Angular-9

**I.Introduction**

**Environmental Setup for Angular9**

**1) download and install NodeJS**

* To install "Angular9" we need "npm".
* "npm" stands for node packaging manager.
* "npm" is the tool present in "NodeJS".

**Website** : https://nodejs.org/en/download/

**file** : node-v12.16.1-x64.msi

**2) install yarn tool**

* "yarn" tool given by facebook.
* "yarn" tool used to download the libraries from GitHub.
* we will install yarn tool by using following command.

> npm install -g yarn@latest

where "-g" stands for global installation.

**3) install Angular9**

* we will install Angular9 by using following command.

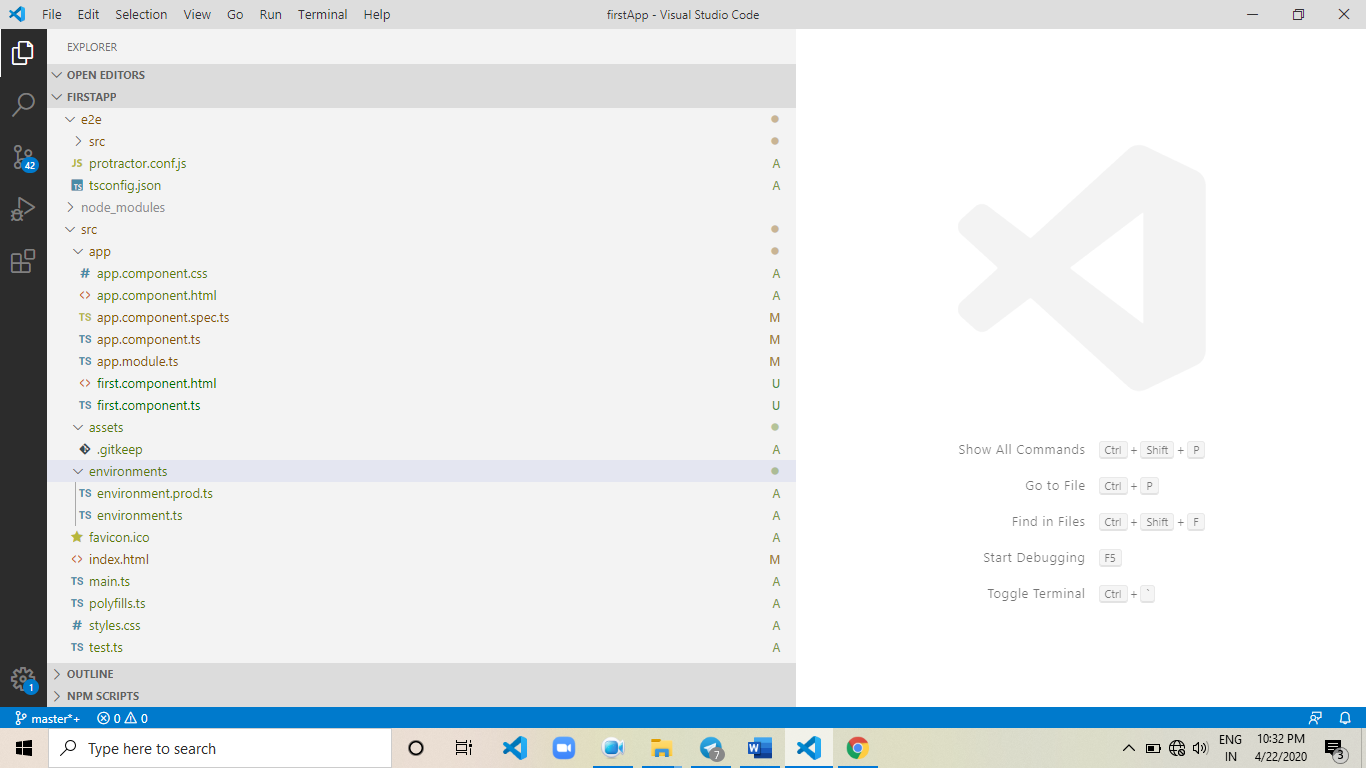
**Command:** npm install -g @angular/cli@latest

