Criando um novo componente Angular

UNISANTA

Índice

Criando um novo componente

Vamos agora criar um novo componente Angular para representar cada "Tarefa" apresentada. Este componente facilitará implementarmos os seguintes recursos:

- Marcar ou desmarcar uma tarefa com o status "Realizada"
- Alterar a descrição de uma tarefa existente
- Remover uma tarefa

Por questões didáticas, vamos batizar este novo componente com o nome "item". É importante lembrar que ItemComponent irá definir os elementos que serão apresentados para cada tarefa, bem como suas funcionalidades e aparência. Vamos utilizar o nome "item" para não causar confusão com outras declarações utilizadas para representar as Tarefas.

Para criar o novo componente, execute o seguinte comando dentro da pasta TODOapp:

```
ng generate component item
```

O comando acima irá criar um componente chamado de ItemComponent, e será considerado um componente filho de AppComponent. A execução do comando apresentará o seguinte resultado:

Referenciando ItemComponent

Agora vamos referenciar ItemComponent a partir da view de AppComponent. Vamos substituir a interpolação que apresentava a descrição da Tarefa, por uma referência a ItemComponent.

Para isso, basta substituir *{{tarefaDoLoop.descricao}}* por *<app-item></app-item>* dentro do arquivo **app.component.html**.

Após a alteração, o arquivo **app.component.html** ficará com o seguinte conteúdo:

```
<div class="main">
    <h1>Mozilla Dev - My To Do List</h1>
    <input class="lg-text-input" #campoNovoItem />
    <button class="btn-primary" (click)="CREATE_tarefa(campoNovoItem.value)">
        Adicionar Tarefa</button>
    <ul>
        <li *ngFor="let tarefaDoLoop of arrayDeTarefas">
            <app-item></app-item>
        </li>
    </ul>
</div>
```

A linha em verde acima, indica que um ItemComponent (chamado de app-item) deve ser apresentado nessa posição.

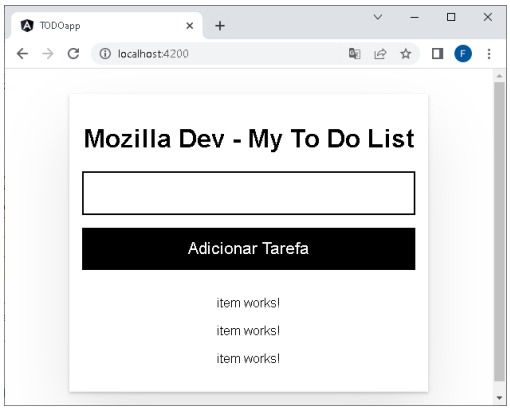
Consulte rapidamente o conteúdo do arquivo **item.component.html** e então visualize o resultado dessa modificação durante a execução do App.

Testando no Browser

Abra novamente o browser utilizando o mesmo endereço:

http://localhost:4200/

Resultado esperado:



Definindo as propriedades de ItemComponent

Substitua do código do arquivo **item.component.ts** por:

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Tarefa } from "../tarefa";

@Component({
  selector: 'app-item',
  templateUrl: './item.component.html',
  styleUrls: ['./item.component.css']
})
export class ItemComponent {
  emEdicao = false;
  @Input() tarefa: Tarefa = new Tarefa("", false);
}
```

O código acima implementa as seguintes funcionalidades:

- Adiciona a linha do "import" permitindo que o ItemComponent também utilize as definições realizadas na classe "Tarefa".
- Define uma propriedade chamada "emEdicao" que permitirá sinalizar se uma Tarefa está em edição ou não.
- Define uma propriedade chamada "tarefa" que permitirá armazenar a Tarefa representada por ItemComponent.

Definindo a view de ItemComponent

Substitua o código de **item.component.html** pelo seguinte conteúdo:

```
<div class="item">
    <input [id]="tarefa.descricao" type="checkbox" [checked]="tarefa.statusRealizada"
        (change)="tarefa.statusRealizada = !tarefa.statusRealizada;" />
    <label [for]="tarefa.descricao">{{tarefa.descricao}}</label>

    <div *ngIf="!emEdicao" class="btn-wrapper">
        <button class="btn" (click)="emEdicao = true ">Editar</button>
        <button class="btn btn-warn">Remover</button>
    </div>
    <div *ngIf="emEdicao">
        <input class="sm-text-input" #editedItem placeholder="escreva o novo nome da tarefa
aqui"
            [value]="tarefa.descricao">
        <div class="btn-wrapper">
            <button class="btn" (click)="emEdicao = false ">Cancelar</button>
            <button class="btn btn-save" (click)="tarefa.descricao=editedItem.value; emEdicao
= false;">Salvar</button>
        </div>
    </div>
</div>
```

O código acima, define os seguintes elementos para ItemComponent:

- Um **CHECKBOX** para informar se a tarefa foi feita ou não
- Um **LABEL** com a descrição da tarefa
- Quando a Tarefa não estiver sendo editada:
  - Um **BUTTON** para permitir editar a tarefa
  - Um **BUTTON** para permitir remover a tarefa
- Quando a Tarefa estiver sendo editada:
  - Um **INPUT** para definir a nova descrição da tarefa
  - Um **BUTTON** para permitir cancelar a edição
  - Um **BUTTON** para permitir salvar a alteração

No exemplo acima, a diretiva ngIf do Angular apresenta a correspondente, dependendo do valor da variável "emEdicao"

Transferindo os dados da Tarefa para ItemComponent

Considerando que agora cada Tarefa será representada por um ItemComponent, temos que transferir os dados da Tarefa armazenada em AppComponent (neste caso considerado como sendo o componente Pai) para o ItemComponent (neste caso considerado como sendo o componente Filho). Isso pode ser facilmente realizado através de um mecanismo chamado de "Property Binding". A transferência é feita durante a indicação do componente Filho dentro da view do componente Pai.

Em nosso caso, basta alterar o código de **app.component.html** acrescentando o trecho destacado abaixo na cor verde:

