-details.component';**

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

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

**@NgModule({**

**declarations: [**

**AppComponent,**

**HomeComponent,**

**AboutComponent,**

**ContactUsComponent,**

**ProductsComponent,**

**ProductDetailsComponent,**

**PageNotFoundComponent**

**],**

**imports: [**

**BrowserModule,**

**RouterModule,**

**AppRoutingModule.ts**

**],**

**providers: [],**

**bootstrap: [AppComponent]**

**})**

**export class AppModule { }**

**app.component,html**

**<div>**

**<a routerLink='/home'>Home</a><br>**

**<a routerLink='/Products'>Products</a><br>**

**<a routerLink='/ContactUs'>ContactUs</a><br>**

**<a routerLink='/About'>About</a><br>**

**<a routerLink='/ProductDetails'>ProductDetails</a><br><br>**

**</div>**

**<div>**

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

**</div>**

**Create a service to get products**

**Ng generate Service Prouctlist**

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

@Injectable({

  providedIn: 'root'

})

export class ProductlistService {

  constructor() { }

  GetAllProdoucts():any

  {

 const products:any[] =[

    {productid:100,productname:'tv',price:200,Quantity:1},

    {productid:200,productname:'refrigirator',price:300,Quantity:1},

    {productid:300,productname:'LAPTOP',price:400,Quantity:1},

    {productid:400,productname:'AC',price:500,Quantity:1},

    {productid:500,productname:'FAN',price:600,Quantity:1},

  ]

  return products;

  }

}

App.component.ts

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

import { ProductlistService } from "./productlist.service";

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'ng-routing';

   Products:any[];

  constructor(private \_productlistService:ProductlistService)

   {

 this.Products =this.\_productlistService.GetAllProdoucts();

  }

}

App.component.html

<div>

    <a routerLink='/home'>Home</a><br>

    <a routerLink='/Products'>Products</a><br>

    <a routerLink='/ContactUs'>ContactUs</a><br>

    <a routerLink='/About'>About</a><br>

    <a routerLink='/ProductDetails'>ProductDetails</a><br><br>

</div>

<div>

    <router-outlet></router-outlet>

    <div>

        <p>products works!</p>

        <table>

            <thead>

                <tr>

                    <th>

                        ProductID

                    </th>

                    <th>

                        ProductName

                    </th>

                    <th>

                        Price

                    </th>

                    <th>

                        Quantity

                    </th>

                </tr>

            </thead>

            <tbody>

                <tr \*ngFor="let product of Products">

                    <td>{{product.productid}}</td>

                    <td>{{product.productname}}</td>

                    <td>{{product.Quantity}}</td>

                    <td>{{product.price}}</td>

                </tr>

            </tbody>

        </table>

    </div>

</div>

ngular Route & Query Parameters Tutorial

In this Angular tutorial we learn how to send query data or dynamic data through an app's Routes with Query & Route parameters

We also cover how to use observables to fetch the data we sent.

* [What are Route Parameters](https://www.koderhq.com/tutorial/angular/routing-parameter/#route-param)
* [How to setup a Route Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#setup-route)
* [How to send a Route Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#send-route)
* [How to access a Route Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#access-route)
  + [Using the snapshot property to access a Route Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#route-snapshot)
  + [Using an Observable to access the Route Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#route-observable)
  + [Why should we use an Observable](https://www.koderhq.com/tutorial/angular/routing-parameter/#why-observable)
* [What are Query Parameters](https://www.koderhq.com/tutorial/angular/routing-parameter/#query-param)
* [How to send a Query Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#send-query)
* [How to access a Query Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#access-query)
  + [Using the snapshot property to access a Query Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#query-snapshot)
  + [Using an Observable to access the Query Parameter](https://www.koderhq.com/tutorial/angular/routing-parameter/#query-observable)

What are Route Parameters

Route parameters allow us to send dynamic data through a route by appending it to the URL as a segment.

For example, if we want to send a user’s ID to another component, we could do it with a route parameter. The URL would look something like the following.

* /user/1
* /user/2
* etc.

Our app may have many users so it’s impractical to hardcode their IDs in separate router links. Instead, we use a placeholder, like ID, for the actual data.

1. Before we go any further, let’s set up an ‘employee’ component to help with demonstration.

Example: CLI command

ng generate component employee

2. Next, set up a route to the new component.

* Import **Routes** and **RouterModule** from ‘@angular/router’.
* Set up a constant array of type Routes with the path and component name.
* Specify the array for Angular to use in the **RouterModule.forRoot()** method in the imports.