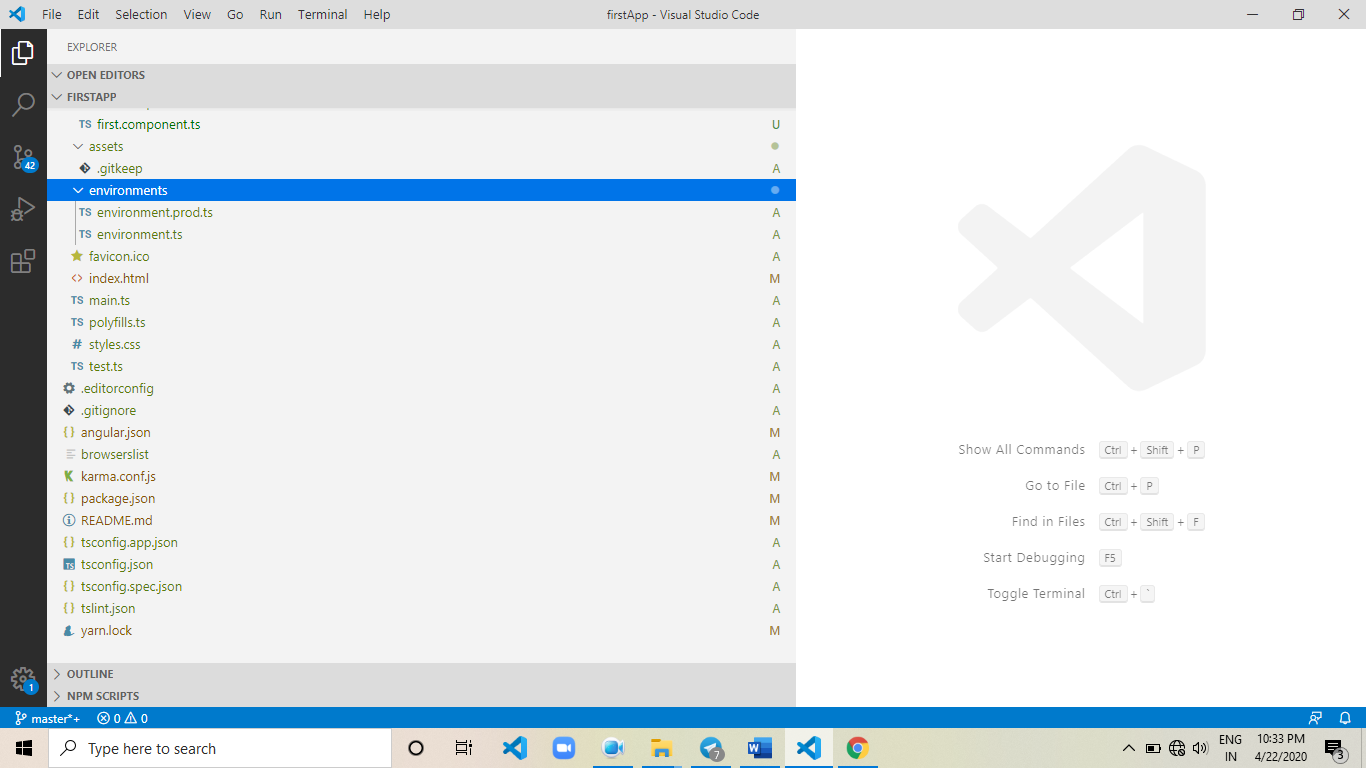
* "cli" stands for command line interface
* "cli" is the tool provided by google.
* "cli" tool used to build and execute the angular applications

**4) link the "yarn" tool to "cli" tool.**

* we will link "yarn" tool to "cli" tool by using following command.  **Command:** ng config -g cli.pacakageManager yarn
* where "M" is the capital in "pacakageManager".

**Directory structure of angular application**





**1) e2e**

* e2e stands for end to end.
* e2e directory used to write the end to end test cases to angular applications.
* in general testing divided into two types.
* Manual Testing
* Automation Testing
* Manual Testing is Deprecated, now a days no project using Manual Testing.
* Automation Testing divided into 3 Types.
* Unit Testing
* Integration Testing
* End to End Testing
* Testing particular functionality with assumptions called as Unit Testing.
* Testing particulat functionalities with exact environment called as Integration Testing (Real Environment).
* Testing Particular functionality with "end to end scenarios (production mode)" called as end to end testing.