```
<div class="main">
    <h1>Mozilla Dev - My To Do List</h1>
    <input class="lg-text-input" #campoNovoItem />
    <button class="btn-primary" (click)="CREATE_tarefa(campoNovoItem.value)">
        Adicionar Tarefa</button>
    <ul>
        <li *ngFor="let tarefaDoLoop of arrayDeTarefas">
            <app-item [tarefa]="tarefaDoLoop"></app-item>
        </li>
    </ul>
</div>
```

O código acrescentado acima, passa a Tarefa referenciada pela variável "tarefaDoLoop" para o atributo "tarefa" de ItemComponent.

Definindo o estilo de ItemComponent

Altere o conteúdo do arquivo **item.component.css** com as seguintes definições:

```
.item {
    padding: .5rem 0 .75rem 0;
    text-align: left;
    font-size: 1.2rem;
}

.btn-wrapper {
    margin-top: 1rem;
    margin-bottom: .5rem;
}

.btn {
    /* menu buttons flexbox styles */
    flex-basis: 49%;
}

.btn-save {
    background-color: #000;
    color: #fff;
    border-color: #000;
}
```

```
.btn-save:hover {
    background-color: #444242;
}

.btn-save:focus {
    background-color: #fff;
    color: #000;
}

.checkbox-wrapper {
    margin: .5rem 0;
}

.btn-warn {
    background-color: #b90000;
    color: #fff;
    border-color: #9a0000;
}

.btn-warn:hover {
    background-color: #9a0000;
}

.btn-warn:active {
    background-color: #e30000;
    border-color: #000;
}

.sm-text-input {
    width: 100%;
    padding: .5rem;
    border: 2px solid #555;
    display: block;
    box-sizing: border-box;
    font-size: 1rem;
    margin: 1rem 0;
}

[type="checkbox"]:not(:checked),
[type="checkbox"]:checked {
    position: absolute;
    left: -9999px;
}

[type="checkbox"]:not(:checked)+label,
[type="checkbox"]:checked+label {
    position: relative;
    padding-left: 1.95em;
    cursor: pointer;
}

[type="checkbox"]:not(:checked)+label:before,
[type="checkbox"]:checked+label:before {
    content: '';
    position: absolute;
    left: 0;
    top: 0;
    width: 1.25em;
    height: 1.25em;
    border: 2px solid #ccc;
    background: #fff;
}

[type="checkbox"]:not(:checked)+label:after,
[type="checkbox"]:checked+label:after {
    content: '\2713\0020';
```

```
    position: absolute;
    top: .15em;
    left: .22em;
    font-size: 1.3em;
    line-height: 0.8;
    color: #0d8dee;
    transition: all .2s;
    font-family: 'Lucida Sans Unicode', 'Arial Unicode MS', Arial;
}

[type="checkbox"]:not(:checked)+label:after {
    opacity: 0;
    transform: scale(0);
}

[type="checkbox"]:checked+label:after {
    opacity: 1;
    transform: scale(1);
}

