Imstall node from website Take LTS or current. Open command prompt check node installed with node v and npm v commands. Npm used to get angular cli and dependancies.

Install visual studio code and/or webstorm

Run folllowing command to install angular cli.

npm install -g @angular/cli

Type ng --version to see if angular cli up.

Ng new first-agnualr-app from the command line or use new angular cli project from webstorm.

After project generated open terminal in project using ALT-F12

Ng serve --open to open the app on <http://localhost:4200>

Will cretae the following file for the app an each component

app.component.css - Pu ant of the css foe this component here.

app.component.html - This is the template for the file. Takes interpolated values fron component via {{filedName}}. This is where you use ngfor \* and ngIf \* etc. All bootstrap html will go here.used for the vie along with css

app.component.spec.ts - This is the test case for the component.

app.component.ts - This is the component data. You export class with variables to be used by the view and to be bound by the template view.

app.module.ts

ng generate component myComponent

Generates following files aand updates app.module.ts with the new component.

my-component

my-component.component.css - view layer css

my-component.component.html - view layer

my-component.component.spec.ts

my-component.component.ts - bindings to obtain data and add to the view

Copy html from bootstrap striped list on w3schools and display in html portion.

Add tag to where you want to add it with just <my-component><my-component>

Create new component user to show databinding. Must create class user.ts to hold data.

Export class User{

Id : number;

Name: string;

Email: string;

Mobile : number;

}

Need to add to user.component to create the binding.

Import {User} from ‘./user’

user : User = {

Id:123456,

Name: “…”,

…..,

}

Need to add the html for table to the view component. Then add the interpolated values to the field instead ie {{user.id}} to get the binded data.

Generate product component . ngFor directive used for list of objects. Can reference in onInit at products : any which basically says it can be any type. Then need to init

products = [{“id”:1234,”brand”:”tsshirt”,”name”:”Polo 1”……},<repeat for more>}]

Then need to copy the list to the html and add directive to row <tr ngFor\*=”let product of products”>

Then need to do ng serve --open to reload changes.

Add <app-product> selector to app-component.

Use ngIf driective to conditionally show components based on data being present or now.

i.e <th \*ngIf=”products.length > 5”

Two way data binding is where the value is reflected on screen from model updateand also in the model from a screen update.

Need FormsModule

In app-module.ts add

import {AppModule} from angular/forms

Add FormsModule to @NGModule imports tag.

Add new field called message:any = “HelloThere”

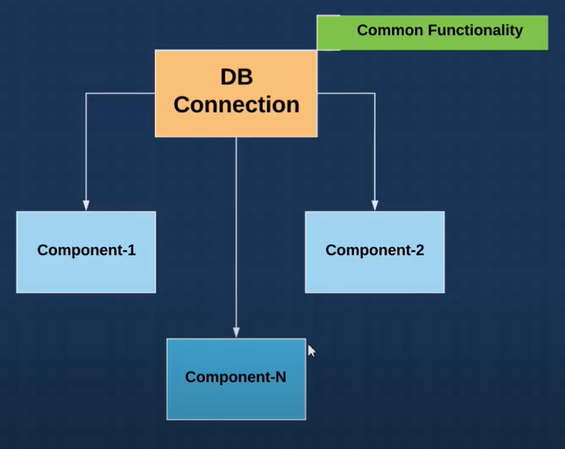
Add too new div {{message}}

Add <input type=text [value]=message> this will bind value one way to field but not update input. To update in put you need ngModel directive as follows

<input type=text [(ngModel)]=message>

Now on updating both screen and the model value get updated.

Service and dependancy injection. If angular app wants to contact a back end db andall components want to connect to the same back-end db logic can be duplicated. The ng generate servie UserService solves this by creating a class that can perform the logic and return the data to the components.



After ng g s Product must import into constructor constructor(private service: ProductsService) then can use in ngOnInit. You must add the services to the providers tag in the app.module.ts and import it into the component. It creates the service at another level outside of components.

Import httpclientmodule under browsermodule for rest requests. Create component users component for users for jsonplaceholder data. Set constructor(private http:HttpClient) to inject.

Call with http.get(url) to get resp. Then because reactive, call resp.subscribe((data)=>console.log(data)) to log the response. Include your component in the app component..

Declare user : any to replace the consolee log and assign data=users. You then open html for users get a table feo w3c schools for bootsstrap. Populate the table using the ngFor directive to list the users.

Create bookservice boot microservice, use initializer. Select web and lombox only for time being. Create packages controller and model. Create book in model along with Data noArgsConstructor and AllArgsContructor annotations. Create BookController and annotate wit @RestController. Creat single getBooks and point that returns a list of books. Hardcode return list of books. Set @GetMapping(‘/findAllBooks’). Add application.yaml and set server port o 9090. Start the service and then call under <http://localhost:9090/findAllBooks.>

Create book-client using ng new or webstorm. In app run ng g c book to create a component fot the book.Open app component html and add app-book somewhere. Import HttpClientModule into app.module to make http request. Wire http constructor(http: HttpClient){

Then in ngOnInt Let reponse = http.get(urlBoot)

Response.subscribe(data=>console.log(data))

Run angular client and boot service and notice blocked by CORS when no console log. Add @CrossOrigin(bootRestUrl) above the get mapping. Try again to see the books loaded. Now time to display on screen.

Go to book html page and add w3c bootstrap list for books id,name,price using \*ngFor=”let book of books” then interpolate the values {{book.id}} etc. In your rest call change from log to books:any….subscribe(data=>this.books=data). Now your component will list the books.

Because others may need books, change this to a service. Ng generate service book. In the service injectable inject the httpclient in constructor as before. Then inject the service instead of the client into the book component constructor(service:BookService)

Move the http call to a getBooks method on the book service. Call the book service in th ebook component service.getBooks().subscribe(data => this.books = data)

Now boot service and angular client is complete.

ngSwicth directive allows you avoid multiple ngIf. See app todo

<td <div [ngSwitch]=”todo.completed”>

<span \*ngSwitchCase=”true”>Completed</span>

<span \*ngSwitchCase=”false”>Completed</span>

<span \*ngSwitchDefault=”true”>Completed</span>

</div>

</td>