**2) node\_modules:**

* "node\_modules" directory containes libraries.
* those libraries helps to execute the angular application.

**3) src/app:**

* this directory used to deploy the angular applications.

Ex.

Components , Directives , Services, Pipes

**4) src/app/app.module.ts**

* this file we can call registration file.
* this file also called as Module file.
* this is the Default Module in Angular Application.
* this file used to register the angular applications.
* once if we register, then only angular applications will
* be executed by angular framework

**5) src/assets:**

- this directory used to deploy the static resources

Ex.

* + images
  + multimedia files
  + xml files
  + json files

**6) environments:**

* in general we have 3 types of environments
* development environment
* production environment
* testing environment
* what ever the required environment, we will configure
* in environments directory.

**7) src/favicon.ico:**

- this is the default logo of angular.

**8) src/index.html:**

* angular starts the execution from "index.html" file.
* "index.html" file is the landing template.
* "index.html" file is the main template in angular application.
* main template internally invokes the "main.ts" file.
* "main.ts" file internally invokes the "app.module.ts" file.
* "app.module.ts" file containes our applications registrations.
* based on registrations our applications will be executed by angular framework.

**9) src/main.ts:**

- this file acting as interface between main template to registration file.

(app.module.ts <==> index.html)

**10) src/polyfills.ts:**

* polyfills.ts file is the library.
* this library helps to execute the projects into different browsers.

Ex.

Chrome, Mozilla…etc

**11) src/styles.css:**

* we will define global styles here.
* what ever the styles we define here, automatically applicable to entire angular application.

**12) src/test.ts:**

- this file representing sample testing file.

**13) editorconfig & .gitignore:**

* these two files not related to angular applications.
* first file related to "VisualStudioCode" Configurations.
* second file related "Git" configurations.

**14) angular.json:**

* this file representing directory structure of angular application.
* we can customize directory structure based on application requirement by using angular.json file.
* this file used to configure the 3rd party technologies

=> jQuery

=> BootStrap

=> ReactJS

**15) browsers list:**

* it will show supporting and non supporting browsers based on Angular9 version.

**16) karma.conf.js:**

* in general we will write unit test cases by using "karma with jasmine" tool.
* "karma.conf.js" file representing the configuration file of karma tool

**17) package.json:**

* this file used to download the 3rd party libraries.
* all these libraries downloads to "node\_modules" folder.

**18) tsconfig.app.json:**

* this file acting as controlling file for entire angular application.
* what ever the business logic written here, automatically applicable to entire angular application.

Ex.

- removing the white spaces in entire angular applications

-overcome the data redundancy in entire angular applications.

**19) tsconfig.json:**

* it contain TypeScript Configurations

**20) tsconfig.spec.json:**

* this file is the controlling file for all unit test cases present in angular project.

**21) tslint.json:**

* this file acting as validator file for angular applications.

**Chapter-1(Components)**

**Components:**

* Angular is the Framework.
* Angular Framework follows the MVC Design Pattern.
  + - M - Model
    - V - View
    - C - Component
* Simple TypeScript class behaves like Component.
* We Can Create more than one component in angular applications.
* Angular Applications are component based applications.
* Because of Components Code Reusability is high in Angular Compared to AngularJS.
* Component acting as Interface Between View and Service in MVC Architecture.
* we can establish the communication between server to database by using modules.
  + Ex.=> Mysql, mssql, mongodb,, firebase
* we can provide communication between service to server by using AJAX Calls (Observables).
* we can establish communication between component to service by using dependency injection.
* the communication between view to component called as two way data binding.

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

firstApp

src

app

first.component.ts

first.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* "first.component.ts" file used to create the component.
* "first.component.html" file used to display the component output.
* "first.component.html" also called as external template of component.
* in general we will register our applications (component) in app.module.ts file.
* index.html file is the main template.