Example: app.module.ts

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

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

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

import { EmployeeComponent } from './employee/employee.component';

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

const appRoutes: Routes = [

{ path: 'employee', component: EmployeeComponent }

];

@NgModule({

declarations: [

AppComponent,

EmployeeComponent

],

imports: [

BrowserModule,

RouterModule.forRoot(appRoutes)

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

3. In the main ‘app’ component’s View, let’s add a router outlet and some menu items.

Example: app.component.html

<ul>

<li><a routerLink="/">Home</a></li>

<li><a routerLink="/employee">Employee</a></li>

</ul>

<hr>

<router-outlet></router-outlet>

How to setup a Route Parameter

To define a parameter we add a slash, followed by a colon and the placeholder name in our route definition.

Syntax:

{ path: 'path\_to\_component/:placeholder\_name', component: component\_name }

If we want to send data to the current component, we simply remove its path before the placeholder.

Syntax:

{ path: '/:placeholder\_name', component: component\_name }

As an example, let’s say we want to send the employee’s ID to the ‘employee’ component.

Example: app.module.ts

...

const appRoutes: Routes = [

{ path: 'employee', component: EmployeeComponent },

{ path: 'employee/:id', component: EmployeeComponent }

];

...

We can now use an ID number in the URL, like: [**http://localhost:4200/employee/1**](http://localhost:4200/employee/1)

How to send a Route Parameter

We don’t want users to have to type parameters manually in the URL bar, we want to send the parameters through code with links, images or buttons.

To do this we use the routerLink directive with the [**Link Parameters**](https://www.koderhq.com/tutorial/angular/routing/#routerlink) array.

Syntax:

<a [routerLink]="['/path', 'parameter']">Link</a>

Let’s create a normal link as well as a button in the main ‘app’ component’s View. These should go to the ‘employee’ component with a number as the ID.

Example: app.component.html

<ul>

<li><a routerLink="/">Home</a></li>

<li><a routerLink="/employee">Employee</a></li>

</ul>

<hr>

<a [routerLink]="['/employee', '1']">Employee 1</a><br><br>

<button [routerLink]="['/employee', '2']">Employee 2</button>

<router-outlet></router-outlet>

Clicking on the links will show the ID number in the URL.

We don’t need to include a slash operator. If we do, it will be converted to a “safe” ASCII url.

Example: app.component.html

<ul>

<li><a routerLink="/">Home</a></li>

<li><a routerLink="/employee">Employee</a></li>

</ul>

<hr>

<a [routerLink]="['/employee', '/1']">Employee 1</a><br><br>

<button [routerLink]="['/employee', '2']">Employee 2</button>

<router-outlet></router-outlet>

If we click on the ‘Employee 1’ link, the URL will be encoded into /employee/%2F1 , which is not what we want.

How to access a Route Parameter

Now that we’ve sent data through the route, we need to access the parameter to be able use the data.

We do this with the **[ActivatedRoute](https://angular.io/api/router/ActivatedRoute" \o "Angular ActivatedRoute)** service, which keeps track of the currently activated route associated with the loaded component.

To use the ActivatedRoute service we need to import and inject it into the component where we will be accessing the data. In our case that’s the ‘employee’ component.

Example: employee.component.ts

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

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

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

constructor(private activatedRoute : ActivatedRoute) { }

ngOnInit(): void { }

}

Angular adds all the route parameters in the **[ParamMap](https://angular.io/api/router/ParamMap" \o "Angular Router ParamMap)** object, which makes it easier to work with parameters.

*The older Params array is now deprecated and is replaced by the ParamMap.*

There are two ways to get the parameter value from the ParamMap object.

* Snapshot
* Observable

Using the snapshot property to access a Route Parameter

The snapshot property returns the initial value of a route. We can then access the paramMap array and retrieve the value of the parameter with the **get()** method.

Example: employee.component.ts

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

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

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

userID: any;

constructor(private activatedRoute : ActivatedRoute) { }

ngOnInit(): void {

this.userID = this.activatedRoute.snapshot.paramMap.get("id");

}

}

In the example above we get the value of the id parameter and store it in userID .

For demonstration, let’s show the ID in the component’s view with property binding.

Example: employee.component.html

<p>User ID: {{ userID }}</p>

Now when we click on the link or button, the parameter will show in the paragraph.

*The snapshot property is typically only used when we need the initial value because it doesn’t update after changes.*

Using an Observable to access the Route Parameter

We can retrieve the value of the parameter by subscribing to the paramMap observable property of the ActivatedRoute .

