Thiago S. Adriano
Enjoy your life
Nov 28, 2017 · 7 min read

Angular 5: Trabalhando com Rotas

A um tempo atrás eu criei um artigo falando sobre rotas com o Angular na versão 4. Depois de algum tempo trabalhando focado em Angular, e aproveitando a sua nova Versão 5, eu resolvi criar um outra artigo mais detalhado sobre rotas.

Bom, o objetivo desse artigo será passarmos por alguns cenários onde nós utilizamos rotas no nosso dia a dia. Todos os exemplos que serão demonstrados nesse artigo, foram pensando nas dificuldades que eu tive durante esse aprendizado e na documentação do Angular v5.

Para que possamos pular a etapa de criação de um novo projeto, eu criei um projeto Angular na versão 5 e subi ele no GitHub no seguinte link AngularV5.

Routing Module

Nosso primeiro passo será a criação do nosso arquivo de Rotas. Para isso, execute o seguinte trecho de código no seu terminal/CMD.

```
ng generate module app-routing --flat --module=app
```

No comando a cima nós estamos gerando um novo módulo chamado *AppModuleRouting* e passando os parâmetros: *-flat* para que ele seja

criado dentro da pasta *src/app* e o *-module=app* para registrarmos ele no nosso arquivo *AppModule*. Podemos ver abaixo o resultado da execução desse passo.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [
    CommonModule
  ],
```

AppRoutingModule

Agora nós precisamos fazer algumas alterações no nosso arquivo de Rotas. Como nós não iremos declarar componentes dentro do nosso arquivo de rotas, nós podemos deletar os seguintes módulos: *NgModule.declarations* e *CommonModule*. Iremos precisar importar dois novos pacotes: RoutingModule e Routes. Podemos ver abaixo o nosso arquivo atualizado.

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

@NgModule({
  exports: [ RouterModule ]
```

AppRoutingModule

Adicionando Rotas

Para que possamos adicionar uma rota, vamos criar um novo componente chamado *HomeComponent*. Para isso, execute o comando abaixo no seu terminal/CMD.

```
ng g c pages/home
```

Agora adicione o seguinte trecho de código no seu arquivo de rotas:

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

```
const routes: Routes = [

{ path: 'home', component: HomeComponent }

];
```

Quando nós trabalhamos com rotas, nós temos duas propriedades:

- *path*: string que desejamos para a nossa rota, no nosso exemplo estamos utilizando home.
- *component*: passamos o nome do *component* que desejamos adicionar para a nossa rota.

Nosso próximo passo será passarmos para o nosso RouterModule as nossas rotas. Para isso, adicione o seguinte trecho de código dentro de NgModule({}).

```
1   @NgModule({
2     exports: [ RouterModule ],
3     imports: [ RouterModule.forRoot(routes) ]
4   })
```

Podemos observar que na linha 3 estamos passando as nossas rotas para o RouterModule através do método forRoot e exportando elas no linha 2.

Caso tenha interesse em entender melhor sobre o método forRoot, segue uma referência retirada da documentação oficial do Angular:

> *The method is called* `forRoot()` *because you configure the router at the application's root level. The* `forRoot()` *method supplies the service providers and directives needed for routing, and performs the initial navigation based on the current browser URL.*

RouterOutlet

Nosso próximo passo será atualizarmos o nosso arquivo *app.component.html*, para que ele entenda as nossas rotas. Para isso, vamos atualizar ele com o código abaixo:

```
<h1>{{title}}</h1>
<router-outlet></router-outlet>
```

RouterOutlet é uma das diretivas de Router. Quando nós importamos `AppRoutingModule` em AppModule, ele nós permite adicioná-la dentro do arquivo HTML do nosso *AppComponent,* dessa forma nós podemos passar as nossas rotas para que ele possa interpreta-las e direcionar para o component correto.