**First.component.ts:**

* Component is predefined class available in @angular/core package
* Component class used to convert the TypeScript Standards to HTML Standards
* we will use Component class by using "@"
* Using the predefined class by using "@" symbol called as Decorator.
* Decorators are used to define th METADATA
* Data About Particular Component Called as METADATA
* Component Class constructor takes the JSON Object as Argument.
* "selector" is the json key used to define the custom HTML Element.
* we will call custom HTML Element in "index.html" file.
* "templateUrl" is the json key used to define the external template to Component.
* in general we will use external templates to display components data.
* export is the keyword in TypeScript
* export keyword used to export the components,services,directives,pipes,....
* anyone can import the exported members in angular applications

**Code:**

import { Component } from "@angular/core";

@Component({

selector:"first",

templateUrl:"./first.component.html"

})

export class firstComponent{

private mean:string;

private mern:string;

private mevn:string;

constructor(){

this.mean = "MEAN Stack...!";

this.mern = "MERN Stack...!";

this.mevn = "MEVN Stack...!";

};

public getMeanData():string{

return this.mean;

};

public getMernData():string{

return this.mern;

};

public getMevnData():string{

return this.mevn;

};

};

**First.component.html**

* this template used to display the component result (variables & functions callings)
* {{}} used to display the data on webpage
* {{}} called as expressions / interpolation / data binding

**Code:**

<html><body>

<h1 style="color: red;">{{getMeanData()}}</h1>

<h1 style="color: green;">{{getMernData()}}</h1>

<h1 style="color: blue;">{{getMevnData()}}</h1> </body> </html>

**App.module.ts:**

* app.module.ts file acting as Registration file.
* this file used to register the Components,Services,Directives,Pipes,.....
* once if we register then only our applications will be executed
* BrowserModule used to execute the projects into Browsers
* NgModule used to create the custom modules
* collection of custom modules called as project
* AppComponent is the default component
* we have four registration arrays

=>@declarations, @imports @providers @bootstrap

* we will register Components,Pipes and directives in "declarations" array
* we will register modules in "imports" array
* we will register services in "providers" array
* we will execute particular component by using bootstrap array.

**Code:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { firstComponent } from "./first.component";

@NgModule({

declarations: [

AppComponent,firstComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [firstComponent]

})

export class AppModule { }

**Index.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8"> <title>FirstApp</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