Example: employee.component.ts

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

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

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

userID: any;

constructor(private activatedRoute : ActivatedRoute) { }

ngOnInit(): void {

this.activatedRoute.paramMap.subscribe(params => {

this.userID = params.get('id');

});

}

}

*An observable is typically used when the value is expected to change over time.*

We cover Observables in more detail in the [**Observables**](https://www.koderhq.com/tutorial/angular/observable/) lesson.

Why should we use an Observable

We retrieve parameter values in the ngOnInit life cycle hook, when the component is initialized.

If a user navigates away from, then back to a component, Angular will reuse the existing instance of it instead of creating a new one.

This means that the ngOnInit hook is not called again, so any different value for the parameter will not show.

By using an observable, we will always retrieve the latest value of the parameter.

What are Query Parameters

Query Parameters are optional parameters that we can pass through a route. They are added to the end of a URL with a question mark.

Example:

/listings?page=4

Where page=4 is the parameter.

Query parameters are common in searches, pagination etc.

Route parameters are part of the route definition and is used by Angular to determine the route. Query parameters are optional and doesn’t stop Angular from navigating to a route.

How to send a Query Parameter

Because query parameters aren’t part of the route, we can send them using the queryParams directive or with the **router.navigate()** method.

When using the queryParams directive, we have to specify a Javascript object with key:value notation.

Syntax: queryParams

<a [routerLink]="['path']" [queryParams]="{ key:value }">Page 2</a>

This will resolve into key=value .

Example: app.component.html

<ul>

<li><a routerLink="/">Home</a></li>

<li><a routerLink="/employee">Employee</a></li>

</ul>

<hr>

<a [routerLink]="['/employee', '1']" [queryParams]="{ page:1 }">Employee 1</a><br><br>

<button [routerLink]="['/employee', '2']">Employee 2</button>

<router-outlet></router-outlet>

If we run the example above and click on the ‘Employee 1’ link, it will add page=1 to the end of the URL.

When navigating programmatically, we use the **router.navigate()** method with the queryParams Javascript object as the second argument. Again, we have to use key:value notation.

Syntax: router.navigate()

this.router\_object.navigate(['/path'], { queryParams: { key: value } });

Remember that the **Router** class must be imported from ‘@angular/router’ and injected in the component’s constructor before we can use the **navigate()** method.

Example: app.component.ts

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 r: Router) { }

navToPage(pageNum: number) {

this.r.navigate(['/employee', '2'], { queryParams: { page: pageNum } });

}

}

In the example above we wrap our link in a method that will allow us to easily specify a value for the query parameter.

Example: app.component.html

<ul>

<li><a routerLink="/">Home</a></li>

<li><a routerLink="/employee">Employee</a></li>

</ul>

<hr>

<a [routerLink]="['/employee', '1']" [queryParams]="{ page:1 }">Employee 1</a><br><br>

<button (click)="navToPage(4)">Employee 2</button>

<router-outlet></router-outlet>

Because we’re using a method, we changed the button from routerLink to a click event.

If we run the example above and click on the ‘Employee 2’ button, it will add page=4 to the end of the URL.

How to access a Query Parameter

We access query parameters in the same way as route parameters, with the ActivatedRoute, but instead of the ParamMap , we use the queryParamsMap .

*The*queryParams*array has been deprecated and is replaced with*queryParamsMap*.*

Using the snapshot property to access a Query Parameter

The snapshot property returns the initial value of a route. We can then access the queryParamMap array and retrieve the value of the parameter with the **get()** method.

Example: employee.component.ts

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

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

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

pageNum: any;

constructor(private activatedRoute : ActivatedRoute) { }

ngOnInit(): void {

this.pageNum = this.activatedRoute.snapshot.queryParamMap.get("page");

}

}

Let’s also output the value to the page.

Example: employee.component.html

<p>Page Number: {{ pageNum }}</p>

When we run the example above and click on the ‘Employee 1’ link, the page number 1 will show in both the URL and on the page.

But, if we click on the ‘Employee 2’ button, it will still show the page number as 1 on the page instead of 4. Like with the Route parameter, the snapshot property only holds the value when the component is initialized, it doesn’t hold any updates to the value from our **navToPage()** method.

Luckily, we can also use an observable to access the query parameter and get the latest value.

*The snapshot property is typically only used when we need the initial value because it doesn’t update after changes.*

Using an Observable to access the Query Parameter

We can retrieve the value of the query parameter by subscribing to the queryParamMap observable property of the ActivatedRoute .

