Angular Components: Conhecendo e configurando no seu projeto

Neste artigo entenderemos como a parte lógica de um projeto Angular pode ser criada utilizando componentes.

Por que eu devo ler este artigo: Neste artigo entenderemos como Components funcionam no Angular. Partindo de um novo projeto, veremos como eles são criados, a sua estrutura e como implementar

Atualmente, o Angular é um dos mais conhecidos frameworks para desenvolvimento web. Criado e mantido pela Google, ele já tem uma longa estrada no mercado. Uma das partes mais importantes de um projeto Angular é o que chamamos de componentes, pois eles são a parte lógica da aplicação.

Estrutura de um Component

Os componentes são classes escritas em TypeScript que recebem o decorator `@Component` , assim como mostra o **Código 1**. Vamos analisar esse código juntos ao

longo dessa seção, portanto não se preocupe em entendê-lo completamente agora. Atente-se apenas para o fato dele ser uma classe com um decorator, por enquanto.

```
@Component({  selector: 'cadastro',
   templateUrl: 'cadastro.component.html',
    styleUrls: ['cadastro.component.css']
})
export class CadastroComponent {

}
```

Código 1. Exemplo de um Component no Angular

Quando analisamos a estrutura de um componente, podemos dividi-lo nas seguintes partes:

- Metadata: são definições que iremos informar ao Angular que a nossa classe é um Component, por meio de decorators, nesse caso `@Component`
- Classe: assim como em qualquer linguagem de programação que utiliza o paradigma da orientação a objeto, possui suas propriedades e métodos

Um Component é um tipo de classe existente em uma aplicação Angular, eles são identificados com o decorator `@Component`. Neste decorator existem algumas propriedades mais utilizadas como: `selector`, `templateUrl` e `style`. Vamos especificá-las:

- **selector** é como identificamos o nosso componente. Para todo o componente existe um elemento único associado que permite que ele seja adicionado em um documento HTML. Nesse caso, o nome do elemento desse componente é `cadastro` e deve ser escrito como `<cadastro></cadastro>`

- **TemplateUrl** é o nome do documento HTML que será a parte visual do componente. Nele podemos ter código em HTML juntamente com todos os bindings e diretivas necessários para a exibição do componente no navegador
- **Template** também usado para descrever a parte visual do componente, porém nesse caso podemos fornecer código HTML "hard coded", como texto
- **styleUrl** é onde informamos quais folhas de estilo contêm o código CSS que será aplicado ao template do componente.

Com isso podemos concluir que a estrutura de um componente Angular deve ser formada por esses três elementos, template (HTML), estilo (CSS) e classe (TypeScript). Essas partes são organizadas em arquivos separados e unidas nos metadados do componente através do decorator `@Component`

No **Código 2** vemos uma outra forma de descrever o template do componente usando a propriedade `template`, sem a necessidade de criar um arquivo HTML no projeto.

```
1 @Component({  selector: 'cadastro',
2     template:'<input type="text"[(ngModel)]="nome" name="nome" id="nome
3     styleUrls: ['cadastro.component.css']
4 })
5 export class CadastroComponent {
6
7 }
```

Código 2. Outro exemplo de Component usando a propriedade **template** ao invés de **templateUrl**

Como boa prática, o código HTML implementado na propriedade `template` deve conter poucas linhas. Caso este seja um código mais elaborado é recomendável colocá-lo em um documento HTML, utilizando a propriedade `templateUrl` para indicar a localização deste arquivo.

Podemos fazer referência ao Component `CadastroComponent` em outros componentes do projeto usando o seletor `cadastro`. Nesse caso basta colocarmos a tag `<cadastro></cadastro>` dentro do HTML do template.

Criando componente com o Angular CLI

Em um projeto Angular usaremos uma ferramenta chamada Angular CLI para criar componentes. Com ela nos preocupamos menos com a estrutura do componente, pois ela adiciona o mesmo ao projeto já com um código inicial automaticamente. Vamos ver como usar essa ferramenta na prática nesta seção.