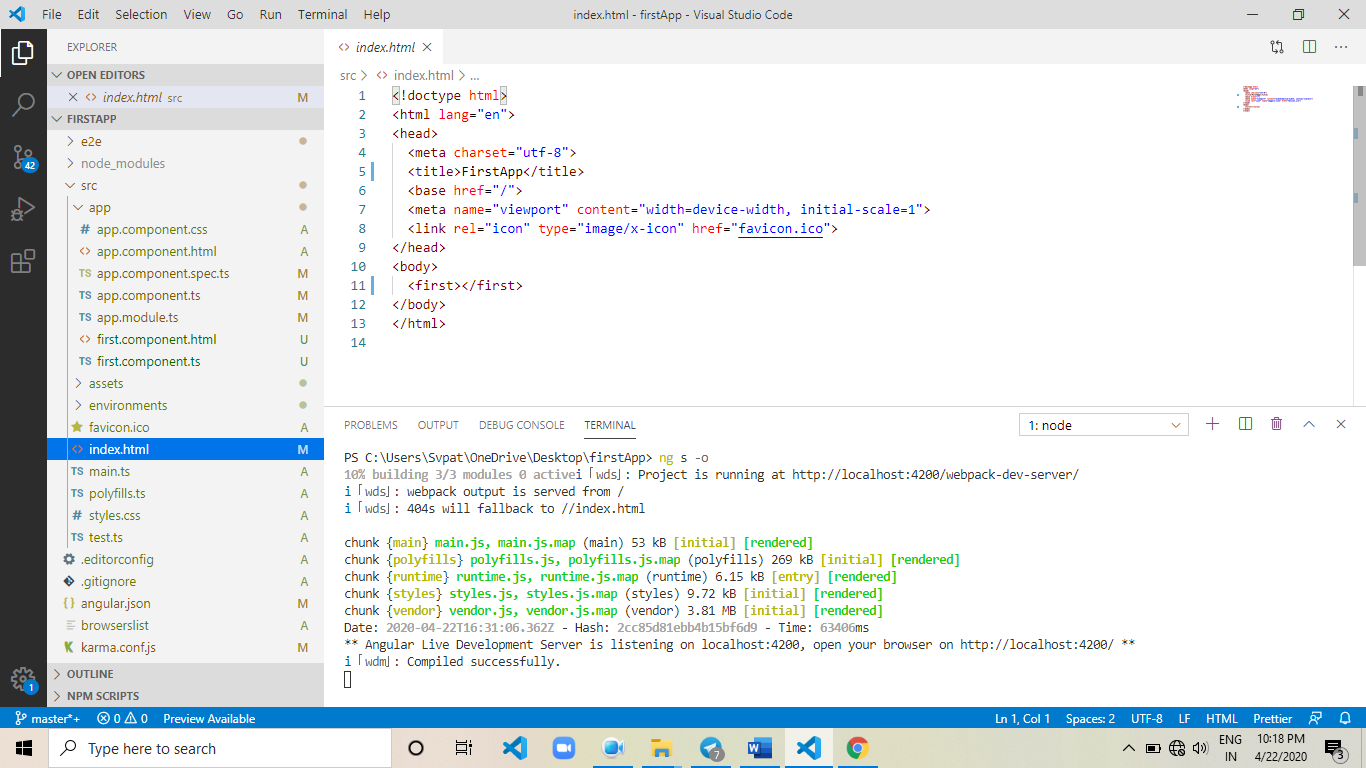
<body>

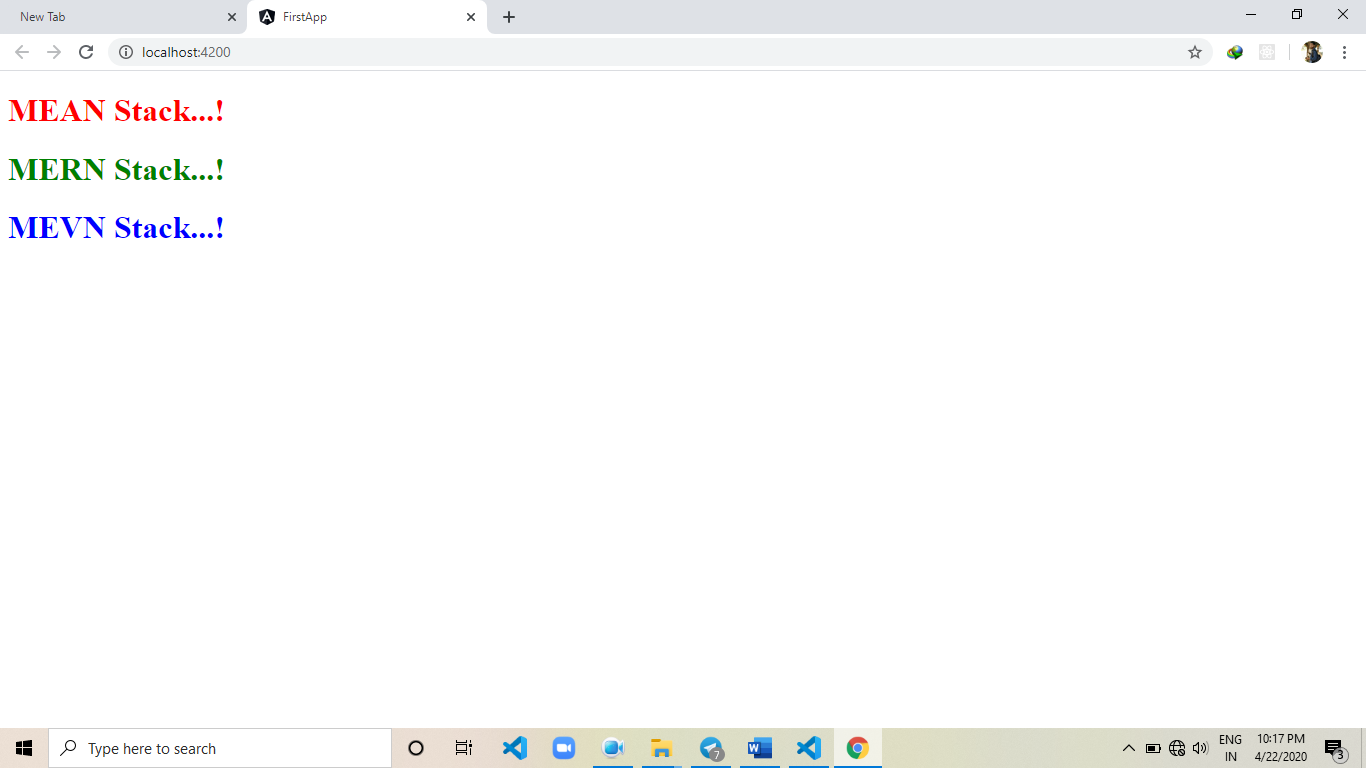
<first></first>

</body>

</html>

**Result:**





**Chapter-2(Services)**

* Services are used to share the common business logic to multiple Components.
* we have two types of Services.
* Predefined Services
* Custom Services
* The Services given by angular called as Predefined Services.
* The Services Developed by us based on Application Requirement Called as Custom Services.

**=>Custom Services:**

* Injectable is the Predefined Class, used to create the Custom Services.
* Injectable class available in @angular/core package.

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

serEx

src

app

services

db.service.ts

components

mongodb.component.ts

mongodb.component.html

mysql.component.ts

mysql.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Db.service.ts:**

//import Injectable

//Injectable used to create the Custom Service

import { Injectable } from "@angular/core";

//use Injectable

//we will use predefined classes by using "@" symbol.

@Injectable({

providedIn:"root"

})

//providedIn used to make the service as global

//providedIn facility available from Angular5 onwards

//export the class

export class dbService{

//mysqlDB()

public mysqlDB():string{

return "MySQL Data Soon...!";

};

//mongodb()

public mongodb():string{

return "MongoDB Data Soon...!";

};

};

**Mangodb.component.ts:**

import { Component } from "@angular/core";

//import dbService

//dbService containes mySQLDB() mongodb()

//our component want to call mongodb()

