Aula 8

Descrição

Recapitular o que vimos, ver exercício angular

Integrando componentes

Descrição

Agora que temos componentes que precisam se conversar, vamos apreender como realizar essa iteração

Aplicativo do tempo

Temos dois componentes, um de busca, e outro que mostra a previsão do dia atual. Porém ainda eles não interagem entre si. Existem 4 técnicas principais para realizar iteração através de componentes:

- Eventos globais
- Componente pai escutando componente filho
- Componentes atuando como módulos por envio de dados
- Componente pai enviando informação para componente filho

Aplicativo do tempo

- Eventos globais

Técnica usada desde sempre, onde vimos em javascript onde tudo é tratado como função. No angular é considerado um anti padrão, acaba gerando código de difícil manutenibilidade e gerando um controle único da aplicação.

Aplicativo do tempo

- Componente pai escutando componente filho

O componente filho deve ficar completamente desacoplado do componente pai.
Vamos implementar este padrão para entendermos.

Primeiro, vamos adicionar um ‘EventEmitter’ no componente de busca.

Aplicativo do tempo

No ts do componente do tempo, criar uma propriedade
@Output() eventoDeBusca = new EventEmitter<string>()
e alterar o metodo “ngOnInit()”

```
ngOnInit(): void {

this.busca.valueChanges.pipe(debounceTime(1000)).subscr
ibe((valorDaBusca: string) => {
    if (!this.busca.invalid) {
      this.eventoDeBusca.emit(valorDaBusca)
    }   })  }
```

Aplicativo do tempo

- Componente pai escutando componente filho

Agora que criamos uma ação a ser enviada quando receber dados, vamos alterar o app.component.html para informar onde será recebido o evento e criar o consumo da api nessa parte.

Aplicativo do tempo

app.component.html :

```
<app-busca-cidade
(eventoDeBusca)="realizarBusca($event)"></app-busca-cidade>
```

app.component.ts:

```
 realizarBusca(valorDaBusca: string) {
    if (valorDaBusca) {
      const valorDoInput = valorDaBusca.split(',').map(letra =>
letra.trim())
      this.tempoService.getCurrentWeather(valorDoInput[0],
valorDoInput.length > 1 ? valorDoInput[1] :
undefined).pipe(debounceTime(1000))
        .subscribe(data => (console.log(data)))
    }
```

Aplicativo do tempo

Agora precisamos repassar o conteúdo para o componente tempo-atual. Vamos alterar a propriedade no tempo-atual para ter somente uma propriedade que será recebida :

```
export class TempoAtualComponent {
  @Input() tempo!: ITempoAtual
}
```

o @input indica que este componente precisa receber informações para quem usa ele.

Aplicativo do tempo

Bem, alteramos nosso componente, porém agora para que seja possível ver o tempo atual, precisamos alterar quem usa este componente, que é o componente pai, app.component:

```
<app-tempo-atual
[tempo]="tempoAtual"></app-tempo-atual>
```

e vamos alterar o app.component.ts aonde irá criar o nosso objeto do tempo

Aplicativo do tempo

```
export class AppComponent implements OnInit {
  tempoAtual!: ITempoAtual
  constructor(private tempoService: TempoService) { }
  ngOnInit(): void {
    this.tempoService.getCurrentWeather('Lages',
'Brasil').subscribe((data) => this.tempoAtual = data)
  }
...
```

Aplicativo do tempo

Vimos duas formas de interação entre componentes: pai recebendo informações do componente filho e o componente pai mandando informações para o componente filho. Isso acaba gerando um auto acoplamento entre estes componentes.

Aplicativo do tempo

- Componentes atuando como módulos por envio de dados, mais avançado.
  No angular podemos atuamos através de observadores, pelo rxjs, e através dele podemos ter componentes que serão observados e dinamicamente terão seus dados atualizados através de *‘subjects’*.
  Existem 3 comportamentos possíveis para os *‘subjects’* que são observados.

Aplicativo do tempo

3 tipos de ‘*subjects*’

- **ReplaySubject:** Irá lembrar todos os pontos de dados que ocorreram, assim podendo replicar todos os eventos que já ocorreram.
- **BehaviorSubject**: Irá lembrar somente do último estado, assim escutando somente novos dados.
- **AsyncSubject:** Este irá acontecer somente uma vez, não é experado que acontece novamente.

Aplicativo do tempo

Para o nosso caso do tempo, precisamos do '*BehaviorSubject*' vamos implementar ele. No serviço do tempo crie a propriedade e defina o valor padrão:

```
tempoAtual: BehaviorSubject<ITempoAtual> = new
BehaviorSubject<ITempoAtual>({
    cidade: '',
    pais: '',
    date: Date.now().toLocaleString(),
    descricao: '',
    temperatura: 0,
    image: ''
  })
```

Aplicativo do tempo

Agora vamos alterar o componente tempo-atual para assinar o valor recebido.

```
export class TempoAtualComponent implements OnInit {
  constructor(private tempoService: TempoService) { }
  ngOnInit(): void {
this.tempoService.tempoAtual.subscribe(data =>
(this.tempo = data)) }
  tempo!: ITempoAtual
```

Aplicativo do tempo

Agora vamos alterar o componente de busca para atualizar os dados:

```
export class BuscaCidadeComponent implements OnInit {
  busca = new FormControl('', [Validators.minLength(2)])
  eventoDeBusca = new EventEmitter<string>()
  constructor(private tempoService: TempoService) { }
  ngOnInit(): void {
this.busca.valueChanges.pipe(debounceTime(1000)).subscribe((valorDaBusca:
string) => {
      const valorDoInput = valorDaBusca.split(',').map(letra => letra.trim())
      this.tempoService.getCurrentWeather(valorDoInput[0], valorDoInput.length
> 1 ? valorDoInput[1] : undefined)
        .subscribe(data => this.tempoService.tempoAtual.next(data))
    })
  }
```

Aplicativo do tempo

Agora volte o app.component.ts:

```
export class AppComponent implements OnInit {
  tempoAtual!: ITempoAtual
  constructor(private tempoService: TempoService) { }
  ngOnInit(): void {
    this.tempoService.getCurrentWeather('Lages', 'Brasil').subscribe((data) =>
this.tempoService.tempoAtual.next(data))
  }
```

Aplicativo do tempo

Agora volte o app.component.ts:

```
export class AppComponent implements OnInit {
  tempoAtual!: ITempoAtual
  constructor(private tempoService: TempoService) { }
  ngOnInit(): void {
    this.tempoService.getCurrentWeather('Lages', 'Brasil').subscribe((data) =>
this.tempoService.tempoAtual.next(data))
  }
```

Agora dinamicamente a cada busca já assina os dados para os valores informados.

Angular - Exercício

1. Crie um componente que mostre a previsão dos dias da semana
2. Interaja com o novo componente pelo do método de iteração pai para filho e filho para pai “@INPUT e @OUTPUT e pelo componente de ´subjects´
3. Implemente iteração com o componente de poluição e busca.