Aula 32 - Componentes, Templates e Binding em Angular

Docupedia Export

Author:Silio Leonardo (CtP/ETS)
Date:05-Jun-2023 16:15

Table of Contents

- Fazendo um projeto um pouco mais complexo
- Fazendo um componente complexo
- Reutilização de Componentes, Gerênciamento de dados e ngFor
- Exercícios

1 Fazendo um projeto um pouco mais complexo

Nesta aula compreenderemos profundamente os componentes, templates e binding. Para isso vamos mostrar um projeto simples e ainda incompleto onde usamos tudo que já sabemos e nos aprofundamos no que tange a componentes.

app.component.html

```
<header>
    <h1>Rede Social Minimalista</h1>
</header>

<nav>
    <app-nav></app-nav>
</nav>

<main>
    <router-outlet></router-outlet>
</main>

<footer>
    Todos os direitos reservados
</footer>
```

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: "", component: HomePageComponent },
  { path: "login", component: LoginPageComponent },
  { path: "feed", component: FeedPageComponent },
  { path: "comunity", component: ComunityPageComponent },
  { path: "newaccount", component: NewAccountPageComponent},
  { path: "recover", component: RecoverPageComponent},
  { path: "user", component: UserPageComponent},
  { path: "**", component: NotFoundPageComponent }
];

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

nav.component.html

```
<ul class="loginBox">
    <li>
        <a href="/">Home</a>
    </li>
    <li>
        <a href="/">Feed</a>
    </li>
    <li>
        <a href="/">Comunidades</a>
    </li>
    <li>
        <a href="/">Login</a>
    </li>
</ul>
```

nav.component.css

```
.loginBox {
```

```
    width: 100%;
    display: flex;
    flex-direction: row;
    list-style-type: none;
    align-content: center;
    align-items: center;
    justify-content: space-evenly;
    background-color: lightgray;
    height: 2rem;
}

.loginBox a {
    padding: 0.5rem;
}

.loginBox a:hover {
    background-color: gray;
    color: white;
}
```

not-found-page.component.html

```
<div class="not-found-box">
    <h1>Not Found</h1>
    <img src="https://media.makeameme.org/created/file-not-found-c17b083c9c.jpg">
</div>
```

not-found-page.component.css

```
.not-found-box {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

home-page.component.html

```
<marquee>
    <h2>
        <a href="/newaccount">Venha fazer parte a nossa Comunidade</a>
    </h2>
</marquee>
```

Essa simples rede social mostrará como os componentes podem ser utilizados. Vamos priorizar a estrutura antes de começar a usar Bootstrap entre outras tecnologias para tornar nosso sistema bonito. Assim implementaremos, nessa aula, as seguintes páginas:

- Login
- NewAccount
- Comunity
- Feed
- Recover (recuperar senha)

Iniciando pelo Login, podemos propor um componente que controle o fluxo de senha.

2 Fazendo um componente complexo

Agora vamos fazer um componente complexo com vários tipos de mecânicas do Angular como two-way binding entre outros tipos, eventos de clico de vida como ngOnInit, Inputs e Outputs e trabalhar mais com eventos.

password.component.html

```
<div>

    <p>
        <label>Senha</label>
        <!-- A linha só será pulada se breakLineOnInput for verdadeiro -->
        <br *ngIf="breakLineOnInput">
        <!--
            ngModel permite que façamos um two-way bind. Isso é, ao alterar uma variável
            na classe alteramos na tela e ao alterarmos na tela (por meio da ação do usuário)
            alteramos na classe também
            Outro fator importante é que chamamos eventos como click e change ao invés de onclick ou onchange.
            <input type={{inputType}}> é, em geral equivalente a <input [type]="inputType">
        -->
        <input type={{inputType}} [style]=inputStyle (focusout)="passwordFocusout()"
            [(ngModel)]="inputText" (keydown)="passwordChanged()" (change)="passwordChanged()" (click)="passwordClick()">
    </p>

    <!-- Este p e tudo que está dentro dele só aparecerá na tela se canSeePassword for verdadeiro -->
    <p *ngIf="canSeePassword">
        <!--
            Porquê não usar click aqui? Bem, o evento click é chamado antes que o Model seja atualizado.
            Assim seePassword apresentará o valor incorreto!
         -->
        <input type="checkbox" [(ngModel)]="seePassword" (change)="checkBoxToogle($event)">
        <label>Mostrar Senha</label>
    </p>

</div>
```

password.component.ts

```ts
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';

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

  // Inputs podem ser acessados de fora do componente como propriedades HTML
  // Outputs podem ser acessados de fora do componente como eventos no estilo onclick
  @Output() valueChanged = new EventEmitter<string>();

  @Input() breakLineOnInput = true;
  @Input() canSeePassword = true;

  @Input() seePassword = false
  @Output() seePasswordChanged = new EventEmitter<boolean>();

  // Membros protegidos não podem ser usados fora da classe, apenas no html
  protected inputType = "text";
  protected inputStyle = "color: black;"
  protected inputText = "";
  protected initialState = true;

  // Implmentamos OnInit para executar algum comportamento quando o componente inicializa
  ngOnInit(): void
  {
    // Atualizamos o inputType que aparece na tela
    this.updateInput()
  }

