The Angular 9/8 Router: Route Parameters with Snapshot and ParamMap by Example

In this tutorial we're going to see how to handle route parameters with the Angular 9/8 Router example using different methods: Snapshot and ParamMap Observable.

Angular provides a powerful router library that allows developers to implement advanced functionality in their Angular applications, beside basic component routing, such as:

* Route protection using guards,
* Route parameters,
* Child routes,
* Auxiliary routes etc.

**Handling Route Parameters with Angular 9**

In the previous [tutorial](https://www.techiediaries.com/angular-router), we have created a basic routing between components with the Angular Router. In this tutorial we're going to see how to handle route parameters in Angular 9.

We are going to start from the simple Angular application we've build in the previous tutorial which you can find from this [repository](https://github.com/techiediaries/angular-router-demo) or in [CodeSandBox](https://codesandbox.io/s/github/techiediaries/angular-router-demo)

This is the implementation of the **ProductDetailComponent:**

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

import { Product } from "../models/product";

@Component({

selector: "product-detail",

templateUrl: "./product-detail.component.html",

styleUrls: []

})

export class ProductDetailComponent implements OnInit {

public products: Product[] = [

new Product(1, "Product 001"),

new Product(2, "Product 002"),

new Product(3, "Product 003"),

new Product(4, "Product 004"),

new Product(5, "Product 005"),

new Product(6, "Product 006"),

new Product(7, "Product 007"),

new Product(8, "Product 008")

];

product: Product = this.products[0];// this will store the current product to display

constructor() {}

ngOnInit() {

}

}

**An Angular Route with a Parameter Example**

In the previous tutorial, we've added this route object in our router configuration:

{ path: "product/:id", component: ProductDetailComponent }

The :id placeholder (called dynamic router parameter) means that the ProductDetailComponent component will be activated when the user visits any path that matches this expression: /product/[0-9|a-b|A-B]+.

The Angular Router will also allow you to retrieve the value of :id from the activated component (i.e in this case ProductDetailComponent ) so let's see how?

**How to Get Route Parameters**

The Angular Router provides two different methods to get route parameters:

* Using the route snapshot,
* Using Router Observables

**Navigation Using The RouterLink Directive with Parameters**

Open src/app/product-list/product-list.component.html then change the list of products to use anchor tags with the routerLink directive to be able to navigate the ProductDetailComponent component.

<h1>Products List</h1>

<ul>

<li \*ngFor="let product of products">

<a [routerLink]="['/product',product.id]"></a>

</li>

</ul>

You can also create links using:

<a routerLink="/product/"></a>

After creating the links with parameters. We can now proceed to see how we can retrieve the id parameter from the URL in the ProductDetailComponent.

The Angular Router provides the [ActivatedRoute](https://angular.io/api/router/ActivatedRoute) class that can be injected in the component. So first start by importing it using:

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

Next, you need to inject this class in the component via the constructor:

constructor(private route: ActivatedRoute) {}

In the ngOnInit life-cycle method of the component we'll add the necessary code to grab the route parameter by subscribing to the params map or paramMap (instance of [ParamMap](https://angular.io/api/router/ParamMap), available only in Angular 4+) of the injected instance:

ngOnInit() {

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

this.products.forEach((p: Product) => {

if (p.id == params.id) {

this.product = p;

}

});

});

}

After subscribing to the paramMap we grab the parameters by their names in the path object (i.e in our case id) params.id. The rest of the code is just iterating over the products array to find the corresponding product to display via its *id*.

You can also use the *snapshot* object of the ActivatedRoute instance: this.route.snapshot.params.id

ngOnInit() {

this.products.forEach((p: Product) => {

if (p.id == this.route.snapshot.params.id) {

this.product = p;

}

});

}

**Conclusion**

The Angular Router allows you to easily retrieve parameters from the URL which is an essential functionality that's required by most web applications. You can use both ways: the paramMap observable or the snapshot way but the latter requires you to be careful when re-using components. You can find the code in this [repository](https://github.com/techiediaries/angular-router-demo).