[type="checkbox"]:checked:focus+label:before,
[type="checkbox"]:not(:checked):focus+label:before {
    border: 2px dotted blue;
}
```

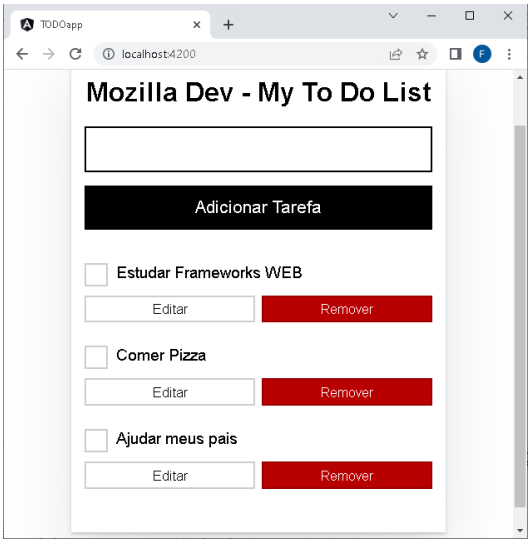
Pelo mesmo motivo explicado na definição do estilo de AppComponent, é possível que nem todas as definições de estilo acima, estejam efetivamente sendo aproveitadas pelos elementos HTML apresentados pelo componente correspondente.

Testando no Browser

Abra novamente o browser utilizando o mesmo endereço:

http://localhost:4200/

Resultado esperado:



Permitindo a remoção de Tarefas

Neste momento o único recurso que falta ser concluído em nosso App é permitir a "Remoção de Tarefas" da lista. É importante saber que, por mais que tenhamos incluído um botão para esta finalidade, o mesmo ainda não é capaz de realizar qualquer ação.

Para implementar essa solução, temos que notar que a lista de Tarefas está armazenada em um array existente dentro de AppComponent, e que o botão "Remover" está definido dentro de ItemComponent. Isso significa que o componente Filho (ItemComponent) deverá avisar o componente Pai (AppComponent) que o botão "Remover" foi pressionado.

Para resolver essa situação podemos fazer uso de um mecanismo chamado de "Event Binding", que permitirá que ItemComponent emita um evento para ser capturado por AppComponent.

Declarando um emissor de evento

O primeiro passo necessário para implementar o Event Binding, é declarar uma propriedade no componente Filho utilizando a diretiva @Output e armazenando um objeto do tipo EventEmitter.

Adicione a seguinte linha destacada em verde ao código de **item.component.ts**:

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Tarefa } from "../tarefa";

@Component({
  selector: 'app-item',
  templateUrl: './item.component.html',
  styleUrls: ['./item.component.css']
})
export class ItemComponent {
  emEdicao = false;
  @Input() tarefa: Tarefa = new Tarefa("", false);
  @Output() removeTarefa = new EventEmitter();
}
```

Emitindo o evento

O próximo passo é avisar o componente Pai (AppComponent) sempre que o botão "Remover" for pressionado. Para isso vamos fazer o componente Filho emitir um evento no momento em que for detectado o "click" do botão "Remover". Segue destacado em verde, o código que deve ser adicionado em **item.component.html** com esse propósito:

```
<div class="item">
    <input [id]="tarefa.descricao" type="checkbox" [checked]="tarefa.statusRealizada"
        (change)="tarefa.statusRealizada = !tarefa.statusRealizada;" />
    <label [for]="tarefa.descricao">{{tarefa.descricao}}</label>

    <div *ngIf="!emEdicao" class="btn-wrapper">
        <button class="btn" (click)="emEdicao = true ">Editar</button>
        <button class="btn btn-warn" (click)="removeTarefa.emit()">Remover</button>
    </div>
    <div *ngIf="emEdicao">
        <input class="sm-text-input" #editedItem placeholder="escreva o novo nome da tarefa
aqui"
            [value]="tarefa.descricao">
        <div class="btn-wrapper">
            <button class="btn" (click)="emEdicao = false ">Cancelar</button>
            <button class="btn btn-save" (click)="tarefa.descricao=editedItem.value; emEdicao
= false;">Salvar</button>
        </div>
    </div>
</div>
```

Preparando AppComponent para eliminar a Tarefa

Considerando que a lista de Tarefas está armazenada no componente Pai (AppComponent), devemos implementar um método que seja capaz de eliminar uma Tarefa do arrayDeTarefas. Segue o código do método que deve ser adicionado em **app.component.ts** com esse propósito:

```
DELETE_tarefa(tarefaAserRemovida : Tarefa) {
    this.arrayDeTarefas.splice(this.arrayDeTarefas.indexOf(tarefaAserRemovida), 1);
}
```

Detectando o evento em AppComponent

Insira o código destacado em verde em **app.component.html.** Esta construção executará o método DELETE_tarefa no momento em que for detectado o evento "removeTarefa" emitido pelo componente Filho.

```
<div class="main">
    <h1>Mozilla Dev - My To Do List</h1>
    <input class="lg-text-input" #campoNovoItem />
    <button class="btn-primary" (click)="CREATE_tarefa(campoNovoItem.value)">
        Adicionar Tarefa</button>
    <ul>
        <li *ngFor="let tarefaDoLoop of arrayDeTarefas">
            <app-item [tarefa]="tarefaDoLoop" (removeTarefa)="DELETE_tarefa(tarefaDoLoop)">
            </app-item>
        </li>
    </ul>
</div>
```

Testando no Browser

Abra novamente o browser utilizando o mesmo endereço:

http://localhost:4200/

Experimente testar todos os recursos do programa como Adicionar, Editar e Remover tarefas.

Considerações finais

Ao chegarmos aqui finalizamos a primeira versão de nosso App Angular.

É importante notar que neste momento, nosso App não é capaz de persistir as informações das Tarefas cadastradas. Isso acontece porque todo o armazenamento dos dados está sendo realizado em memória.

Ao longo das aulas vamos aprender como armazenar as informações em um Banco de Dados, fazendo uso de uma arquitetura simplificada de micro-serviços disponibilizados a partir de uma REST API.

Histórico de revisões

Revisão: 00
Data: 13/07/2022
Descrição das alterações:
Documento original