**CALCULADORA**

Procedimentos pós criação

Após criar o projeto com o comando “ng new calculadora”, executar o comando ng server para verificar se a aplicação está executando.

No arquivo de teste /e2e/src/app.e2e-spec.ts comentar a linha que realiza o teste do título da página.

  it('should display welcome message', async () => {

    await page.navigateTo();

    expect(await page.getTitleText()).toEqual('calculadora app is running!');

  });

No diretório do componente padrão, devemos comentar o teste no arquivo /src/app/app.component.spec.ts.

  it(`should have as title 'calculadora'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('calculadora');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement;

    expect(compiled.querySelector('.content span').textContent).toContain('calculadora app is running!');

  });

No arquivo /src/app/app.component.ts retirar a propriedade title do AppComponent

export class AppComponent {

  //title = 'calculadora';

}

Na view do projeto, removemos a referência da propriedade “title” -> /src/app/app.component.html

<h1>

  {{ title }}

</h1>

**IMPORTANTE**

O Angular em sua última versão habilita por padrão o modo **restrito (strict)** e **noPropertyAccessFromIndexSignature** do Typescript, e este modo gera algumas incompatibilidades com o código criado aqui por ele fazer muitas restrições quanto a criação de objetos.

Para evitar erros de compilação por causa do modo restrito, o recomendado é desabilitá-lo no projeto.

Para desabilitar o modo **restrito (strict)** e **noPropertyAccessFromIndexSignature** no Angular, edite o arquivo tsconfig.json, localizado na raiz do projeto, e altere as seguintes linhas de:

1. "strict": true,
2. "noPropertyAccessFromIndexSignature": true,

Para:

1. "strict": false,
2. "noPropertyAccessFromIndexSignature": false,

Pronto, feito isso nenhum erro de compilação por causa deles serão gerados.

**Criando o módulo Calculadora**

O angular é modular, ou seja, é formado por um conjunto de módulos. Cada módulo ele vai representar uma unidade ou algo específico dentro da aplicação e todos os módulos vão ficar dentro do módulo da aplicação, ou seja, o módulo principal. Iremos criar apenas um modulo e ele será o único desse projeto que fará parte do todo o módulo da aplicação.

Comando: ng g module calculadora

@NgModule({

  declarations: [],

  imports: [

    CommonModule

  ]

})

Como ele não é um módulo da aplicação e ele é um sub-módulo da aplicação principal. Nós devemos importar CommonModule (imports), ele possui alguns utilitários necessários para a criação desse módulo. Em "declarations" onde nós iremos colocar os nossos componentes que farão parte do módulo.

Vamos associar esse módulo ao módulo principal da aplicação. Em /src/app/app.module.ts importamos o módulo calculadora.

import { AppRoutingModule } from './app-routing.module';

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

import { CalculadoraModule } from './calculadora/calculadora.module';

Depois importamos este módulo para dentro deste módulo através da chave "imports".

  imports: [

    BrowserModule,

    AppRoutingModule,

    CalculadoraModule

  ],

No diretório /src/app/calculadora criamos o arquivo index.ts com o alias para este modulo com o seguinte conteúdo:

export \* from './calculadora.module';

Isso é um comando do TypeScript que ao executar o comando acima sobre a classe em questão, ela se torna pública para quem acessar o diretório. Após isto podemos simplificar o import desta classe no arquivo /src/app/app.module.ts substituindo o comando como segue abaixo:

DE import { CalculadoraModule } from './calculadora/calculadora.module';

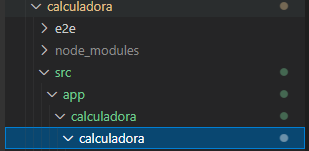
PARA import { CalculadoraModule } from './calculadora';

**Criando o componente Calculadora**

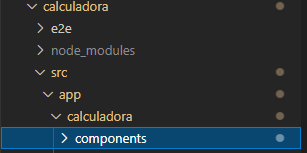
Assim como o módulo seria um domínio que nós estamos tratando de forma individual, o componente ele seria uma unidade desse domínio. Podemos ter vários componentes em um único modulo. Então é importante que o componente ele faça uma coisa específica e que ele seja uma parte dentro desse módulo, uma parte específica de uma subdivisão desse módulo, aqui na calculadora que é um projeto simples, pois necessita apenas de um módulo e então não fica claro a importância dos componentes, pois teremos um único componente.

