

Roteamento com Angular

Em aplicações feitas com frameworks, como o Angular, o roteamento entre páginas não é feito como em um site normal feito apenas com as tecnologias base (HTML, CSS e Javascript).

Como estamos trabalhando com uma Single Page Application (SPA), significa que nossa aplicação não possui nada mais além de uma página, logo, não navegamos em mais de um documento HTML. Tudo que é mostrado na tela ocorre somente em um único arquivo HTML onde, graças ao Javascript, a página torna-se dinâmica, adicionando ou removendo elementos na tela.

A partir dessa dinamicidade que o Javascript traz para a aplicação, nós conseguimos fazer um roteamento sem precisar de outros arquivos HTML, evitando com que seja necessário fazer diversas requisições ao servidor pedindo para que ele traga uma nova página sempre que necessário, fazendo com que esse trabalho seja feito no lado do cliente utilizando Javascript para isso.

Em resumo, para trabalhar com roteamento de páginas em uma SPA, dentro do Angular, utilizamos o módulo chamado **RouterModule**. O RouterModule é o módulo responsável por fazer o roteamento entre uma página e outra.

Primeiros Passos

Quando trabalhamos com roteamento em uma aplicação, nós criamos módulos específicos que contêm toda a lógica de roteamento. Vamos criar um módulo de roteamento para o módulo principal da nossa aplicação (AppModule). Esse novo módulo se chamará **AppRoutingModule** e será importado no módulo principal.



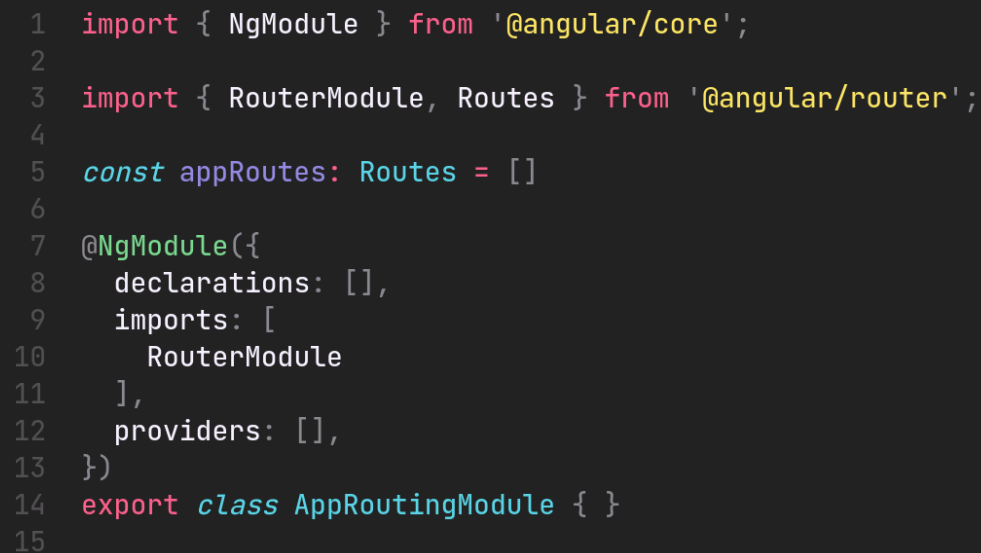
```
1 import { NgModule } from '@angular/core';
2
3 @NgModule({
4   declarations: [],
5   imports: [],
6   providers: [],
7 })
8 export class AppRoutingModule { }
9
```

Dentro desse módulo, nós chamaremos o RouterModule



```
1 import { NgModule } from '@angular/core';
2
3 import { RouterModule } from '@angular/router';
4
5
6 @NgModule({
7   declarations: [],
8   imports: [
9     RouterModule
10  ],
11   providers: [],
12 })
13 export class AppRoutingModule { }
14
```

A partir disso, precisamos declarar nossas rotas para que seja possível fazermos o roteamento. Para isso, criaremos uma variável do tipo **Routes**, onde guardaremos nossas rotas.



```
1 import { NgModule } from '@angular/core';
2
3 import { RouterModule, Routes } from '@angular/router';
4
5 const appRoutes: Routes = []
6
7 @NgModule({
8   declarations: [],
9   imports: [
10     RouterModule
11   ],
12   providers: [],
13 })
14 export class AppRoutingModule { }
15
```

Dentro desse Array, nós vamos declarar nossas rotas. Cada rota dentro desse Array é um objeto que possui algumas propriedades. Em suma, cada rota faz referência a um componente. Quando uma determinada rota for acessada, esse componente será mostrado.

Vamos criar componentes dentro de uma pasta chamada **pages**, onde guardaremos os componentes que são rotas. Assim, teremos mais organização no nosso projeto. Vamos criar um componente chamado **Page1**, para testarmos.

Com o nosso componente criado, vamos criar uma rota para acessarmos esse componente dentro do AppRoutingModule

```

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

```

Um objeto **Route** precisa, no mínimo, de duas propriedades: a propriedade **path**, que informa qual o nome da rota que precisamos acessar. Quando a propriedade **path** possuir uma string vazia, significa que é a página principal e a propriedade **componente**. Ela informa qual componente deve ser exibido quando a rota for acessada. Nesse caso, exibiremos o nosso componente Page1.

