



En la carpeta que deseo me posicioneo con el cmd y ejecuto

ng new my-app

para lanzar el serdor de angular ejecuto

ng serve –open

la etiqueta que se visualiza en el index.html

  <app-root></app-root>

Es un selector del app.compomponent.ts

Que se relaciona con código HTML desarrollado en

  templateUrl: './app.component.html',

dentro de

export class AppComponent {

  title = 'my-app-01';

}

Se genera una variable con un contenido y se lee desde el template relacionado previamente por medio de

{{ title }}

Pasar datos de un componente de padre a hijo

Decoradores input output, se los importa en la clase .ts que va a recibir los datos de otro componente

Se declara el tipi input en la clase que va a recibir datos de otra clase

*@Input*() dataEntrante:*any*;

Se asocial el input en la etiqueta del componente html que recibirá los datos

<app-card *[dataEntrante]*="card" *\*ngFor*="*let* card of cards"></app-card>

Se lee los datos que ingresan mediante las anotación {{dataEntrante.nombreDeVariable}}

Pasar datos de un componente a cualquirera

Para la creacionde un servicio se ejecuta por terminal

ng generate service servicioDeFavoritos

el servicio se pide en el constructor de la clase que se requiere

*constructor*(*private* serviceFavoritos:ServicioDeFavoritosService) { }

en el servicio se crea la anotación

*@Output*()disparadorDeFavoritos:EventEmitter<*any*>=*new* EventEmitter();

El cual es llamado en el evento que desea emitirle datos de la siguiente forma en la que se le asigna al objeto data los datos a enviar

*this*.serviceFavoritos.disparadorDeFavoritos.emit({data:*this*.dataEntrante});

luego se lo recibe en otro clase componente los datos emitidos, dentro de las llaves de subscribe se ejecuta funciones que tendrán en “data” lo enviado preciamente para ser utilizado en esta clase

*this*.serviceFavoritos.disparadorDeFavoritos.subscribe(data=>{  
 console.log('recibiendo data....',data);  
})

PETICIONES POR GET

se debe pasar como parámetro por el constructor el HttpClient en el servicio que se desea utilizar

*constructor*(*private* http:HttpClient) { }

y luego se debe agregar en el app.module.ts en los imports

HttpClientModule,

Posteriormente se crea un método en el servicio que resalice la petición tipo GET y retorne los datos de la API

*public* get (url:*string*){  
 *return this*.http.get(url);  
}

Luego en la clase que se desea obtener datos de una API se pasa como parámetro del constructor el servicio REST

Y se crea un método el cual le pase la URL del la API y la acción que realizara la promesa

*public* cargarData() {  
  
 *this*.RestService.get('http://localhost:8080/data').subscribe(respuesta => {  
 *this*.cards=respuesta;  
 });  
}

CREAR RUTAS

Se crea un componente con lo que se desee vicualizar

Luego en el app-routing.module.ts dentro de la variable se agrega tantas rutas como se desee, colocando la ruta a utilizar en el path y se le asigna a esa ruta en componente deseado en el component

*const* routes: Routes = [  
 {  
 path: '',  
 component: HomeComponent  
 },  
 {  
 path: 'detail/:id',  
 component: PostDetailComponent  
 }  
];

con : se realiza el uso de un URL dinamica

path: 'detail/:id',

posteriormente se coloca en el app.componente.html la etiqueta que permite el pasaje de las rutas por la app

<router-outlet></router-outlet>

De la siguiente forma se utiliza para hacer mención al uso de una ruta

<button *[routerLink]*="['/','detail',dataEntrante.id ]">Ver</button>

Para extraer los parametros de la ruta se realiza la instancia en el constructor de ActivatedRoute

*constructor*(*private* rutaActiva: ActivatedRoute, *private* rest: RestService) {  
}

la cual se invoca en el ngInit para extraer los parámetros pasados por GET los cuales nos lo devuelve en un json de la promesa para ser usados en lo requerido

*this*.rutaActiva.params.subscribe((paramsMaps: *any*) => {  
 console.log(paramsMaps);  
 *this*.findPelicula(paramsMaps.id);  
})

PETICIONES POR POST

En un servicio declaramos la peticion por POST medicnete un metodo que reciva la URL y el BODY, el cuel debe implementar las mismas clases que para peticiones por GET

*public* post (url:*string*,body: *object*){  
 *return this*.http.post(url,body);  
}

Luego se debe importar en el app.modules.ts en imports el siguente modulo para formularios reactivos

ReactiveFormsModule

Luego se debe crear la variable que almacenara el formulario como publica y en el constructor inicializarla por medio del formBuielder group, decharando los campos que tendrá en json que se enviara al componente

*public* forms;  
  
*constructor*(*private* formBuilder: FormBuilder) {  
 *this*.forms = *this*.formBuilder.group({comentario: ''});

}

Posteriormente se utiliza el forms creado en el servicio del componente asignándolo en el formsGroup, para almacenar los datos ingresantes de la vista con el mismo nombre que fue creado cada campo del json

<form *(ngSubmit)*="enviarComentarios()" *[formGroup]*="forms">  
 <input *type*="text" *formControlName*="comentario">  
  
 <button *type*="submit">Enviar</button>  
</form>

para enviar el comentario se declara una función (ngSubmit)="enviarComentarios()" que enviara a la URL deseada el body con los datos, posteriormente se reinicia el formulario para limpiar los campos

enviarComentarios() {  
  
 *this*.rest.post('http://localhost:8080/comentar',*this*.forms.value).subscribe(peli => {  
 })  
 *this*.forms.reset();  
  
}

DATA TABLE

<https://swimlane.gitbook.io/ngx-datatable/readme/installing>

<https://swimlane.github.io/ngx-datatable/#bootstrap>

Se instala la librería con

npm i @swimlane/ngx-datatable --save