Considerando que o Angular já se encontra instalado no seu sistema operacional, iremos em um terminal, no caso do Windows o **cmd** e no caso de Linux e MacOS o **terminal**, para criar uma pasta para um projeto de exemplo com o nome "artigo". Para isso usamos o comando no **Código 3** para criação do projeto.

```
1 | ng new artigo
```

Código 3. Comando para criação do projeto Angular

Durante a execução do comando, dependendo da versão do Angular CLI instalada em seu computador, podem aparecer algumas perguntas relacionadas a criação do

projeto, conforme a **Figura 1.**

Figura 1. Perguntas que ocorrem durante a criação do projeto

Para este projeto, não precisaremos utilizar rotas, que tem a ver com a navegação entre as páginas do projeto. Portanto, na primeira pergunta informe **N** para não gerar um arquivo de rota. Como linguagem de estilo usaremos o CSS, então pressione a tecla **ENTER** sobre essa opção. Durante o processo de criação do projeto será criada uma pasta chamada "artigo" no diretório atual com toda estrutura do projeto.

No final da instalação pode aparecer uma pergunta se se deseja enviar informações para o time do Angular, como visto na **Figura 2.**, mas isso é opcional. Para saber quais informações serão coletadas consulte a documentação Usage Metrics Gathering .

Figura 2. Se deseja enviar informações para o time do Angular

Quando terminar, vamos entrar na pasta criada para executarmos o comando no **Código 4**. Esse comando vai iniciar a aplicação na porta padrão, 4200.

```
1 | ng serve
```

Código 4. Comando usado para iniciar a aplicação

Se ao digitar `ng serve` não ocorrer nenhum erro, conseguiremos executar a aplicação. Digite "http://localhost:4200" em seu navegador para visualizarmos a tela com as informações sobre a aplicação, como mostra a **Figura 3**.

Figura 3. Aplicação aberta no browser

Localizando os componentes do projeto

No projeto criado já existe um componente criado por padrão, como podemos observar na **Figura 4**.

Figura 4. Estrutura do projeto criado

Analisando a **Figura 4** podemos identificar dentro da pasta chamada "app" o arquivo "app.component.ts", criado automaticamente. Se o abrirmos veremos que a sua estrutura é igual a do **Código 1**, apenas um classe decorada.

Criação de um componente

Agora que temos o projeto criado vamos adicionar nele um novo componente. A maioria das aplicações Angular será composta de muitos componentes. Para criar um novo componente deveremos executar no terminal o comando visto no **Código 5** dentro da pasta do projeto.

```
ng g component pagina
```

Código 5. Código para a criação do Component

O comando acima gerará um novo componente dentro da pasta "app" chamado pagina, dentro de uma pasta chamada "pagina". Esse componente será composto pelos arquivos "pagina.component.html", "pagina.component.css" e "pagina.component.ts", que contém a estrutura do Component presente no **Código 6**.

```
import { Component, OnInit } from '@angular/core';
  @Component({
    selector: 'app-pagina',
    templateUrl: './pagina.component.html',
    styleUrls: ['./pagina.component.css']
})
export class PaginaComponent implements OnInit {

```

```
    constructor() { }

    ngOnInit() { }

}
```

Código 6. Conteúdo do arquivo **pagina.component.ts**

O Angular CLI facilita muito a criação de um novo componente, pois além de criar os arquivos que o compõem, também registra o componente a nível de projeto. No Angular, todo componente precisa ser registrado em um módulo antes de ser utilizado. O Angular CLI, ao criar o componente, também configura a referência para ele no arquivo **app.modules.ts**, conforme **Código 7**, de modo que não precisamos nos preocupar com esse passo.

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

import { AppComponent } from './app.component';
import { PaginaComponent } from './pagina/pagina.component';

@NgModule({
    declarations: [
      AppComponent,
      PaginaComponent
    ],
    imports: [
      BrowserModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Código 7. Vínculo do **PaginaComponent** no arquivo **app.modules.ts**

Como vimos, as referências aos componentes ficam dentro do decorator `@NgModule` na propriedade `declarations`. É nela que, a princípio, ficarão todos os componentes da nossa aplicação.

Utilizando um componente

Agora, utilizaremos o componente "pagina" para entender como ele pode funcionar em uma aplicação. Serão exibidos alguns textos no browser, porém o conteúdo do texto estará no arquivo **pagina.component.ts** e usaremos o recurso do binding do Angular para exibir na view.

No arquivo **pagina.component.ts** colocaremos as variáveis que receberão o título e o conteúdo do texto.