Example: employee.component.ts

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

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

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

pageNum: any;

constructor(private activatedRoute : ActivatedRoute) { }

ngOnInit(): void {

this.activatedRoute.queryParamMap.subscribe(params => {

this.pageNum = params.get('page');

});

}

}

This time we get the latest value so both the link and the button will show the correct values on the page.

*An observable is typically used when the value is expected to change over time.*

How to pass data in angular route component 12 . ?

[Angular](https://edupala.com/category/angular/) / By [ngodup](https://edupala.com/author/tamo/" \o "View all posts by ngodup) / May 12, 2021

In [previous](https://edupala.com/how-to-implement-angular-routing/) articles, we have learned, what is routing and how to configure routing in an angular application in detail. In part two on Angular routing, we will learn how to pass data in angular route between components.

When we are building a scalable web application, then we need to send and retrieve data via routing. Let’s generate a couple of components to use for navigation on our previous example of routing articles 1.

ng generate component hotels

ng generate component hotel

ng generate component login

Table of Contents

* [Navigation within the application: Pass data in Angular route](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Navigation_within_the_application_Pass_data_in_Angular_route)
* [Using a state object of navigate() the method of Angular router object to pass data between components:](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Using_a_state_object_of_navigate_the_method_of_Angular_router_object_to_pass_data_between_components)
* [Reading state object data in navigate() method.](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Reading_state_object_data_in_navigate_method)
* [Using Route params : To pass data in Angular route](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Using_Route_params_To_pass_data_in_Angular_route)
  + [Navigating route with params in a template.](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Navigating_route_with_params_in_a_template)
  + [Navigating route with params in a component typescript.](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Navigating_route_with_params_in_a_component_typescript)
* [How to retrieve query param from an activated route in a component.](https://edupala.com/how-to-pass-data-in-angular-routed-component/#How_to_retrieve_query_param_from_an_activated_route_in_a_component)
* [Angular route navigate with paramter using queryParams property](https://edupala.com/how-to-pass-data-in-angular-routed-component/#Angular_route_navigate_with_paramter_using_queryParams_property)

**Navigation within the application: Pass data in Angular route**

Angular allows us to navigate between different components mainly using two approaches. First, we can use the router link directive in a template and the second router object in the typescript component. We will demonstrate router navigation using both approaches on the following link.

http://localhost:4200/about

http://localhost:4200/hotels/

http://localhost:4200/hotels/id

We have already created routing.module.ts  and import the routing module in app.module.ts  in the [previous example.](https://edupala.com/a-complete-guide-to-routing-in-angular/) Edit app-routing.module.ts to add a new routing for hotels and their children routes. We have hotels and its children route with hotels/: id. The slash in a route path is a dynamic value where we are passing hotel id.

import { LoginComponent } from './login/login.component';

import { HotelsComponent } from './hotels/hotels.component';

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

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

import { AboutComponent } from './about/about.component';

import { NavbarComponent } from './navbar/navbar.component';

import { EditHotelComponent } from './hotels/edit-hotel/edit-hotel.component';

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

const routes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'login', component: LoginComponent },

{ path: 'about', component: AboutComponent },

{ path: 'navbar', component: NavbarComponent },

{

path: 'hotels', component: HotelsComponent, children: [

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

]

}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

**Edit navbar.component.html to include a new route for hotels, login, and hotels/id .**

<nav class="navbar navbar-inverse">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<div id="navbar" class="collapse navbar-collapse">

<ul class="nav navbar-nav">

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

<li><a [routerLink]="['/about']">About</a></li>

<li><a [routerLink]="['/hotels']">Hotels</a></li>

</ul>

</div>

</div>

</nav>

We have used routerLink directive in the navbar component to navigating to about and hotels components. The routerLink directive pointing to the path of the component we want to visit. This directive can be applied to any element, although it is typically applied to button and anchor elements

**Using a state object of navigate() the method of Angular router object to pass data between components:**

Angular allows different ways of [passing data between components.](https://edupala.com/sharing-data-between-angular-component/) We will first learn and demonstrate the state object of navigate() method, this support came since the release of Angular 7.2. We want to send data from home to about components using the state object as shown in the above picture. When a user clicks on the send button in the home component it will call the sendRouteObject () method. This method sends data as an object and navigates to about component.

