Nota de Revisão: Curso Angular #13: Reusando Componentes com Input properties

Introdução

Neste vídeo da Loiane Groner, aprendemos a reutilizar componentes em Angular usando Input properties. A autora demonstra como criar um novo componente, declarar um Input property e utilizá-lo em outro componente. Ela também discute as melhores práticas para usar Input properties em projetos reais.

Tópicos principais

- **Criando um novo componente:** A autora utiliza o comando `ng generate component` para criar um novo componente chamado `embedded-título`.
- **Declarando um Input property:** Ela adiciona uma propriedade chamada `nome` ao componente `embedded-título`. Essa propriedade recebe um valor do tipo string.
- **Utilizando um Input property:** A autora utiliza o componente `embedded-título` em outro componente chamado `data-base-componente`. Ela passa o valor "Curso de Angular" para a propriedade `nome`.
- **Melhorando a legibilidade do código:** A autora sugere usar o decorador `@Input()` para declarar Input properties. Isso torna o código mais legível e facilita a compreensão do que cada propriedade faz.
- **Usando Input properties em projetos reais:** A autora fornece um exemplo de como usar Input properties em um projeto real. Ela cria um componente para exibir os detalhes de um Pokémon. O componente recebe um objeto Pokémon como entrada e exibe as informações do Pokémon na tela.

Conclusão

Este vídeo é um recurso valioso para aprender a reutilizar componentes em Angular usando Input properties. A autora fornece uma explicação clara e concisa do tópico, e ela também compartilha dicas úteis para usar Input properties em projetos reais.

Recursos adicionais

- Código-fonte: https://github.com/loianegruner/curso-angular/tree/master/13-reutilizando-componentes-com-input-properties
- Material de apoio: https://www.loiane.com/curso-angular/modulo-3/aula-13-reutilizando-componentes-com-input-properties/

Exemplos de Reutilização de Componentes em Angular com Input Properties

1. Botão de Curtir:

Imagine um botão de curtir reutilizável que pode ser usado em diferentes partes do seu aplicativo, como em posts de blog, comentários ou produtos. Este botão pode ser criado como um componente Angular com uma propriedade de entrada `likes` que recebe o número de curtidas. O componente pode então exibir o número de curtidas e um ícone de curtir.

HTML do Componente:

HTML

```
<button class="like-button">
  <span class="likes-count">{{ likes }}</span>
  <i class="fas fa-heart"></i>
</button>
```

TypeScript do Componente:

TypeScript

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

```typescript
@Component({
  selector: 'app-like-button',
  templateUrl: './like-button.component.html',
  styleUrls: ['./like-button.component.css']
})
export class LikeButtonComponent {
  @Input() likes: number = 0;
}
```

Utilizando o Componente:

```HTML
<app-like-button [likes]="post.likes"></app-like-button>
```

2. Componente de Produto:

Crie um componente de produto reutilizável para exibir informações sobre um produto, como nome, imagem e preço. Este componente pode ter propriedades de entrada para cada um desses campos.

HTML do Componente:

```HTML
<div class="product-card">
  <img src="{{ product.image }}" alt="{{ product.name }}">
  <h3>{{ product.name }}</h3>
  <p class="price">{{ product.price }}</p>
</div>
```

TypeScript do Componente:

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

@Component({
  selector: 'app-product-card',
  templateUrl: './product-card.component.html',
  styleUrls: ['./product-card.component.css']
})
export class ProductCardComponent {
```

```
  @Input() product: any;
}
```

Utilizando o Componente:

```HTML
<app-product-card [product]="produto"></app-product-card>
```

3. Formulário de Cadastro:

Um formulário de cadastro reutilizável pode ser criado como um componente com propriedades de entrada para cada campo do formulário, como nome, email e senha. O componente pode então exibir os campos de entrada e validar os dados inseridos pelo usuário.

HTML do Componente:

```HTML
<form [formGroup]="cadastroForm">
  <input type="text" formControlName="nome" placeholder="Nome completo">
  <input type="email" formControlName="email" placeholder="Email">
  <input type="password" formControlName="senha" placeholder="Senha">

  <button type="submit">Cadastrar</button>
</form>
```

TypeScript do Componente:

```TypeScript
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';

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

```
  cadastroForm: FormGroup;

  constructor() { }

  ngOnInit(): void {
    this.cadastroForm = new FormGroup({
      nome: new FormControl('', [Validators.required]),
      email: new FormControl('', [Validators.required, Validators.email]),
      senha: new FormControl('', [Validators.required,
Validators.minLength(6)])
    });
  }

  onSubmit() {
    // Processar o formulário
  }
}
```

Utilizando o Componente:

```
HTML
<app-cadastro-form></app-cadastro-form>
```

Observações:

- Estes são apenas alguns exemplos simples de como reutilizar componentes em Angular com Input properties.
- Você pode criar componentes muito mais complexos com várias propriedades de entrada, saídas e métodos de ciclo de vida.
- A reutilização de componentes é uma ótima maneira de modularizar seu código, melhorar a legibilidade e facilitar a manutenção do seu aplicativo Angular.

Recursos Adicionais:

- Documentação Angular sobre Componentes [URL inválido removido]
- Curso Angular #13: Reusando Componentes com Input properties