Conforme **Código 8**, criaremos duas variáveis do tipo string: uma chamada `titulo` e a outra chamada `texto`. Ambas terão o modificador de acesso `public`, já que usaremos em uma view (todo ou parte de uma página web) da aplicação.

```
1  import { Component, OnInit } from '@angular/core';
2  @Component({
3      selector: 'app-pagina',
4      templateUrl: './pagina.component.html',
5      styleUrls: ['./pagina.component.css']
6  })
7  export class PaginaComponent implements OnInit {
8
9      public titulo: string;
10     public texto: string;
11
```

```
11
12      constructor() { }
13
14      ngOnInit() { }
15
16  }
```

Código 8. Implementação das variáveis no arquivo **pagina.component.ts**

Dentro do decorator `@Component` temos a propriedade `templateUrl`, onde nesse caso colocaremos o código HTML no arquivo "pagina.component.html".

No arquivo "pagina.component.html", implementamos um código que exibirá na tela o título e o texto, como veremos no **Código 9**.

```
1  <div id="header">
2      <h1><p class="textoTitulo">Artigo</p></h1>
3  </div>
4  <div id="titulo">
5        <h2>{{titulo}}</h2>
6  </div>
7  <div id="texto">
8        {{texto}}
9  </div>
```

Código 9. Implementação na view pagina.component.html

Para que os valores das variáveis possam ser exibidos na view podemos utilizar a interpolation (interpolação), que consiste em adicionar o nome da variável entre chaves duplas `{{ }}` dentro do elemento em que o valor da variável deve aparecer.

Colocaremos no arquivo "pagina.component.ts" o conteúdo do **Código 10** com os valores das variáveis criadas para exibição na view.

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

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

    public titulo: string;
    public texto: string;

    constructor() {}

    ngOnInit() {

      this.titulo = 'O que são Components no Angular?';
      this.texto = `Lorem ipsum dolor sit amet, consectetur adipiscing
                    sed do eiusmod tempor incididunt ut labore et dolo
                    aliqua. Nibh mauris cursus mattis molestie a iacul
                    Ipsum dolor sit amet consectetur adipiscing elit u
                    purus. Egestas tellus rutrum tellus pellentesque e
                    tortor aliquam nulla. In dictum non consectetur a
                    Enim ut sem viverra aliquet eget sit amet tellus c
                    Fames ac turpis egestas integer. Odio pellentesque
                    commodo sed egestas. Augue lacus viverra vitae con
                    ac. Et malesuada fames ac turpis egestas maecenas
                    Sed lectus vestibulum mattis ullamcorper velit sed
                    Sollicitudin tempor id eu nisl nunc mi ipsum fauci

    }

}
```

Código 10. Colocando os valores das variáveis no arquivo "pagina.component.ts"

Para conseguirmos visualizar o conteúdo da nossa view, visto que não configuramos uma rota no nosso projeto, teremos que fazer uma alteração no arquivo "app.component.html" na raiz da nossa pasta "app". Excluiremos o conteúdo já existente e colocaremos o seletor do nosso componente **app-pagina** para quando carregarmos a aplicação a nossa view possa aparecer, como mostra a **Figura 5**.

Figura 5. Colocando o seletor no arquivo app.component.html

Caso a aplicação esteja em execução, logo a nova página será exibida, do contrário, deverá executar o comando `ng serve` em um terminal dentro da pasta do nosso projeto.

Poderemos dar uma melhorada no visual da aplicação editando o arquivo **pagina.compoment.css**, onde colocaremos o código CSS do **Código 11**.

```
#header{   position: relative;
      width: 100%;
      height: 50px;
      background-color: darkgrey;
}