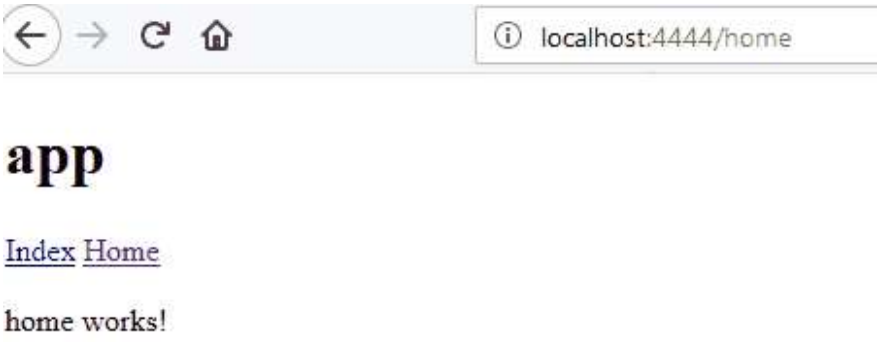
Quando nós criamos uma APP com o Angular Cli, ele já configura o AppComponent como root, essa configuração fica em bootstrap: [AppComponent] dentro do arquivo app-module.ts.

RouterLink

Para que possamos navegar pelas nossas rotas, nós podemos utilizar o routerlink de dentro de uma marcação HTML. Para que você possa entender melhor, atualize o seu arquivo *app.component.html* novamente com o código abaixo:

```
<nav>
<a routerLink="/">Index</a>
<a routerLink="/home">Home</a>
</nav>
```

Notem que é bem simples, para que ele funcione só precisamos passar a string junto de uma /, essa string deve ser uma das que nós configuramos no nosso arquivo *app.routing.module.ts*. Podemos ver abaixo o resultado dessa implementação.



Rota Default

Quando a nossa APP abre pela primeira vez, nós precisamos que uma rota seja seja carregada como default. Para que possamos entender

melhor, vamos criar um novo component chamado *AboutComponent*. Para isso, execute o seguinte comando no seu terminal/CMD.

```
ng g c pages/about
```

Com o nosso *component* criado, vamos atualizar o nosso arquivo de rotas:

```
1   import { NgModule } from '@angular/core';
2   import { RouterModule, Routes } from '@angular/router';
3   import { HomeComponent } from './pages/home/home.component
4   import { AboutComponent } from './pages/about/about.compon
5
6
7   const routes: Routes = [
8     { path: 'home', component: HomeComponent },
9     { path: 'about', component: AboutComponent },
10    { path: '', redirectTo: '/about', pathMatch: 'full' },
11
12  ];
13
```

Notem que na linha 09 criamos uma nova rota chamada about, em seguida nós passamos que quando a rota for algo como: localhost:porta ele deve fazer um redirect para /about. Dessa forma toda vez que a nossa App abrir ela irá direcionar para a nossa rota default, que nesse exemplo é a rota about.

Rota de Error404

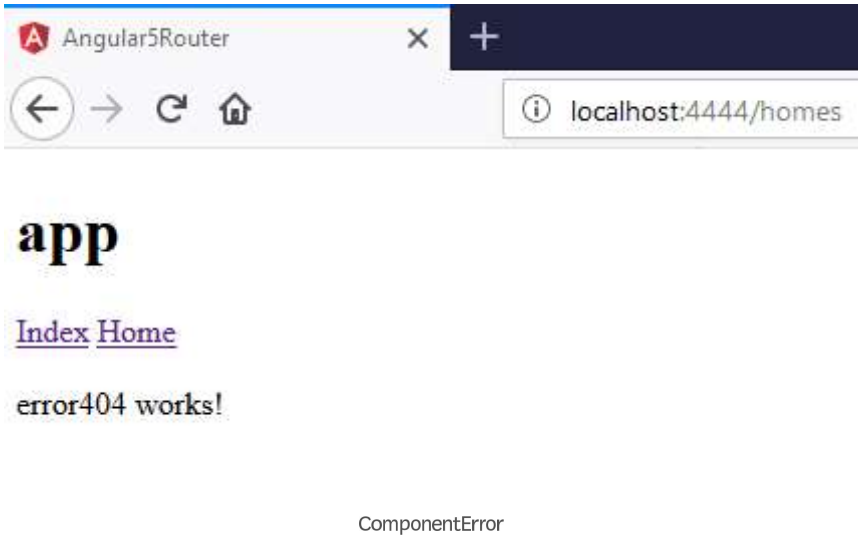
Para que o nosso usuário não seja direcionado para a nossa rota default toda vez que digitar uma rota errada, vamos criar uma nova rota de erro 404. Para isso, iremos precisar criar um novo component, execute o seguinte comando no seu terminal/CMD:

```
ng g c pages/error404
```

Agora adicione a nova rota no arquivo *app-routing.module.ts:*

```
{
   path: '**', component: Error404Component
}
```

Após atualizar o arquivo, tente digitar uma rota errada e note que você será direcionado para o nosso novo component de error404. Podemos ver esse resultado na imagem abaixo:



ComponentError

Passando parâmetros entre rotas

Vejamos agora como podemos passar dados de uma rota para outra. Para isso, vamos alterar os arquivos: *about.component.ts*, *app.component.html* e *app-routing.module.ts*.

