Aula 33 - SPA, Roteamento avançado e Injeção de Dependência

Docupedia Export

Author:Silio Leonardo (CtP/ETS)
Date:06-Jun-2023 15:49

Table of Contents

- Roteamento Avançado e SPA
- Roteamento Aninhado
- Titulos das Páginas
- Usando injeção de dependência
- ActivatedRoute
- Objetos injetáveis costumizáveis

1 Roteamento Avançado e SPA

SPA ou Single-Page Application é uma aplicação Web que possui apenas uma única página. Isso signiifca que tudo que acontece ocorre sobre um único HTML. O Angular em si é um SPA. Isso significa que todas as páginas renderizadas são uma única página que altera seu conteúdo constantemente. Para isso serve o sistemas de rotas: Ao alterar a URL podemos mudar a renderização e se basear em uma aplicação de única página.
Já vimos antes como escolher o componente renderizado baseado na rota na Aula 11 e usamos bastante na aula 12:

app-routing.module.ts

```
1  const routes: Routes = [
2    { path: "", component: HomePageComponent },
3    { path: "login", component: LoginPageComponent },
4    { path: "feed", component: FeedPageComponent },
5    { path: "comunity", component: ComunityPageComponent },
6    { path: "newaccount", component: NewAccountPageComponent},
7    { path: "recover", component: RecoverPageComponent},
8    { path: "user", component: UserPageComponent},
9    { path: "**", component: NotFoundPageComponent }
10 ];
```

Vimos que o "**" permite que nós criemos uma página de NotFound e ainda aprendemos a usar tags 'a' para redirecionar para outras páginas. Vamos fazer nossas SPA usando apenas o roteamento do Angular.

2 Roteamento Aninhado

No Angular podemos usar roteamento aninhado:

app-routing.module.ts

```
const routes: Routes = [
  { path: "", component: HomePageComponent },
  {
    path: "login",
    component: LoginPageComponent,
    children: [
      { path: "newaccount", component: NewAccountPageComponent }
    ]
  },
  { path: "feed", component: FeedPageComponent },
  { path: "comunity", component: ComunityPageComponent },
  { path: "recover", component: RecoverPageComponent },
  { path: "user", component: UserPageComponent },
  { path: "**", component: NotFoundPageComponent }
];
```

Aqui dizemos que a nova conta é uma rota interna do login. Isso nos permite fazer o seguinte:

login-page.component.html

```
<h1>
    Login
</h1>

<p>
    <label>Email/Username</label>
    <br>
    <input>
</p>

<app-password [seePassword]="false" [breakLineOnInput]="true" />

```

```
13  <p>
14      <button>Logar</button>
15  </p>
16
17  <p>
18      Não possui conta? <a routerLink="newaccount">Crie uma agora mesmo!</a>
19  </p>
20
21  <p>
22      Esqueceu sua senha? <a href="/recover">Recupere agora!</a>
23  </p>
24
25  <router-outlet></router-outlet>
```

Usando routerLink ao invés de href sem a barra indicamos querer acessar a url /login/newaccount que renderizará no lugar do "router-outlet". Isso é prefeito para fazer as mais variáveis páginas.

3 Titulos das Páginas

É possível também

app-routing.module.ts

```
const routes: Routes = [
  { path: "", title: "Rede Social Minimalista", component: HomePageComponent },
  {
    path: "login",
    title: "Autentificação",
    component: LoginPageComponent,
    children: [
      { path: "newaccount", component: NewAccountPageComponent }
    ]
  },
  { path: "feed", title: "Feed", component: FeedPageComponent },
  { path: "comunity", title: "Comunidades", component: ComunityPageComponent },
  { path: "recover", title: "Recuperar Senha", component: RecoverPageComponent },
  { path: "user", title: "Página de Usuário", component: UserPageComponent },
  { path: "**", title: "Not Found", component: NotFoundPageComponent }
];
```

4 Usando injeção de dependência

Injeção de dependência acontece, no Angular, quando o framework cria um objeto para nós que podemos requisitar no construtor de um componente. Ou seja, não criamos e configuramos um objeto, mas esperamos recebê-lo no construtor. Você verá um exemplo abaixo, onde usamos a injeção de dependência para conseguir um objeto que controla as rotas no angular.

