Aula 31 - Angular Básico

Docupedia Export

Author:Silio Leonardo (CtP/ETS)
Date:05-Jun-2023 13:59

Table of Contents

- Iniciando no Angular
- O básico sobre Componentes
- O básico sobre Rotas
- O básico sobre Binding e Templates
- Próximos passos
- Exercícios

1 Iniciando no Angular

Para começar vamos a instalação do Angular. Você só precisa do **npm** instalado no seu computador para que possa executar o seguinte comando:

```
npm install -g @angular/cli
```

Em alguns computadores é necessário permitir a execução de scripts PowerShell não assinados, você pode fazer isso executando:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

E então, dentro de uma pasta qualquer inicializar o seu projeto da seguinte forma:

```
ng new example-app
```

Existe uma grande possibilidade do comando ng não ser encontrado. Ele é instalado junto ao primeiro comando na pasta %AppData%, mas pode não ser reconhecível pelo PowerShell em um primeiro momento. Você pode executá-lo através do **npm** para que ele seja reconhecível da seguinte forma:

```
npm run ng new example-app
```

Note que é comum, no Angular, usar um convensão para nomes exatamente iguail ao padrão de nome de repositórios no GitHub.
Nosso último comando irá criar um projeto Angular vazio, muito embora o projeto base seja extremamente carregado de arquivos. A estrutura incial do projeto é semelhante a abaixo:

example-app
- .angular/
- .vscode/
- node_modules/
- **src**
  - **app**
    - app-routing.module.ts
    - app.component.css
    - app.component.html
    - app.component.spec.ts
    - app.component.ts
    - app.module.ts
  - assets/
  - index.html
  - main.ts
  - style.css
- .editorconfig

.gitignore
angular.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json

Enquanto alguns arquivos são mais utilizados para configuração e geração de código, como index.html e o styles.css que muitas vezes não são alterados por serem apenas templates para o projeto, a pasta app é onde o trabalho em geral acontece. Para iniciar a execução do projeto você pode utilizar um comando que fica definido na package.json:

```
npm start
```

Ele irá compilar todo código e criar um servidor web que responderá em um link que será mostrado no terminal no seguinte modelo: http://localhost:port/. Comumente a porta será 4200. Ao acessar, o servidor nos dará uma página 100% Html, Css e JavaScript algumas vezes irreconhecível ao código original.
Agora vamos compreender como um projeto angular está estrututrado.

2 O básico sobre Componentes

O Angular é estruturado em componentes, isto é, uma estrutura que pode ser replicada como se fosse uma tag própria. Ela tem seu prórpio html, cdd e comportamento, que por sua vez não é definido com JavaScript (.js) no Angular, mas sim TypeScript (.ts), uma tecnologia que se converte em JavaScript ao ser executado. Assim, todo componente terá 4 arquivos associados a ele:

- x.component.css: O css que é aplicado apenas a este componente.
- x.component.html: O html a ser reproduzido ao utilizarmos este componente.
- x.cmponent.spec.ts: Especificação de testes do componente.
- x.component.ts: Classe TypeScirpt que define comportamento do componente.

Como você pode perceber, você inicia sua aplicação com um componente chamado app. Note que existem ainda dois arquivos não mensionados acima, o 'app-routing.module.ts' e o 'app.module.ts', mas esses não fazem parte do componente app e sim arquivos importantes de configuração do projeto em si.
Toda vez que você inicia uma aplicação o componente app será renderizado para você. Note que existe um grane comentário neste componente apontando que todo conteúdo nele é apenas de exemplo. Por isso, no component.html iremos simplificá-lo deixando-o apenas assim:

app.component.html

```
1 <router-outlet></router-outlet>
```

A tag 'router-outlet' será subsittuido por um código presente em alguma rota específica (depende do URL acessado). Mais tarde veremos como isso funciona, mas antes, vamos a uma definição básica para compreendermos como tudo funciona. Vamos criar nosso próprio componente. Você poderia fazer isso apenas criando arquvios e digitando código, mas também pode usar o seguinte código em um terminal PowerShell qualquer:

```
npm run ng generate component Main
```

Uma pasta dentro de app será criada com o nome 'main'. Assim você cria seus próprios componentes.
A ideia dos componentes é utilizá-los em outros componentes e usar como telas existentes em sua aplicação. Ambos os casos são possíveis. A seguir veremos como usar estes componentes como nossas telas. Para isso criaremos um outro componente:

```
npm run ng generate component Second
```