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './pages/home/home.component
import { AboutComponent } from './pages/about/about.compon
import { Error404Component } from './pages/error404/error4


const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about/:id', component: AboutComponent },
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  {
    path: '**', component: Error404Component
  }

];
```

Nos alteramos para que home seja a nossa rota default e que about possa receber um parâmetro.

about.component.ts

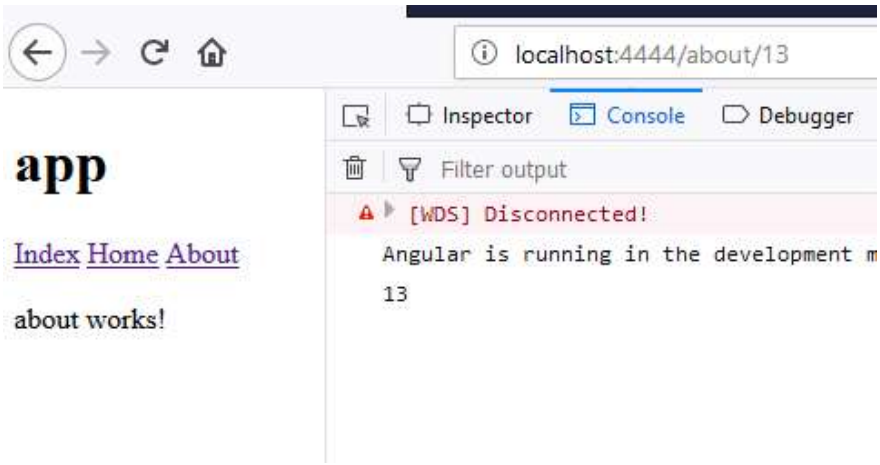
```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit {

  constructor(private route: ActivatedRoute) {
    this.route.params.subscribe(res => console.log(res.id)
  }
```

No código a cima nós estamos injetando o módulo ActivetedRoute no nosso component, e através do método route.params estamos fazendo um subscribe para que possamos receber os dados de um Observable.

app.component.html

```
<h1>{{title}}</h1>
<nav>
  <a routerLink="/">Index</a>
  <a routerLink="/home">Home</a>
  <a routerLink="/about/13">About</a>
```

Agora quando você clicar na rota about ela irá receber o parâmetro e apresentar no console do seu navegador. Podemos ver o resultado dessa implementação abaixo:



Navegando entre Componentes

Para que possamos navegar entre os nossos components sem a necessidade de criarmos os RouterLinks, como em uma validação de login onde temos os seguintes cenários:

- Usuário passou os dados corretos? Direcionar para XPTOComponent.
- Usuário passou dados inválidos? Direcionar ele para o formulário de login.

Nós podemos utilizar o método `navigate` do nosso módulo Router. Para que possamos entender melhor, vamos criar um novo component de login. Execute o seguinte comando no seu terminal/CMD:

```
ng g c pages/login
```

Nosso próximo passo será atualizar o nosso Modulo de gerenciamento de rotas, para que essa seja a nossa rota default.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './pages/home/home.component
import { AboutComponent } from './pages/about/about.compon
import { Error404Component } from './pages/error404/error4
import { LoginComponent } from './pages/login/login.compon


const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about/:id', component: AboutComponent },
  { path: 'login', component: LoginComponent },
  { path: '', redirectTo: '/login', pathMatch: 'full' },
  {
    path: '**', component: Error404Component
  },


```

Agora vamos atualizar os arquivos login.component.ts e login.component.html com os código abaixo:

login.component.ts

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

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  constructor(private router: Router) { }

  LogOn() {
    this.router.navigate(['home']);
```

login.component.html

```
Login:
<br>
<input type="text" value="Mickey">
<br> Password:
<br>
<input type="password" value="Mouse">
<br>
```

Podemos ver essa implementação na imagem abaixo:

Protegendo as Rotas

Vamos pensar em um cenário onde algumas das nossas rotas só possam ser acessadas por um grupo restrito de usuário que tiverem logado no nosso sistema. Para esse tipo de situação nós utilizamos Interface **CanActivate**. Para que possamos entender ela melhor, vamos restringir o acesso as nossas rotas: Home e About. Para isso, vamos atualizar o nosso arquivo de Rotas novamente:

```
1   import { NgModule } from '@angular/core';
2   import { RouterModule, Routes } from '@angular/router';
3   import { HomeComponent } from './pages/home/home.component
4   import { AboutComponent } from './pages/about/about.compon
5   import { Error404Component } from './pages/error404/error4
6   import { LoginComponent } from './pages/login/login.compon
7   import { AuthGuard } from './services/auth/auth.guard';
8
9
10  const routes: Routes = [
11    {
12      path: 'home', component: HomeComponent,
13      canActivate: [AuthGuard]
14    },
15    {
16      path: 'about/:id', component: AboutComponent,
17      canActivate: [AuthGuard]
18    },
19    {
20      path: 'login', component: LoginComponent
21    },
22    { path: '', redirectTo: '/home', pathMatch: 'full' },
23    {
```

Nós atualizamos as seguintes linhas do código a cima:

- 07: Nós adicionamos uma nova Class chamada AuthGuard que deve implementar a interface *CanActivate*.
- 13 e 17: Estamos passando que as nossas rotas são privadas e que a nossa class *AuthGuard* é responsável por gerenciar o acesso a elas.
- 22: estamos informando que a rota Home é a default.

Agora crie um novo arquivo chamado *auth.guard.ts* no seguinte caminho *src/app/services/auth/auth.guard.ts e* atualize ele com o seguinte trecho de código abaixo:

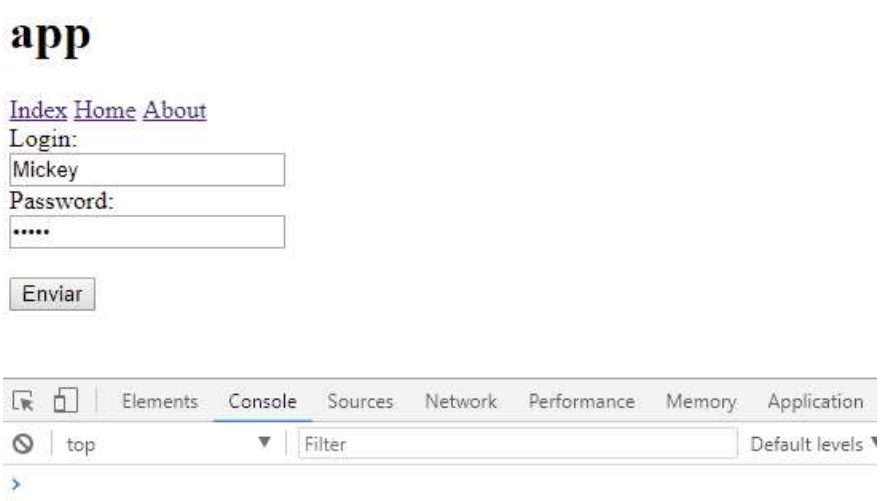
```
import { Observable } from 'rxjs/Observable';
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateS

@Injectable()
export class AuthGuard implements CanActivate {

    constructor(private router: Router) { }

    canActivate(
        route: ActivatedRouteSnapshot,
        state: RouterStateSnapshot): Observable<boolean>

        if (localStorage['token'] != null) {
```

Não irei entrar em detalhes sobre o código a cima, somente iremos entender a sua finalidade que é verificar se temos um token salvo no localStorage da nossa aplicação, caso sim ele permite o acesso a rota, caso não ele direciona para página de login.

Para que possamos testa-lo, adicione a linha de código abaixo no método logOn do seu LoginComponent.

```
localStorage['token'] = 'xptoh26410x5=50';
this.router.navigate(['home']);
```

Execute o seu código e note que quando não temos um token ele direciona para a nossa página de login, quando clicamos no botão enviar ele direciona para a nossa rota home permitindo o acesso a ela e a nossa rota About. Podemos ver essa implementação na imagem abaixo:



Bom, acredito que com isso nós finalizamos esse artigo, espero ter conseguido passar um pouco mais sobre como podemos trabalhar com rotas em projetos Angular. Caso você tenha interesse na versão final do código que nós utilizamos nesse artigo, segue o seu link no GitHub.