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

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

@Component({

selector: 'app-home',

template: `

<div class="container">

<div class="row">

<div class="col-md-6">

<h4>1. In template, route to hotels/id where id is 2</h4>

</div>

<div class="col-md-4">

<button class="btn btn-primary" [routerLink]="['/hotels/' + '2']">Navigation using RouterLink in template</button>

</div>

</div>

<div class="row">

<div class="col-md-6">

<h4>2. Programmatically route to hotels with params id </h4>

</div>

<div class="col-md-4"> <button class="btn btn-primary" (click)="routeToHotelId(2)">

Navigation using router object in typescript</button>

</div>

</div>

</div>

<div class="mt">

<h4>3. Send data from search to about component.</h4>

<p> {{ "{ site: 'edupala.com' frontEnd: { name: Angular version: 12 } }" }}

</p>

<button class="btn btn-primary" (click)="sendRouteObject()">Route to aboute component with state data</button>

</div>`,

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

})

export class HomeComponent {

constructor(private router: Router) { }

routeToHotelId(id: number) {

this.router.navigate(['/hotels/' + id]);

}

sendRouteObject() {

this.router.navigate(['/about'], {

state: {

frontEnd: JSON.stringify({ framwork: 'Angular', version: '9' }),

site: 'edupala.com'

}

});

}

}

The navigated method is one of the most used methods of router objects. The navigate takes an argument which is URL to navigate to the new route. We can also pass data via state object in navigate() method. In our example state object, we have two fields, a frontend object which we have to use JSON.stringify, and site as a simple variable we can pass directly.

**Reading state object data in navigate() method.**

Once we navigate to the requested component. At the activated route component about we can receive data as below and add the following code in the constructor object.

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

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

@Component({

selector: 'app-about',

template: `

<h4>About works!</h4>

<div \*ngIf="data.frontEnd">

<p>

Route state object data from homeomponent:

<strong>Data: {{ data.frontEnd?.framwork }} - {{ data.frontEnd?.version }} Site: {{ data.site }}</strong>

</p>

</div>`,

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

})

export class AboutComponent {

data: any = {};

routeState: any;

constructor(private router: Router) {

if (this.router.getCurrentNavigation().extras.state) {

this.routeState = this.router.getCurrentNavigation().extras.state;

if (this.routeState) {

this.data.frontEnd = this.routeState.frontEnd ? JSON.parse(this.routeState.frontEnd) : '';

this.data.site = this.routeState.site ? this.routeState.site : '';

}

}

}

}

**Using Route params : To pass data in Angular route**

Angular routes can be more flexible and we can pass data parameters to components in the routes as part of the URL path, for example, hotels/:id. Where id is the dynamic value of hotel id of hotel records. To demonstrate route params, let first create a hotel.service.ts class containing dummy hotel records and methods to manage the hotel records.

ng generate service hotels/hotel

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

@Injectable({ providedIn: 'root' })

export class HotelService {

private hotels = [

{ id: 1, name: 'Taj Hotel', country: 'India' },

{ id: 2, name: 'Trump', country: 'United State' },

{ id: 3, name: 'Hotel Oberoi', country: 'India' }

];

getHotels() {

return this.hotels;

}

getHotel(id: number) {

return this.hotels.find((h: any) => h.id === id);

}

updateHotel(id: number, hotelData: { name: string, country: string }) {

const hotel = this.hotels.find((h) => h.id === id);

if (hotel) {

hotel.name = hotelData.name;

hotel.country = hotelData.country;

}

}

}

In the hotels component, we have a list of all hotel records and router-outlet. As the hotels component itself is a child component, we need a nested router-outlet as a place where we render the child element of the hotels component. Edit hotels.component.html file

<table class="table table-striped">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Hotel</th>

<th scope="col">Country</th>

<th scope="col">Edit</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let hotel of hotels; index as i">

<th scope="row">{{ i + 1 }}</th>

<td>

{{ hotel.name }}

</td>

<td>{{ hotel.country }}</td>

<td>

<button class="btn btn-primary" [routerLink]="['/hotels/' + hotel.id]">Edit</button>

</td>

</tr>

</tbody>

</table>

<div class="col-xs-12 col-sm-4">

<router-outlet></router-outlet>

</div>

**Navigating route with params in a template.**

When the user clicks on one of the edit buttons of the hotel record. We will navigate to the hotel component and in the URL path, we will pass the hotel id. Where id is a dynamic value representing hotel id and is pass on navigation URL as params

<button [routerLink]="['/hotels/' + hotel.id]">Edit</button>

**Navigating route with params in a component typescript.**

When some operation is finished or the user clicked on some button and then we want to trigger the navigation from typescript code. In home.component.html we have a button, which on click will call a method in typescript component, which used a router object method to navigation. The router object has lots of methods, and navigation is the most used one. The navigation first parameter is a path of a component where we want to navigate.

