

Rotas Com Parâmetros

É muito comum vermos que algumas URLs de sites utilizam parâmetros. Esses parâmetros são dados que podemos recuperar para podermos utilizar para realizar alguma pesquisa ou qualquer outra lógica que acharmos necessário. Pensando nisso, o Angular também possui sua maneira de criar rotas com parâmetros.

Para testarmos como funciona a criação de rotas com parâmetros, vamos utilizar a nossa aplicação anterior com duas páginas (page1 e page2). Vamos fazer com que, para acessar a page2, nós precisamos passar um parâmetro que receberá um **id**, por exemplo. Para isso, basta que, depois da nomeação da sua rota, você utilize **/:nomeDoParâmetro**. Com essa nomenclatura, o Angular vai entender que um parâmetro deve ser passado nessa URL para que seja possível acessá-la.

```
1 import { NgModule } from '@angular/core';
2
3 import { RouterModule, Routes } from '@angular/router';
4 import { Page1Component } from './pages/page1/page1.component';
5 import { Page2Component } from './pages/page2/page2.component';
6
7 const appRoutes: Routes = [
8   {
9     path: '',
10    pathMatch: 'full',
11    component: Page1Component
12  },
13  {
14    path: 'page2/:id',
15    component: Page2Component
16  }
17 ]
18
19 @NgModule({
20   declarations: [],
21   imports: [
22     RouterModule.forRoot(appRoutes)
23   ],
24   exports: [
25     RouterModule
26   ],
27   providers: [],
28 })
29 export class AppRoutingModule { }
30
```

O nome dado ao parâmetro é **id**. Mas você pode colocar o nome que desejar. Você também pode colocar vários parâmetros em uma rota. Basta utilizar a mesma nomenclatura quantas vezes achar necessário.

```
1 import { NgModule } from '@angular/core';
2
3 import { RouterModule, Routes } from '@angular/router';
4 import { Page1Component } from '../pages/page1/page1.component';
5 import { Page2Component } from '../pages/page2/page2.component';
6
7 const appRoutes: Routes = [
8   {
9     path: '',
10    pathMatch: 'full',
11    component: Page1Component
12  },
13  {
14    path: 'page2/:id/:para2/:para3',
15    component: Page2Component
16  }
17 ]
18
19 @NgModule({
20   declarations: [],
21   imports: [
22     RouterModule.forRoot(appRoutes)
23   ],
24   exports: [
25     RouterModule
26   ],
27   providers: [],
28 })
29 export class AppRoutingModule { }
30
```

Assim, estaremos informando que, para acessar essa rota, precisaremos informar 3 parâmetros na URL. **LEMBRE-SE**, esses parâmetros são obrigatórios. Você deve passar todos para acessar a rota (nesse caso, três parâmetros). Caso você tente acessar essa rota sem passar os parâmetros, ocasionará em erro.

Agora, para testarmos, precisaremos alterar o valor do nosso routerLink, informando agora os parâmetros necessários. Nesse caso, três parâmetros.



```
1  <!-- template HTML do AppComponent -->
2  <ul>
3    <li>
4      <a routerLink="/">Página 1</a>
5    </li>
6    <li>
7      <a routerLink="/page2/15/val2/val3">Página 2</a>
8    </li>
9  </ul>
10 <router-outlet></router-outlet>
```

Agora, com o valor do `routerLink` alterado, passando os parâmetros necessários, será possível acessar a rota. Caso você teste, você verá que estará funcionando.

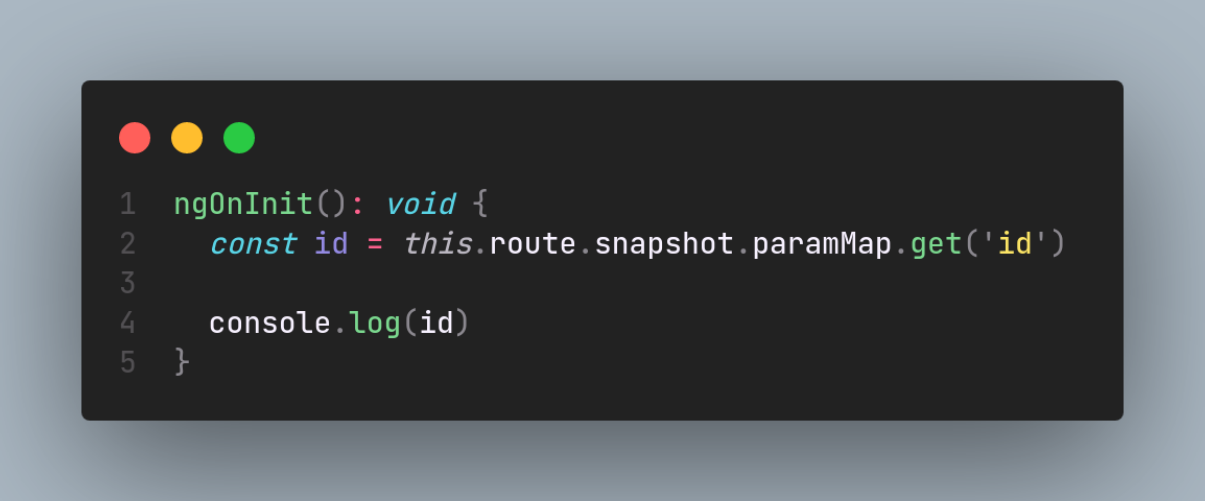
Com os parâmetros informados, nós precisamos recuperá-los para poderem ser utilizados de alguma maneira dentro do componente da rota. Para acessarmos, dentro da classe do componente, iremos injetar um objeto do **ActivatedRoute**. Esse objeto possui dados da rota ativa, como os parâmetros que foram enviados.



```
1  import { Component, OnInit } from '@angular/core';
2  import { ActivatedRoute } from '@angular/router';
3
4  @Component({
5    selector: 'app-page2',
6    templateUrl: './page2.component.html',
7    styleUrls: ['./page2.component.css']
8  })
9  export class Page2Component implements OnInit {
10
11    constructor(
12      private route: ActivatedRoute
13    ) { }
14
15    ngOnInit(): void {
16
17    }
18  }
19
```

Agora, para testarmos, dentro do lifecycle `ngOnInit`, vamos recuperar estes parâmetros. Para recuperarmos devemos usar o método **`get()`**, que está dentro do objeto **`paramMap`**, está dentro do objeto **`snapshot`**. Esse método recebe uma string com o nome do parâmetro que você deseja recuperar.

AVISO: Qualquer valor de parâmetro informado é retornado como string. Caso queira manipulá-lo de outra maneira, você deve convertê-lo.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains a TypeScript code snippet for the `ngOnInit` method. The code is as follows:

```
1  ngOnInit(): void {  
2    const id = this.route.snapshot.paramMap.get('id')  
3  
4    console.log(id)  
5  }
```

Neste caso, estamos recuperando somente o parâmetro com o nome *id*. Será retornado para a variável o valor do parâmetro **`id`**. Caso o parâmetro não existisse, seria retornado o valor **`null`**.

Agora, com o valor recuperado, nós poderíamos, por exemplo, fazer uma requisição a uma api, caso precisássemos, ou qualquer outra lógica que seja necessária com os parâmetros.