A propriedade **pathMatch** é uma propriedade que é recomendada utilizar quando tivermos rotas com o path vazio. Quando colocamos o valor **full**, significa que o caminho deve ser idêntico ao informado. Desde o início da URL. Caso não seja, ele não entrará na rota desejada. Se essa propriedade não for adicionada, poderemos ter um loop infinito na sua aplicação, pois, basicamente, qualquer rota pode ser referenciada com um path vazio. Por isso, a propriedade **pathMatch** é importante ser utilizada nesse caso.

Agora, com a nossa primeira rota feita, precisamos fazer somente mais alguns passos. Na parte de importação do **RouterModule**, vamos executar o método **forRoot()** dele, passando as nossas rotas criadas. Esse método fará com que ele configure as rotas e as deixe preparadas para sua execução na aplicação. Além disso, precisaremos utilizar a propriedade **exports** do **NgModule**, para podermos exportar para outros módulos o **RouterModule**.

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

Agora, com tudo pronto, basta importar esse módulo no módulo principal e as rotas já estarão funcionando.

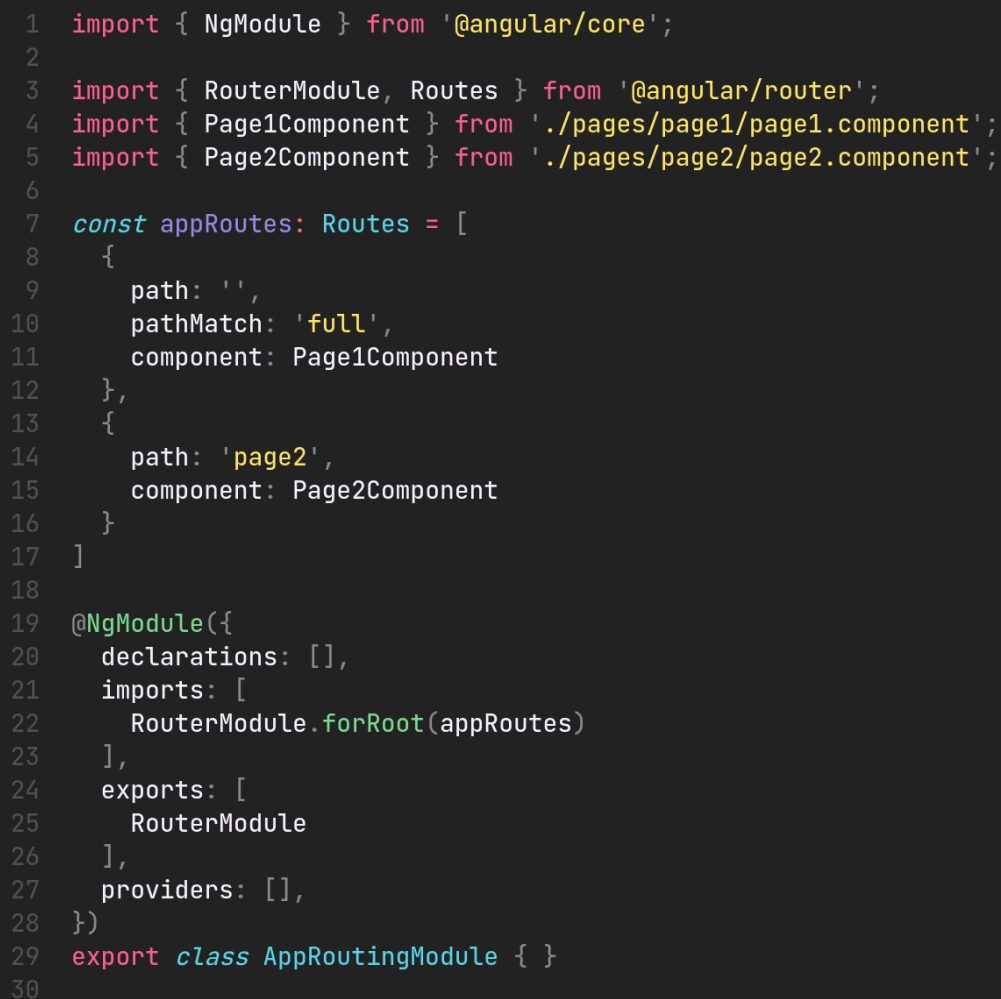
O último passo será, dentro do template HTML do componente principal (AppComponent), colocaremos no seu HTML um componente do RouterModule chamado **router-outlet**. Será esse componente responsável pela troca das páginas e mostrar os componentes de cada página.



```
1  <!-- template HTML do AppComponent -->
2
3  <router-outlet></router-outlet>
```

Somente com essa linha de código dentro do seu componente principal, o roteamento estará funcionando 100%. Abra seu navegador e veja que o conteúdo que você adicionou no componente Page1 está aparecendo na página inicial.