import { dbService } from "../services/db.service";

//use Component

@Component({

selector:"mongodb",

templateUrl:"./mongodb.component.html"

})

//export the class

export class mongodbComponent{

//declare the result variable

//result variable used to hold the result coming from dbService

private result:string;

//create the object to the dbService

//in general we will create objects by using constructors

//dependency injection

constructor(private obj:dbService){}

//ngOnInit()

//ngOnInit() method called as main method

//ngOnInit() method used to write the business logic

//ngOnInit() method called as first life cycle hook of component

ngOnInit(){

this.result = this.obj.mongodb();

}

};

**Mangodb.component.html:**

<html>

<body>

<h1 style="color: red;">{{result}}</h1> </body> </html>

**Mysql.component.ts:**

import { Component } from "@angular/core";

import { dbService } from "../services/db.service";

@Component({

selector:"mysql",

templateUrl:"./mysql.component.html"

})

export class mysqlComponent{

private result:string;

constructor(private obj:dbService){}

ngOnInit(){

this.result = this.obj.mysqlDB();

}

};

**Mysql.component.html:**

<html>

<body>

<h1 style="color: rosybrown;">{{result}}</h1>

<mongodb></mongodb>

</body>

</html>

**App.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { mysqlComponent } from './components/mysql.component';

import { mongodbComponent } from './components/mongodb.component';

@NgModule({

declarations: [

AppComponent,mysqlComponent,mongodbComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [mysqlComponent]

})

