**primeira aula, baixando o projeto inicial**

Por algum motivo, não iremos fazer o projeto completamente do 0, o que me deixa puto xD

deve ser baixando alguma coisas prontas do repositório

serão baixados os repos:

# meat-app-starter

Depois de baixado, é preciso dar um npm install na pasta raiz dessa aplicação para baixar as dependências

tive que rodar esse comando pois estava tendo um erro bolado

npm config set registry http://registry.npmjs.org/

xD

Após isso atualizei o angular cli com o comando

npm install --save-dev @angular/cli@latest

=======================================================================

Em outra aula, foi componentizado o header (que já estava no projeto) e foi incluído um corpo da index do projeto de template no projeto meat construído em angular.

nada novo xD

=======================================================================

Rotas e navegação entre componentes

Na maioria das aplicações existe algo que é fixo, e algo que muda de acordo com a navegação do usuário

Ilustrando isso

Na maioria das páginas existe um header (cabeçalho) fixo, e tem um conteúdo que muda de acordo com a navegação

De acordo com q a url muda, só muda o corpo. a cada url, um corpo novo

Para colocar um conteúdo dinâmico na página, é necessário uma tag chamada router-outlet que marca uma região de conteúdo dinâmico .

Para isso é necessário mapear cada componente em uma rota.

Rotas são basicamente um array que contém um mapeamento de cada caminho pra um component, rotas aceitam parâmetros usando a sintaxe de dois pontos seguido do nome do parâmetro

Dando exemplo de uma rota:

export const ROUTES: Route = [

{ path: ‘’, component: RestaurantsComponent },

{ path: ‘restaurant/:id’, component: RestaurantComponent },

{ path: ‘about’, component: AboutComponent }

];

Por fim , é preciso indicar que rotas são usadas em um módulo atráves da função forRoot na raiz ou forChild através de outros módulos(será visto em outro momento no curso).

exemplo:

@NgModule({

declarations: [...],

imports: [ …, RouterModule.forRoot(ROUTES) ],

})

export class AppModule

Como os caminhos são acionados? como consigo navegar?

O módulo de roteamento disponibiliza uma diretiva chamada routerLink onde podemos passar um caminho a ser acionado ou um conjunto de parâmetros.

Podem ser usadas das seguintes maneiras

<a routerLink=”/restaurants”> Restaurantes </a>

ooou

<a [routerLink]=”[‘/restaurants’]”> Restaurantes </a>

Criando as primeiras rotas

Pra começar a trabalhar com roteamento, nesse momento é preciso criar mais um component, que será o component da página sobre (about), que terá um conteúdo fixo e depois disso será visto como se define rotas e como navegar. É importante isso ser feito agora pois uma aplicação angular depende mt de rotas e roteamento, pois tem vários components que serão usados

Criando o component about

será copiado o conteúdo do projeto de template para o component criado .

Feito isso, é hora de começar a criar as rotas

Vamos começar a criar as rotas em um arquivo separado (o cara não vai falar sobre modulos ainda porque não está afim)

será criado um arquivo chamado app.routes.ts na pasta app, que será declarado as rotas.

Para usar as rotas, é necessário importar o módulo de rotas: Segue import

import { Routes } from "@angular/router";

feito isso pode ser definido o conteúdo do array Routes.

o documento nesse momento fica assim

*import { Routes } from "@angular/router";*

*import { HomeComponent } from './home/home.component';*

*export const ROUTES: Routes [*

*{path: '', component: HomeComponent}*

*];*

Quando um compoent é exportado, ele precisa ser importado, então por isso que importei o componente

Depois de criado esse arquivo, deve-se importar ele no modulo principal, lembrando de importar o arquivo onde está o ROUTES

Depois de importar isso, deve-se declarar esse ROUTES como argumento de um método forRoots do import RouterModule, ficando assim:

RouterModule.forRoots(ROUTES)

Depois disso, deve-se ir no template do componente principal, e deve-se usar a diretiva <router-outlet>, para marcar a região dinâmica do template.