  /*  Aqui o evento foi pedido usando checkBoxToogle($event) no html e foi recuperado aqui.
   *  Isso não necessariamente precisa ser feito. Neste caso poderiamos usar apenas checkBoxToogle()
   *  e deixar essa função sem parâmetros já que não usamos o resultado do evento 'newValue'.
```

```
36      */
37    protected checkBoxToogle(newValue: any)
38    {
39      this.updateInput()
40      this.seePasswordChanged.emit(this.seePassword);
41    }
42
43    protected updateInput()
44    {
45      if (this.initialState)
46      {
47        this.inputText = "Escreva sua senha..."
48        this.inputType = "text"
49        this.inputStyle = "color: gray;"
50        return
51      }
52
53      this.inputStyle = "color: black;"
54      this.inputType = this.seePassword ? "text" :  "password";
55    }
56
57    protected passwordChanged()
58    {
59      this.updateInput()
60      this.valueChanged.emit(this.inputText)
61    }
62
63    protected passwordClick()
64    {
65      if (!this.initialState)
66        return
67
68      this.initialState = false;
69      this.inputText = "";
70      this.updateInput();
71    }
72
73    protected passwordFocusout()
74    {
```

```
    if (this.inputText !== "")
      return

      this.initialState = true
      this.updateInput()
  }
}
```

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'; // Adicionado para poder usar o ngModel

@NgModule({
  declarations: [
    AppComponent,
    NavComponent,
    LoginPageComponent,
    HomePageComponent,
    NotFoundPageComponent,
    FeedPageComponent,
    ComunityPageComponent,
    NewAccountPageComponent,
    RecoverPageComponent,
```

```
    UserPageComponent,
    PasswordComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule  // Adicionado para poder usar o ngModel
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Agora temos um interessante componente para colocar a senha com comportamento mais complexo. Note que nesta aula iremos apenas construir uma tela, não nos preocuparemos em adicionar reais funcionalidades como Login funcional, por exemplo. Assim nossa tela de Login é bem simples e com pouco comportamento:

login-page.component.html

```
<h1>
    Login
</h1>

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

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

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

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

<p>
```

```
    Esqueceu sua senha? <a href="/recover">Recupere agora!</a>
</p>
```

3 Reutilização de Componentes, Gerênciamento de dados e ngFor

Agora vamos fazer a tela de criação de conta para termos uma ideia do que mais poderíamos fazer:

create-password.component.html

```
<app-password (valueChanged)="passwordChanged($event)" />

<!-- Para cada elemento em passStrong coloca uma div a mais -->
<div class="strongBox">
    <div *ngFor="let x of passStrong">
        <div class="strongPass"></div>
    </div>
    {{passClassify}}
</div>

<p>
    <label>Repetir Senha</label>
    <br>
    <input type="password" [(ngModel)]="repeat" (change)="updateRepeatCondition()">
</p>

<p style="color: red;">
    {{ repeatEqualToPass ? "" : "As senhas devem ser iguais"}}
</p>
```

create-password.component.ts

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

@Component({
  selector: 'app-create-password',
  templateUrl: './create-password.component.html',
  styleUrls: ['./create-password.component.css']
})
export class CreatePasswordComponent {

```

```
10    protected passStrong = Array(1);
11    protected password = "";
12    protected repeat = "";
13    protected passClassify = "";
14    protected repeatEqualToPass = true;
15
16    protected updateStrongBar()
17    {
18      let finalStrong = 1;
19
20      if (this.password.length > 3)
21        finalStrong++;
22
23      if (this.password.length > 5)
24        finalStrong++;
25
26      if (this.password.length > 7)
27        finalStrong++;
28
29      if (this.password.length > 9)
30        finalStrong++;
31
32      if (this.password.match("[a-z]") != null)
33        finalStrong++;
34
35      if (this.password.match("[A-Z]") != null)
36        finalStrong++;
37
38      if (this.password.match("[0-9]") != null)
39        finalStrong++;
40
41      if (this.password.match("[\W]") != null)
42        finalStrong++;
43
44      this.passStrong = Array(finalStrong);
45
46      if (finalStrong < 3)
47      {
48        this.passClassify = "Senha muito fraca";
```

```
    }
    else if (finalStrong < 5)
    {
      this.passClassify = "Senha fraca"
    }
    else if (finalStrong < 7)
    {
      this.passClassify = "Senha mediana"
    }
    else if (finalStrong < 9)
    {
      this.passClassify = "Senha forte"
    }
    else
    {
      this.passClassify = "Senha muito forte"
    }
  }

  protected updateRepeatCondition()
  {
    this.repeatEqualToPass = this.password === this.repeat
  }

  protected passwordChanged(event: any)
  {
    this.password = event;
    this.updateStrongBar()
    this.updateRepeatCondition()
  }
}
```

create-password.component.css

```
.strongPass {
  width: 20px;
  height: 20px;
  margin: 2px;
```

```
  background: green;
}

.strongBox {
  display: flex;
  flex-direction: row;
}
```

new-account-page.component.html

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

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

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

<app-create-password/>

<p>
    <button>Criar Conta</button>
</p>
```

4 Exercícios

1. Faça um componente personalizado para armazenar e validar CPF.
2. Adicione-o a tela de criação de contas.
3. Faça um componente de card que possa armazenar uma imagem e um texto.
4. Tente criar vários posts na tela de feed com base nesses cads.