5 ActivatedRoute

Desejamos que, caso o usuário decida recuperar sua senha e tenha deixado o email na sua tentativa de login, a página de recuperar senha abra com o email já definido para, supostamente, enviar um email de recuperação de senha. Podemos passar informações entre as telas da seguinte forma:

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { ComunityPageComponent } from './comunity-page/comunity-page.component';
import { FeedPageComponent } from './feed-page/feed-page.component';
import { HomePageComponent } from './home-page/home-page.component';
import { LoginPageComponent } from './login-page/login-page.component';
import { NewAccountPageComponent } from './new-account-page/new-account-page.component';
import { NotFoundPageComponent } from './not-found-page/not-found-page.component';
import { RecoverPageComponent } from './recover-page/recover-page.component';
import { UserPageComponent } from './user-page/user-page.component';

const routes: Routes = [
  { path: "", title: "Rede Social Minimalista", component: HomePageComponent },
  {
    path: "login",
    title: "Autentificação",
    component: LoginPageComponent,
    children: [
      { path: "newaccount", component: NewAccountPageComponent }
    ]
  },
  { path: "feed", title: "Feed", component: FeedPageComponent },
  { path: "comunity", title: "Comunidades", component: ComunityPageComponent },
  { path: "recover/:email", title: "Recuperar Senha", component: RecoverPageComponent }, // Agora podemos mandar recover/valor para mandar um parâmetro para rota
  { path: "recover", title: "Recuperar Senha", component: RecoverPageComponent }, // Podemos fazer rotas com e sem parâmetros
  { path: "user", title: "Página de Usuário", component: UserPageComponent },
  { path: "**", title: "Not Found", component: NotFoundPageComponent }
];

```

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Agora vamos alterar a tela de login para permitir que essa rota seja atingida:

login-page.component.ts

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

@Component({
  selector: 'app-login-page',
  templateUrl: './login-page.component.html',
  styleUrls: ['./login-page.component.css']
})
export class LoginPageComponent {
  email = ""
  link = ''
}
```

login-page.component.html

```
<h1>
    Login
</h1>

<p>
    <label>Email/Username</label>
    <br>
    <input [(ngModel)]="email">
</p>

<app-password [seePassword]="false" [breakLineOnInput]="true" />

<p>
```

```
    <button>Logar</button>
</p>

<p>
    Não possui conta? <a routerLink="newaccount">Crie uma agora mesmo!</a>
</p>

<p>
    Esqueceu sua senha? <a href="{{'/recover/' + email}}">Recupere agora!</a>
</p>

<router-outlet></router-outlet>
```

recover-page.component.html

```
<p>
    Digite seu email para recuperar a senha:
</p>

<input value="{{email}}"/>

<button>Enviar Email</button>
```

recover-page.component.ts

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

@Component({
  selector: 'app-recover-page',
  templateUrl: './recover-page.component.html',
  styleUrls: ['./recover-page.component.css']
})
export class RecoverPageComponent implements OnInit, OnDestroy {
  email = "";
  subscription: any;
```

```
12
13    constructor(private route: ActivatedRoute) { }
14
15    ngOnInit() {
16      this.subscription = this.route.params.subscribe(params => {
17         this.email = params['email'];
18      });
19    }
20
21    ngOnDestroy() {
22      this.subscription.unsubscribe();
23    }
24  }
```

Outro objeto interessante que podemos obter com a injeção de dependências é o Router. Ele permite que controlemos a rota ao invés de apenas obter dados dela:

recover-page.component.html

```
1  <p>
2      Digite seu email para recuperar a senha:
3  </p>
4
5  <input value="{{email}}"/>
6
7  <button (click)="send()">Enviar Email</button>
```

recover-page.component.ts

```
1  import { Component, OnInit, OnDestroy } from '@angular/core';
2  import { ActivatedRoute, Router } from '@angular/router'; // Adicionamos o Router aqui
3
4  @Component({
5    selector: 'app-recover-page',
6    templateUrl: './recover-page.component.html',
7    styleUrls: ['./recover-page.component.css']
8  })
9  export class RecoverPageComponent implements OnInit, OnDestroy {
```