Para usar links que darão acesso ao template dos outros componentes, deve-se trocar os links por uma diretiva que acessa aos routers. Como é essa diretiva?

<a routerLink=”/restaurants”> Restaurantes </a>

ooou

<a [routerLink]=”[‘/restaurants’]”> Restaurantes </a>

Também temos uma diretiva pra indicar que quando o link estiver ativo, será ativado alguma classe

A diretiva é o routerlinkActive, que pode ser usado da seguinte forma

<li routerlinkActive="active">

Quando esse routerLink estiver ativo, a classe active também será ativada sobre o list item.

(Não entendi direito o routerLink, assistir depois novamente xD)

Criando o componente de Restaurantes

Nessa aula, será criado o componente de restaurants, será criado outros componentes individuais, será criado também serviços

Nessa aula será criado o component de restaurants. O conteúdo também será copiado do projeto template.

Primeira coisa a se fazer é arrumar as rotas. Existe um link na página principal que leva para o componente de restaurants e também há um link no header que leva para esse componente.

Então, depois de criado o componente, esse é cadastrado nas rotas e é adicionado os links com routerLink nos componentes header e home.

Próximo passo é criar componentes individuais para cada item da página restaurants, pq, se observado, as informações se repetem entre os itens.

Por ter informações repetidas, esses itens dessa página são fortes candidatos a ser componentes

Vai ser criado um componente individual que representa os dados individuais de cada restaurant.

Esse componente será criado na mesma pasta de restaurants, pq o professor quer q os componentes estejam próximos aos componentes relacionados.

Como isso será feito?

Para as versões recentes do angular cli, pode ser feito da seguinte maneira:

ng g c restaurants/restaurant

Será criado um componente dentro do outro da pasta de outro component

Criado o componente de um restaurant individual, será observado o componente restaurants.

Lá, pode ser visto que pra cada restaurante as informações link da imagem, nome, estrelas, detalhes e tempo de espera sempre se repetem. Esse pedaço vai ser tirado do componente de restaurants e será levado para o componente restaurant.

Esse componente individual precisa dos dados do restaurant para poder trabalhar, então o que pode ser feito? Criar uma propriedade com nome de restaurant e passar as propriedades desse elemento para o componente individual, em algum momento essas informações serão passadas para esse componente

Nesse momento, será criado uma model que identifique os dados que serão passados, e essa model será utilizada no componente com o decorator input, para que ele possa receber esses dados de algum lugar.

Primeiro, será criado o model, que será um arquivo chamado restaurant.model.ts, que irá declarar uma interface definindo um restaurante.

Esse modelo que será usado como tipo do array de restaurantes criado no componente restaurants; lembrando que precisamos importar o tipo lá em cima, por que estamos lidando com es2015, exportou de um lado, importa do outro.

Após isso, é preciso mecher com o template do componente, usando de template interpolation o property binding para vincular as propriedades do restaurante que serão passadas para o componente ao template.

O caminho ainda não será criado.

Agora é preciso mudar a página de restaurants, pois os restaurants serão colocados lá usando um \*ngFor. Como ainda não aprendemos sobre serviço, esses restaurants devem vir da memória, então deveremos mexer na classe do componente.

Depois de criado o array com alguns restaurants (no caso 2, que pegamos do arquivo db.json no projeto) será modificado o template, agora será usado o componente restaurant no lugar dos vários restaurants que foram usados de maneira fixa.

Será usado de fato o \*ngFor.

Algumas considerações importantes. O ngFor deve ser declara da tag que irá se repetir, usando uma iteração com of (recurso ES6).