In home.component.html

-----------------------

<button class="btn btn-primary" (click)="routeToHotelId(2)">

Navigation using router object in typescript</button>

</button>

In home.component.ts

---------------------

routeToHotelId(id: number) {

this.router.navigate(['/hotels/' + id]);

}

**How to retrieve query param from an activated route in a component.**

Once we navigate to a component, we have to retrieve query params from the activated route. Angular provides two approaches to retrieve query params at the navigate component.

* You can read the query parameter from the activatedroute snapshot
* The activatedRoute query parameters as an observable

import { HotelService } from './../hotel.service';

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

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

@Component({

selector: 'app-edit-hotel',

template: `

<div \*ngIf="hotel">

<div class="form-group">

<label for="name">Hotel Name</label>

<input type="text" id="name" class="form-control" [(ngModel)]="hotel.name">

</div>

<div class="form-group">

<label for="status">Country</label>

<select id="status" class="form-control" [(ngModel)]="hotel.country">

<option value="India">India</option>

<option value="United State">United State</option>

</select>

</div>

</div>`,

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

})

export class EditHotelComponent implements OnInit {

hotel: { id: number, name: string, country: string };

name = '';

country = '';

constructor(

private route: ActivatedRoute,

private hotelService: HotelService) { }

ngOnInit() {

let id = +this.route.snapshot.params['id'];

this.hotel = this.hotelService.getHotel(id);

this.route.params.subscribe((params: Params) => {

id = +params['id'];

this.hotel = this.hotelService.getHotel(id);

console.log('hotel' + this.hotel);

});

// this.route.paramMap.subscribe((params) => {

// id = +params.get('id');

// this.hotel = this.hotelService.getHotel(id);

// console.log('hotel' + this.hotel);

// });

}

}

The snapshot from the ActivatedRoute to read our parameters in the ngOnInit and use when we loaded component it from another component. If we need to load the same component with different parameters in the same component then it is recommended that we do not rely on the snapshot.

As snapshot call once and the only URL will update. Instead, we can treat the parameters and query parameters as observable, just like service calls and HTTP requests. This way, the subscription will get triggered each time the URL changes, allowing us to reload the data rather than relying on the snapshot.

**Note:**We have comments, instead of param observable, we can use router paramMap observable.

**Angular route navigate with paramter using queryParams property**

We can also pass optional query parameters using [queryParams] directive along with [routerLink] to pass query parameters in URL.  In URL queryparams with optional parameters, this is indicated by after question marks.

<button [routerLink]="['/hotels', hotel.id]"

[queryParams]="{by: 'edupala'}">

Country detail

</button>

// localhost:8100/hotels/2?by=edupala

// http://localhost:4200/countries/1?by=edupala.com

someMethod(hotel: IHotel) {

this.router.navigate(['/hotels', hotel.id],

{queryParams:{by:'edupala'}});

}

To retrieve optional parameters at routed component use the queryParams observable instead of the params observable of activatedRoute.

 ngOnInit() {

  this.route.queryParams.subscribe( (param.) => {

  let data = params.by;

  });

 }

**Conclusion**  
We have completed how-to pass data via the [Angular route](https://angular.io/guide/router). Use a snapshot to read data from the route object only when we navigate from another component. Snapshot is one time read, not recommend to load the same component with a different parameter. Instead, use params observable to get the latest update value of params from the route object.

Using ParamMap Observable For Handling Route In Angular

**Introduction**

In this article, we are going to learn how to use paramMap observable of ActivatedRoute and also learn how it is better to use that snapshot to read the parameter. So far we were using the snapshot to read the route parameter but using snapshot has several drawbacks. So, we will also see the difference of how the paramMap is better than the snapshot.

**prerequisites**:

* HTML, CSS and Js
* Basics of Typescript.
* [Route Parameter in angular](https://www.c-sharpcorner.com/article/overview-of-route-parameters-in-angular/)

Before we start please go through the creation of an application using a [route parameter in angular](https://www.c-sharpcorner.com/article/overview-of-route-parameters-in-angular/).

Let us modify some files and observe what is the drawback of using the snapshot to read route parameter and using the component.