Comando: ng g component calculadora/calculadora

Após criar o componente, será criada a pasta Calculadora, para organização do projeto vamos renomear para componentes



Para



Criamos um alias para os componentes através do arquivo index.ts no diretório /src/calculadora/components com o conteúdo abaixo:

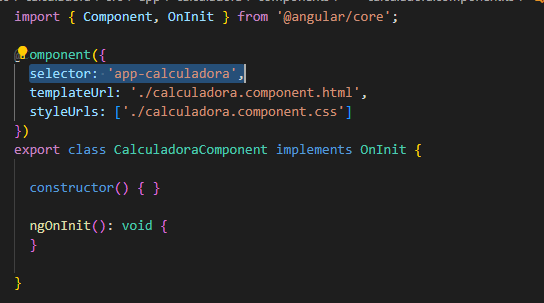
export \* from './calculadora.component';

E aplicamos o alias no módulo calculadora /src/calculadora/ calculadora.module.ts

DE import { CalculadoraComponent } from './components/calculadora.component';

PARA import { CalculadoraComponent } from './components';

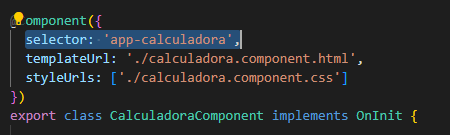
O componente (calculadora.component.ts) possui a tag @Component que gera uma tag HTML, através atributo "selector".



Então essa é uma nossa tag HTML e podemos acessar ela de outros arquivos HTML. Nas propriedades templateUrl e styleUrl temos a página do componente e seu estilo.

Agora vamos exibir esse componente na nossa aplicação e no arquivo calculadora.component.html é que colocaremos o código da calculadora, o qual é o HTML dela.

Para isso nós devemos editar o componente HTML da aplicação (app.component.html) que está sendo responsável em exibir o HTML no navegador. Iremos usar a marcação dada pela propriedade "selector" do componente.



Basta adicionar a tag no arquivo app.component.html conforme abaixo:

<app-calculadora></app-calculadora>

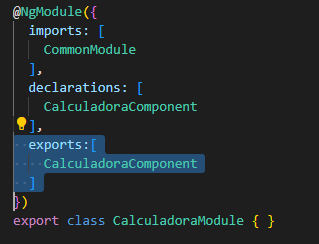
Mas é necessário um ajuste para que o HTML do componente seja visualizado.

Como nós temos uma hierarquia de módulos então app.component.html está tentando exibir um componente desconhecido para ele.

Quem tem a responsabilidade de informar ou tornar esse módulo visivel é o calculadora.module e ele não está pronto para fazer isso.

A tag <app-calculadora> é desconhecido pelo módulo da app.

Editamos o arquivo (calculadora.module.ts) e adicionamos o exports para exportar o componente calculadora:



Quando o app.module.ts faz o import do module calculadora



E o modulo calculadora informa a existência do componente calculadora e a tag <app-calculadora> será utilizada para renderizar a tela. Como o modulo da app está associado ao componente padrão e neles que definimos a tag <app-calculadora> então a aplicação consegue acessar este componente

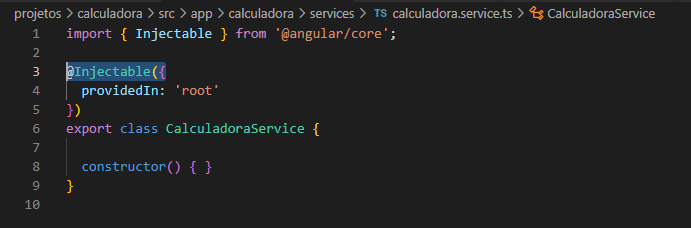


**Criando o serviço Calculadora**

O serviço ele é uma classe normal e a ideia do serviço é que ele mantenha a lógica de negócio na nossa aplicação. Então é importante você fazer uma diferenciação como essa é uma diferença básica em questão de navegação ou regras de negócio entre o componente e serviço. No componente a ideia é que ele tenha o acesso a view e ele faça a intermediação ou cuide da navegação, da comunicação entre view e a regra de negócio. Ele seria muito parecido com um controle. No angular a nossa classe de negócio que vai conter todos os códigos, as regras de negócio, chamadas a api's é um serviço. Vamos criar um serviço de calculadora quem será o responsável por executar as operações no nosso modelo de calculadora o controller saberá fazer a navegação e ele vai estar associado com a view e a medida que as operações forem sendo executadas como soma, subtração ele irá transferir para o serviço para efetuar o cálculo da operação solicitada.