Aquela página que antes estava bem poluída ficou simples, pois agora só é preciso usar essa linha para repetir tudo

<div \*ngFor="let restaurant of restaurants" class="col-sm-6 col-xs-12">

<mt-restaurant [restaurant]="restaurant"></mt-restaurant>

</div>

Lembrando de uma coisa

No componente restaurant, ele espera receber um input, um objeto que siga a interface Restaurant. Nessa sintaxe, o [restaurant] é um meio de definirmos aquela propriedade usando property binding, e o argumento passado é um item da iteração do ngFor. Veja bem que o nome é o mesmo no ngFor e na declaração da property binding.

O que é injeção de dependência?

Injeção de dependência é um padrão de projeto onde é possível deixar de instanciar os objetos manualmente e passa a depender do framework para obter os objetos que a aplicação precisa

O framework gerencia a instanciação dos objetos e também suas dependências e disponibiliza isso para os componentes da aplicação

Segue exemplo:

Se um componente precisa utilizar um serviço, podíamos instanciar isso manualmente. Mas com injeção de serviços, podemos passar isso no construtor do componente, e o framework vai instanciar esse serviço quando ele quiser

Ao inverter essas responsabilidades, o código fica mais testável e mais limpo

Para deixar um serviço disponível para ser injetado em um componente, esse deve ser declarado na lista de providers e devidamente importando em um componente ou modulo

Se for em um componente, o serviço fica disponível para o componente ou seus filhos.

Se for declarado no modulo principal, esse serviço fica disponível para todos componentes da aplicação.

O que é um serviço?

Serviços são classes comuns em angular que você pode usar para injetar em componentes ou em outros serviços. Serviços são geralmente usados em uma aplicação para encapsular o acesso a api de beck end, isso vai acontecer muito em uma aplicação angular

Serviços podem ser singletons, ou seja, só ter uma instancia deles, e são ótimos candidatos para guardar dados compartilhados que servem toda a aplicação (cuidado com isso), se quiser, também pode ser usado serviços para suprir somente parte de uma aplicação.

No geral, são 3 escopos que podem ser utilizados para declarar um serviço.

~> Módulo angular, onde serão declarados na lista de providers. Se declarado no modulo raiz, ficará disponível para todas as classes disponíveis no mesmo módulo, componentes e outros serviços também. Todos eles irão compartilhar a mesma instancia desse serviço.

Para outros módulos que não sejam o módulo raiz existe algumas particularidades que serão discutidas mais à frente.

~> Componentes e seus filhos ~> se declarado na lista de providers de um componente um serviço vai ser instanciado e compartilhado apenas para o componente e seus componentes filhos. Mesmo se um serviço de tipo idêntico existir como provider em um modulo o componente receberá uma outra instancia.

~> Somente Component ~> se declarado na lista de viewProviders de um componente, esse serviço será usado somente só no componente, não podendo ser usado nos seus filhos.

Serviços também podem ser solicitar injeção de outros serviços, para isso ele deve declarar o decorator @Injectable(), mas atenção, esse decorator não é necessário para um serviço que será injetado em outro objeto, apenas para receber injeções do framework.

Esse serviço é importado da seguinte maneira:

import { Injectable } from ‘@angular/core’

Segue um exemplo

import { Injectable } from ‘@angular/core’

import { Http } from ‘@angular/http’

@Injectable()

export class MyService{

constructor (private http:Http){ }

list(){

return this.http.get(‘/url’);

}

}

Nesse exemplo vemos um serviço marcado com injectable recebendo um serviço do framework, que é o http.

Alguns serviços que o próprio framework fornece:

Title, http e router.

http ~> encapsula o acesso a http

router ~> realiza a navegação de forma programática

Será falado sobre eles em outro momento

Title ~> serviço para obter e alterar o título de uma página

O componente pode requisitar a injeção e usar o método setTitle para substituir para o título que quiser.

Esse serviço é necessário pois não é possível utilizar expressões angular na página html inteira, pois o título fica no header, e essa parte não faz parte do bootstrap, foi criada essa alternativa.