Open student-list.component.ts and add the below contents:

1. **import** { Component, OnInit } from '@angular/core';
2. **import** { ActivatedRoute } from '@angular/router';
4. @Component({
5. selector: 'app-student-list',
6. templateUrl: './student-list.component.html',
7. styleUrls: ['./student-list.component.css']
8. })
9. **export** **class** StudentListComponent **implements** OnInit {
11. **public** student = {
12. id: "",
13. name: "",
14. marks: ""
15. };
16. **public** students = [
17. {"id" : 1001, "name" : "Irshad", "marks" : 90},
18. {"id" : 1002, "name" : "Imran", "marks" : 80},
19. {"id" : 1003, "name" : "Rahul", "marks" : 70},
20. {"id" : 1004, "name" : "Ajay", "marks" : 85},
21. {"id" : 1005, "name" : "Sunny", "marks" : 60}
22. ];
23. constructor(**private** route: ActivatedRoute) { }
25. ngOnInit() {
26. **this**.student.id = **this**.route.snapshot.paramMap.get('id');
27. **this**.student.name = **this**.route.snapshot.paramMap.get('name');
28. }
30. }

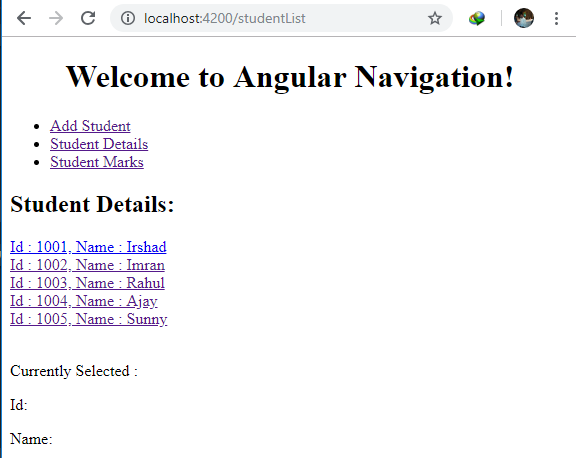
Open student-list.component.html and add the below contents:

1. <h2>Student Details:</h2>
2. <div \*ngFor="let stud of students">
3. <a routerLink="/studentList/{{stud.id}}/{{stud.name}}"> Id : {{stud.id}}, Name : {{stud.name}} </a>
4. </div>
5. <br>
6. <p>Currently Selected :</p>
8. <p>Id: {{student.id}}</p>
9. <p>Name: {{student.name}}</p>

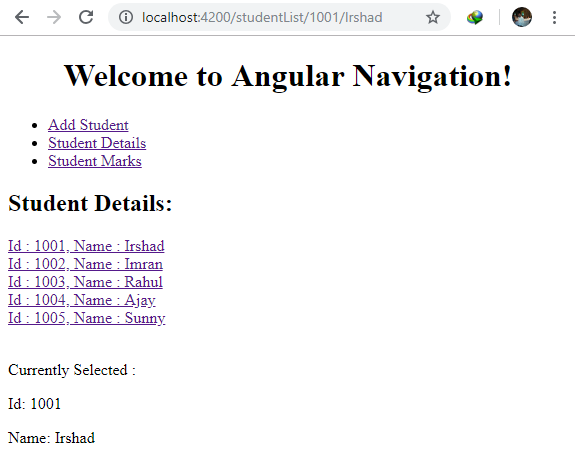
Open app-routing.module.ts and add the below values to the routes

1. **const** routes:Routes = [
2. {path: '', redirectTo : '/studentList', pathMatch : 'full'},
3. {path: 'studentList', component : StudentListComponent},
4. {path: 'studentList/:id/:name', component : StudentListComponent},
5. {path: 'studentMarks', component : StudentMarksComponent},
6. {path: 'addStudent', component : StudentComponent},
7. {path: "\*\*", component : NotFoundComponent}
8. ];

Run the application,



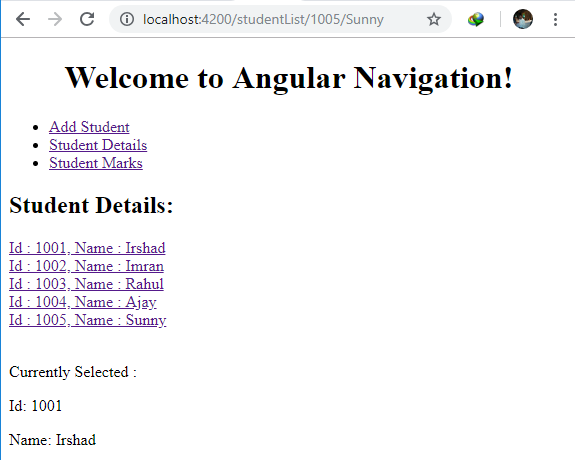
Click on any record (say ‘Irshad’) and see the URL and contents,



We have utilized this path to render the contents of the studentListComponent.