Comando: ng g service calculadora/services/calculadora

O serviço também é uma classe chamada CalculadoraService e o que torna ele um serviço é a notação @Injectable



Isso significa que ele consegue prover esta classe para outras classes de forma automática que é o que nós chamamos de injeção de dependências. Basicamente quem irá consumir esse serviço é o nosso componente da calculadora. Então vamos informar ao angular que o componente calculador necessita utilizar um serviço através dessa anotação injectable, ele é capaz de prover o serviço automaticamente sem a necessidade de criar manualmente a instância do serviço.

Criamos um alias para o service através do arquivo index.ts no diretório /src/calculadora/services com o conteúdo abaixo:

export \* from './calculadora.service';

Também podemos exportar os components e services no arquivo index.ts no diretório /src/calculadora conforme abaixo:

export \* from './calculadora.module';

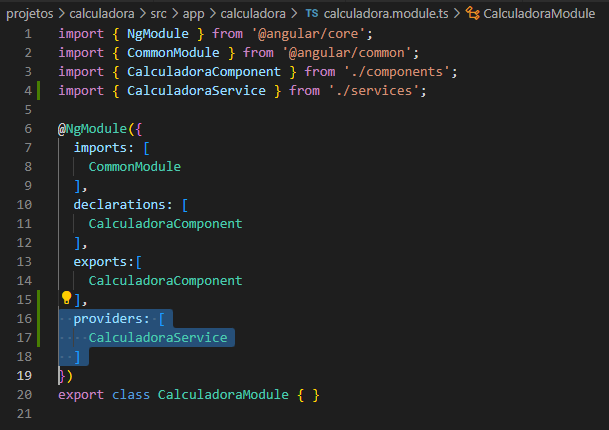
export \* from './components';

export \* from './services';

Para registra o serviço, devemos editar o arquivo src/calculadora/calculadora.module.ts conforme abaixo:

import { CalculadoraService } from './services';

e adicionar a propriedade “providers”



E no componente iremos fazer a injeção de dependência, basta editar o arquivo src/calculadora/components/calculadora.components.ts e importar o service com o comando abaixo:

import { CalculadoraService } from '../services';

E a injeção de dependência ocorre no construtor do component, ao declarar uma variável no construtor, o typescript irá criar esta variável como uma propriedade de nossa classe. Eg.: veja a variável “calculadoraService”

export class CalculadoraComponent implements OnInit {

  constructor(private calculadoraService : CalculadoraService) { }

  ngOnInit(): void {

    this.calculadoraService

  }

}

Quando o angular chamar o cliente calculadora componente ele automaticamente vai criar uma instância do service.

**Implementando o serviço CalculadoraService**

Agora iremos implementar a regra de negócio do Service, que são os cálculos aritméticos aplicados no CalculadoraService (calculadora.service.ts).

**Testando o serviço CalculadoraService**

Antes de utilizar o nosso código, iremos primeiramente criar uma classe de teste para validar se o código está correto. Iremos usar apenas os testes unitários para testar a lógica de negócio. (calculadora.service.spec)

E antes de fazer o teste do serviço, nós temos que primeiramente corrigir esses testes. O nosso app-component não está carregando o módulo de calculadora. Se olharmos o app.module.ts, vemos a importação do CalculadoraModule:

 imports: [

    CalculadoraModule

  ]

E no caso do app.component.spec.ts, devemos trazer a mesma informação. No teste do angular é feito pela classe TestBed, devemos chamar o método configureTestingModule, fazendo que um contexto de teste seja criado.

Importamos o modulo da calculadora:

import { CalculadoraModule } from './calculadora';

E no método configureTestingModule definimos o modulo:

    await TestBed.configureTestingModule({

      imports: [

        RouterTestingModule,

        CalculadoraModule

      ],

      declarations: [

        AppComponent

      ]

    }).compileComponents();

Agora temos que importar o Serviço calculadora para o módulo de testes e no mesmo método configureTestingModule definimos o serviço:

    await TestBed.configureTestingModule({

      declarations: [ CalculadoraComponent ],

      providers: [ CalculadoraService ]

    })

**Testando o CalculadoraComponent**

Vamos criar o último teste de calculadora para fazer um teste unitário para o componente. Esse teste ele é um pouco mais complexo do que o serviço, porque o serviço se baseia muito em executar uma função da classe do serviço.

A calculadora componente ele já vai simular algumas operações de usuários já interagindo com o sistema, então eu vou simular uma conta sendo realizada na calculadora clicando nos botões e verificando o resultado. (calculadora.component.spec.ts)