3 O básico sobre Rotas

O arquivo app-routing.module.ts indica qual componente deve ser renderizado em um 'router-outlet' com base no URL da página. É simples trabalhar com ele, observe:

app-routing.module.ts

```
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3  import { MainComponent } from './main/main.component';
4  import { SecondComponent } from './second/second.component';
5
6  const routes: Routes = [
7    { path: '', component: MainComponent },
8    { path: 'second-component', component: SecondComponent }
9  ];
10
11 @NgModule({
12   imports: [RouterModule.forRoot(routes)],
13   exports: [RouterModule]
14 })
15 export class AppRoutingModule { }
```

Nas linhas 7 e 8 eu defino que o MainComponent deve ser renderizado na página principal, ou seja, na http://localhost:port/, e a segunda página http://localhost:port/second-component. Para criar uma navegabilidade iremos estruturar as páginas da seguinte forma:

main.component.html

```
1  <p>main works!</p>
2  <a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a>
```

second.component.html

```
1  <p>second works!</p>
2  <a routerLink="" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a>
```

Assim podemos fazer uma navegabilidade simples mas funcional.
Note que tudo está sendo renderizado no app.component.html. Se alteramos esse aquivo colocando alguma informação como, por exemplo, passando as tags 'a' de seus componentes para o appcomponent.html, poderemos criar uma guia de navegação:

app.component.html

```
<nav>
    <ul>
        <li>
            <a routerLink="" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a>
        </li>
        <li>
            <a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a>
        </li>
    </ul>
</nav>

<router-outlet></router-outlet>
```

Note que este 'nav' ficará em todas as páginas pois ela está no app. Tudo será renderizado logo abaixo da navegação.
Por organização, você ainda poderia separar a navegação em mais um componente:

```
npm run ng generate component Nav
```

nav.component.html

```
<ul>
    <li>
        <a routerLink="" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a>
    </li>
    <li>
        <a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a>
    </li>
</ul>
```

app.component.html

```
<header>
    Meu site
</header>

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

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

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

Note que o nome do componente é app seguido do nome do componente.

4 O básico sobre Binding e Templates

A comunicação do Angular com a tela e os eventos pode ser complexa de se compreender. É possível associar valores dentro da classe TypeScript e a tela. Enquanto o Template permite a construção de telas mais facilmente e o Binding a conectar eventos e informações ao código de comportamento do componente.

main.component.html

```
<section>
    <p>Bem-vindo ao meu sistema!</p>
</section>

<section>
    <p>
        <input (input)="update($event)">
    </p>
    <p>
        <button (click)="onClick()">Salvar</button>
    </p>
</section>

<section>
    <p>
        Texto Salvo: {{savedText}}
    </p>
</section>
```

main.component.ts

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

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.css']
})
export class MainComponent {
  text = "Altere aqui..."
```

```
10    savedText = ""
11
12    onClick()
13    {
14      this.savedText = this.text
15    }
16
17    update(event:any)
18    {
19        this.text = event.target.value
20    }
21  }
```

Os parêntesis ao redor de 'input' apontam um biding que levará dados da tela (view) para o código (data source). Assim quando um texto é digitado a função update é chamada e atualiza a propriedade text que o componente tem internamente. Ao clicar em Salvar, temos outro biding semelhante que chamará o onClick no componente colocando o texto no savedText. Quando savedText é atualizado, a última secção que utiliza chaves duplas atualiza seu valor. Essas chaves duplas é uma interpolação de texto, isso significa que o que estiver na variável savedText é apresentado na tela. Note que .html e o .ts conversam de diversas formas.
Note que dentro das funções TypeScript podemos utilizar JavaScript puro.

5 Próximos passos

Nas próximas aulas iremos ir a fundo nos tópicos que apresentamos nessa aula. Componentes, Rotas, Templates, Binding além de outros tópicos como formulários e bilbiotecas de componentes. Num primeiro momento, este é o overview necessário para se compreender como o Angular conversa e atual sobre os dados.

6 Exercícios

Faça um sistema de utilidades com uma calculadora, um conversor de temperaturas e um conversor de binário para decimal. Utilize as mesmas estruturas e recursos vistas nessa aula introdutória e crie um css para deixar seu sistema apresentável.