1. {path: 'studentList/:id/:name', component : StudentListComponent}

Now, again click on some other record (say ‘Sunny’) and see the URL and contents,



You can see the URL and the value has been changed but the rendered contents are still the same.

What we have observed so far is that the view is not getting updated. This is the drawback of using snapshot. The snapshot approach fails if we are navigating from one component back to the same component. The reason is that Angular needs to figure out whether it needs to initialize the new component or if it can reuse the same component. If we are navigating back to the same component. Angular reuses the same component. This is why our view is not getting updated.

The reason is the ngOnInit() method is not getting called and hence every time we are coming to the same component it simply reused the component and the value is not getting updated.

**paramMap Observable**

paramMap is the method of ActivatedRoute and it will return an observable and it will give data only if we subscribe to it. The argument in the subscribe method is an arrow function. The observable provides the parameter which we strongly typed to paramMap. paramMap is coming from Router package. We should have imported the Router package.

The paramMap provides us with the get method to retrieve the parameters.

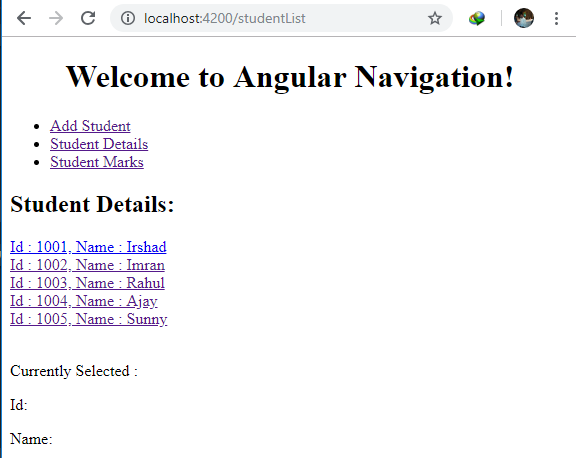
Let us start with the code for better understanding.

Open student-list.component.ts and change ngOnInit() method.

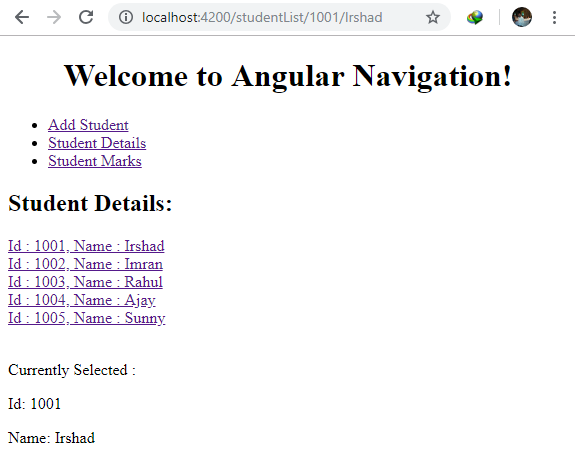
1. ngOnInit() {
2. **this**.route.paramMap.subscribe((params : ParamMap)=> {
3. **this**.student.id=params.get('id');
4. **this**.student.name=params.get('name');
5. });

Now, anytime the parameter value changes and even the navigation is being done on the same component, paramMap observable will detect and read it.

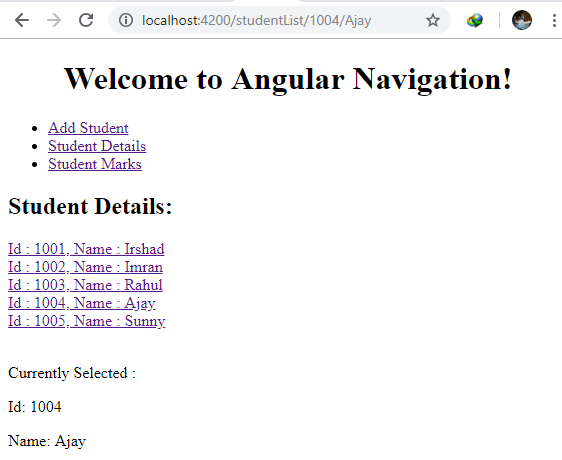
Save the files and run the application.



Click on record (‘irshad’)



Click on any other record ( say ‘ajay’)



Great, we are getting the updated value even on navigation on the same component. You can see the URL value is getting changed and also making the view updated. Clicking on any record will give you the updated view.

So, this is how you can read the route parameter value using the param observable.