Para vermos o roteamento funcionando de verdade, precisamos criar outras rotas. Vamos criar um outro componente chamado Page2 e criar uma rota para ele. Ao criar o componente, seu módulo de rotas deve parecer com isso



```

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',
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

```

Repare que, na segunda rota, não utilizamos o `pathMatch`, pois o path dessa rota não está vazio. Então não precisaremos dessa vez.

Agora, dentro do nosso `AppComponent`, ou em qualquer outro componente que desejar, podemos criar uma lista de links para acessar as páginas que desejamos.

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

O primeiro **href**, com a rota vazia, se refere à rota principal da aplicação. O segundo, à rota **page2**. Agora, se você acessar a página 2, você verá que a alteração de componente foi feito.

Mas... você notou algo? A nossa página recarregou... 🤔. SPAs não devem possuir esse comportamento. Se mantermos esse comportamento, significa que estamos fazendo com que a aplicação faça outra requisição no servidor para recuperar a página e, aí sim, mostrar para o usuário. Esse comportamento ocorre devido à propriedade **href**. Quando trabalhamos com o roteamento no Angular, não podemos utilizar o atributo **href** dentro das tag **<a>**. Para rotearmos a aplicação, utilizamos o atributo **routerLink** no lugar do href. Assim, o comportamento de recarregamento não acontecerá


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

Se você perceber, agora a página não estará mais sendo recarregada. Isso, pelo fato de não estarmos mais utilizando o **href**, mas sim o **routerLink**.

Você reparou mais uma coisa...? 😞 Mesmo com a rota sendo alterada, a lista de links não some nunca! Por qual motivo isso acontece? 😞

Simples! O roteamento acontece dentro do componente **router-outlet**. Esse componente tem a única função de alterar o componente que deve ser mostrado a cada alteração da rota. Então, se algum conteúdo adicional for colocado junto a ele, ele não removerá esse conteúdo, ele apenas irá adicionar o conteúdo da rota ao template. Isso é útil para o reaproveitamento mais fácil do código.

Suponha que temos uma barra de navegação e ela vai se repetir em todas as páginas da nossa aplicação. Ao invés de repetir a chamada dessa barra de navegação dentro de cada componente/página da nossa aplicação, podemos chamar a barra de navegação uma única vez dentro do componente principal e a barra de navegação ficará lá em todas as páginas de uma maneira mais simples. A única coisa que irá ser alterada é o resto do conteúdo da página.

Rota Coringa

O usuário pode tentar algum recurso (rota) que pode não existir dentro da nossa aplicação. Quando o usuário tenta acessar esse tipo de rota, podemos utilizar uma rota coringa do Angular. Essa rota coringa permite nós mostrarmos um componente padrão mostrando uma

mensagem de erro ou qualquer coisa que desejar, informando que esse recurso não existe ou até redirecionar o usuário para uma outra página. Para utilizarmos essa rota coringa, basta que, no valor da propriedade **path**, coloquemos dois asteriscos (**). O Angular entende que estamos definindo uma rota coringa para quando o usuário acessar alguma rota que não existe na nossa aplicação. Vamos criar um componente chamado **ErrorPage** e vamos mostrá-lo quando ele acessar qualquer rota que não exista na nossa aplicação.

```
1 import { NgModule } from '@angular/core';
2
3 import { RouterModule, Routes } from '@angular/router';
4 import { ErrorPageComponent } from '../pages/error-page/error-page.component';
5 import { Page1Component } from '../pages/page1/page1.component';
6 import { Page2Component } from '../pages/page2/page2.component';
7
8 const appRoutes: Routes = [
9   {
10     path: '',
11     pathMatch: 'full',
12     component: Page1Component
13   },
14   {
15     path: 'page2',
16     component: Page2Component
17   },
18   {
19     path: '**',
20     component: ErrorPageComponent
21   }
22 ]
23
24 @NgModule({
25   declarations: [],
26   imports: [
27     RouterModule.forRoot(appRoutes)
28   ],
29   exports: [
30     RouterModule
31   ],
32   providers: [],
33 })
34 export class AppRoutingModule { }
35
```

Agora, toda vez que o usuário acessar uma página que não existe, como **/nao-existe**, esse componente com uma mensagem de erro irá aparecer. Nós também podemos mandar ele para outra rota. Basta utilizar a propriedade **redirectTo**, informando a rota que ele deve acessar.

```
1 import { NgModule } from '@angular/core';
2
3 import { RouterModule, Routes } from '@angular/router';
4 import { ErrorPageComponent } from '../pages/error-page/error-page.component';
5 import { Page1Component } from '../pages/page1/page1.component';
6 import { Page2Component } from '../pages/page2/page2.component';
7
8 const appRoutes: Routes = [
9   {
10     path: '',
11     pathMatch: 'full',
12     component: Page1Component
13   },
14   {
15     path: 'page2',
16     component: Page2Component
17   },
18   {
19     path: '**',
20     redirectTo: '/'
21   }
22 ]
23
24 @NgModule({
25   declarations: [],
26   imports: [
27     RouterModule.forRoot(appRoutes)
28   ],
29   exports: [
30     RouterModule
31   ],
32   providers: [],
33 })
34 export class AppRoutingModule { }
35
```

Nesse caso, ele será mandado para a página inicial.