.textoTitulo{

```

```
9         position: absolute;
10        left: 12px;
11        top: -25px;
12    }
13
14    #titulo{
15        text-align: center;
16    }
17
18    #texto{
19        text-align: justify;
    }
```

Código 11. Inclusão da folha de estilos no **pagina.compoment.css**

Quando a página for executada novamente veremos a aplicação com um visual bem melhor.

Conclusão

Os componentes são uma parte importante de toda aplicação Angular, pois é neles que criamos a lógica por trás do funcionamento das views. Neste artigo vimos como eles são estruturados e como implementá-los em um projeto já criado. Também vimos como podemos configurar as variáveis criadas dentro de um componente e fazer com que os seus valores sejam visualizados em um template.

Tecnologias:

Angular TypeScript

Confira outros conteúdos:

Angular Filter e pipes

Angular: Interação entre componentes

Por **Devmedia**
Em 2019

<Formação completa Programador FullStack/>

- Conteúdo Front-end, Back-end e Mobile
- Plano de estudo linear
- +10 mil exercícios gamificados
- +50 projetos reais
- Comunidade com + 200 mil alunos
- Suporte 365 dias do ano
- 12 meses de acesso

Comece agora

Perguntas Frequentes

Quem somos?

Por que a programação se tornou a profissão mais promissora da atualidade?

Como faço para começar a estudar?

Em quanto tempo de estudo vou me tornar um programador?

Sim, você pode se tornar um programador e não precisa ter diploma de curso superior!

O que eu irei aprender estudando pela DevMedia?

Principais diferenciais da DevMedia

Qual o investimento financeiro que preciso fazer para me tornar um programador?

Como funciona a forma de pagamento da DevMedia?

Nossos casos de sucesso

Leonardo Carlos

Eu sabia pouquíssimas coisas de programação antes de começar a estudar com vocês, fui me especializando em várias áreas e ferramentas que tinham na plataforma, e com essa bagagem **consegui um estágio logo no início do meu primeiro período na faculdade.**

Lucas Rodrigues

Estudo aqui na Dev desde o meio do ano passado! Nesse período a Dev me ajudou a crescer muito aqui no trampo.
Fui o primeiro desenvolvedor contratado pela minha empresa. Hoje eu lidero um time de desenvolvimento!
Minha meta é continuar estudando e praticando para ser um Full-Stack Dev!

Heráclito Júnior

Economizei 3 meses para assinar a plataforma e sendo sincero valeu muito a pena, pois a **plataforma é bem intuitiva e muuuuito didática a metodologia de ensino.** Sinto que estou EVOLUINDO a cada dia. Muito obrigado!

Julio Cablen

Nossa! Plataforma maravilhosa. To amando o curso de desenvolvimento front-end, tinha coisas que eu ainda não tinha visto. **A didática é do jeito que qualquer pessoa consegue aprender.** Sério, to apaixonado, adorando demais.

Joelberth Sena

Adquiri o curso de vocês e logo percebi que são os melhores do Brasil. É um passo a passo incrível. **Só não aprende quem não quer. Foi o melhor investimento da minha vida!**

Felipe Nunes

Foi um dos melhores investimentos que já fiz na vida e tenho aprendido bastante com a plataforma. Vocês estão fazendo parte da minha jornada nesse mundo da programação, **irei assinar meu contrato como programador graças a plataforma.**

Wanderson Oliveira

Comprei a assinatura tem uma semana, aprendi mais do que 4 meses estudando outros cursos. Exercícios práticos que não tem como não aprender, estão de parabéns!

José Lucas

Obrigado DevMedia, nunca presenciei **uma plataforma de ensino tão presente na vida acadêmica de seus alunos**, parabéns!

Eduardo Dorneles

Aprendi React na plataforma da DevMedia há cerca de 1 ano e meio... **Hoje estou há 1 ano empregado** trabalhando 100% com React!

Adauto Junior

Já fiz alguns cursos na área e **nenhum é tão bom quanto o de vocês.** Estou aprendendo muito, muito obrigado por existirem. Estão de parabéns... Espero um dia conseguir um emprego na área.

Ver todos os casos de sucesso

Termos de uso

Política de estorno

DevMedia: 08.401.613/0001-42

Rua Victor Civita, 66 - Salas 306, 307 e 308 -

Jacarepaguá

Rio de Janeiro - RJ, 22775-044

Tecnologia:

HTML CSS Algoritmo Javascript React React Native Node.js SQL MySQL UML Scrum Levantamento de Requisitos Padrão de Projeto Teste de Software C# Delphi Dart Java Kotlin PHP Python TypeScript Angular Vue.js Django Laravel Spring .NET Flutter Modelagem de Dados Oracle REST PostgreSQL SQL Server MVC Orientação a Objeto Docker Git Scrum

Cursos:

HTML e CSS Javascript Programação para Iniciantes Angular React Vue.js Node.js Spring .NET Core Mobile React Native Android Flutter Algoritmo Automação Delphi Java PHP Python SQL e Banco de Dados Engenharia de Software Canal Mais Gratuitos

Artigos:

Front-End Javascript Iniciantes Angular Dart Engenharia Mobile Node.js Python React Native Vue.js Android Banco de Dados Delphi Flutter Java Kotlin .Net PHP React Spring Gratuitos

DevCast:

HTML e CSS Javascript Angular Engenharia Mobile Node.js Python React Native Android Banco de Dados Delphi Flutter Java Automação .Net PHP React Spring Gratuitos Canal Mais

Guia:

Fundamentos .NET PHP Python Java Delphi HTML e CSS JavaScript Node React Native Flutter Banco de Dados Mobile Spring Arquitetura Automação Engenharia + Assuntos