```
10    email = "";
11    subscription: any;
12
13    constructor(private route: ActivatedRoute, private router: Router) { } // E aqui
14
15    ngOnInit() {
16      this.subscription = this.route.params.subscribe(params => {
17         this.email = params['email'];
18      });
19    }
20
21    ngOnDestroy() {
22      this.subscription.unsubscribe();
23    }
24
25    send() {
26      // Send Email Here
27      this.router.navigate(["/login"]) // E usamos para redirecionar aqui
28    }
29  }
```

6 Objetos injetáveis costumizáveis

Podemos fazer nossos próprios objetos injetáveis para criar serviços. Isso é importante e será muito usado no desenvolvimento backend também. Vamos criar uma pasta de serviços e então trabalhar em cima disso:

```
cd .\src\
cd .\app\
mkdir cep-service
ni cep-data.ts
cd ..
cd ..
npm run ng generate service cep-service/cep
```

app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NavComponent } from './nav/nav.component';
import { LoginPageComponent } from './login-page/login-page.component';
import { HomePageComponent } from './home-page/home-page.component';
import { NotFoundPageComponent } from './not-found-page/not-found-page.component';
import { FeedPageComponent } from './feed-page/feed-page.component';
import { ComunityPageComponent } from './comunity-page/comunity-page.component';
import { NewAccountPageComponent } from './new-account-page/new-account-page.component';
import { RecoverPageComponent } from './recover-page/recover-page.component';
import { UserPageComponent } from './user-page/user-page.component';
import { PasswordComponent } from './password/password.component';
import { FormsModule } from '@angular/forms';
import { CreatePasswordComponent } from './create-password/create-password.component'; // Added for use ngModel
import { HttpClientModule } from '@angular/common/http'; // Added for use HttpClient

@NgModule({
  declarations: [
    AppComponent,
    NavComponent,
```

```
    LoginPageComponent,
    HomePageComponent,
    NotFoundPageComponent,
    FeedPageComponent,
    ComunityPageComponent,
    NewAccountPageComponent,
    RecoverPageComponent,
    UserPageComponent,
    PasswordComponent,
    CreatePasswordComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule, // Added for use ngModel
    HttpClientModule // Added for use HttpClient
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

cep-data.ts

```
export interface CepData
{
    cep: string;
    logradouro: string;
    complemento: string;
    bairro: string;
    localidade: string;
    uf: string;
    ibge: string;
    gia: string;
    ddd: string;
    siafi: string;
}
```

cep.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { CepData } from './cep-data'

@Injectable({
  providedIn: 'root'
})
export class CepService {

  constructor(private http: HttpClient) { }

  getStreet(cep: string)
  {
    return this.http.get<CepData>("https://viacep.com.br/ws/" + cep + "/json/");
  }
}
```

new-account-page.component.html

```
<h1>
    Nova Conta
</h1>

<p>
    <label>Email</label>
    <br>
    <input>
</p>

<p>
    <label>CEP</label>
    <br>
    <input [(ngModel)]="cepvalue" (change)="cepAdded()">
</p>
```

```
16
17  <p>
18      <label>Rua</label>
19      <br>
20      <input [(ngModel)]="ruavalue">
21  </p>
22
23  <p>
24      <label>Username</label>
25      <br>
26      <input>
27  </p>
28
29  <app-create-password/>
30
31  <p>
32      <button>Criar Conta</button>
33  </p>
```

new-account-page.component.ts

```
1   import { Component } from '@angular/core';
2   import { CepService } from '../services/cep.service';
3
4   @Component({
5     selector: 'app-new-account-page',
6     templateUrl: './new-account-page.component.html',
7     styleUrls: ['./new-account-page.component.css']
8   })
9   export class NewAccountPageComponent {
10    cepvalue = ""
11    ruavalue = ""
12
13    constructor(private cep: CepService) { }
14
15    cepAdded()
16    {
17      this.cep.getStreet(this.cepvalue)
```

```
18          .subscribe(x =>
19          {
20            this.ruavalue = x.logradouro
21          })
22      }
23  }
```

Com esse serviço criamos um sistema de acesso ao CEP. Usamos a classe HttpClient para acessar um serviço de CEP para obter a rua de um CEP.