export class AppModule { }

**Index.html:**

<html>

<body>

<mysql></mysql>

</body>

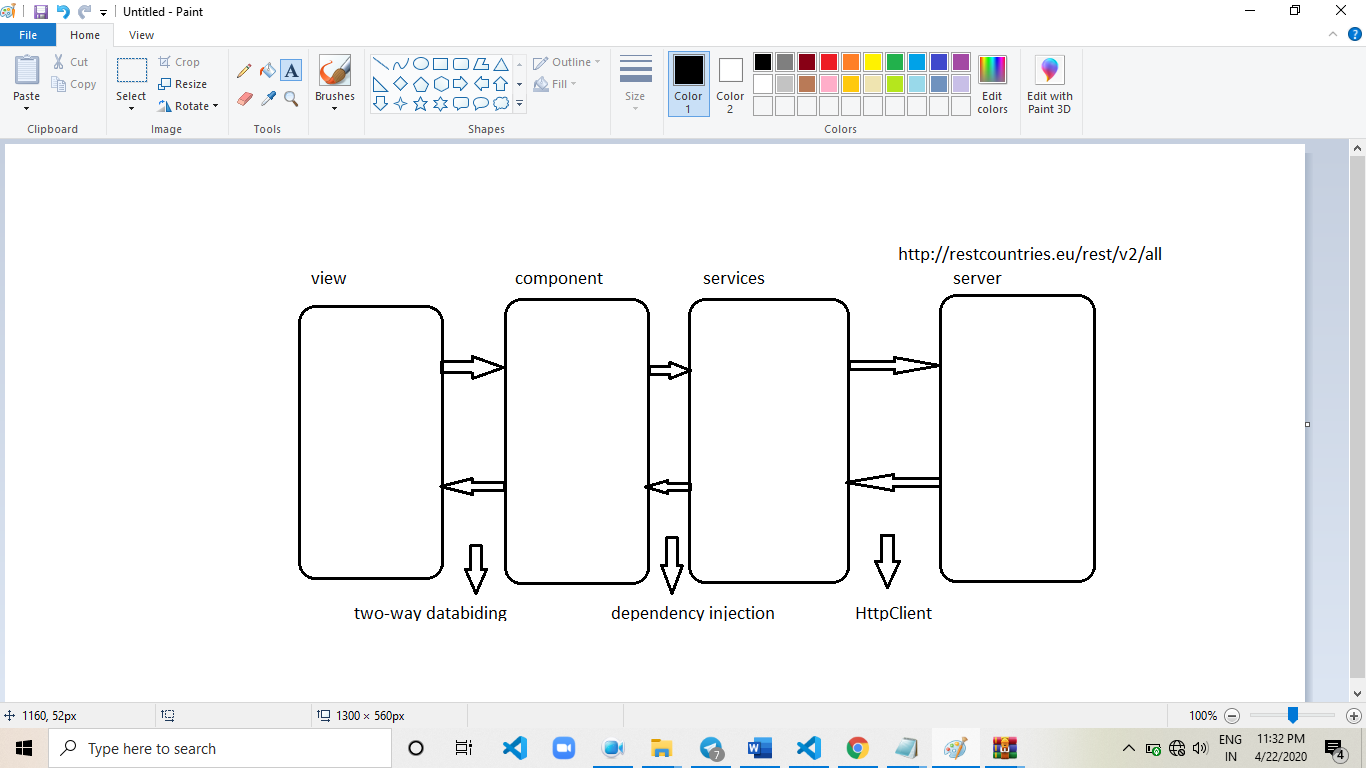
</html>

**=> Predefined Services:**

* The Services Provided by angular framework called as Predefined Services.
* "HttpClient" is the Predefined Service.
* "HttpClient" used to make the rest api calls.
* "HttpClient" present in "HttpClientModule"
* we must register "HttpClientModule" in "imports" array (app.module.ts).
* "HttpErrorResponse" is the Predefined Service.
* "HttpErrorResponse" used to handle the "Errors" thrown by servers.
* "HttpClient","HttpClientModule", "HttpErrorResponse" present in "@angular/common/http" package.
* Observable present in rxjs package.
* "rxjs" stands for reactive extension javascript.
* Observables sends the Packets (Stream of Data) in Sequence from Server to

Client.

**Diagrame:**



**Example:.**

**Directory structure**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

preSerEx

src

app

services

countries.service.ts

components

countries.component.ts

countries.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**countries.service.ts:**

//import Injectable

//Injectable used to create the Custom Service

import { Injectable } from "@angular/core";

//import HttpClient

//HttpClient used to make the rest api calls

import { HttpClient } from "@angular/common/http";

//import Observable

//HttpClient return type is Observable

//Continuous flow of data from server called as Observable.

import { Observable } from "rxjs";

//use Injectable

@Injectable({

providedIn:"root"

})

//providedIn makes the service as global

//export the class

export class countriesService{

//create the object to HttpClient

//we will create objects by using constructor

//dependency injection

constructor(private obj:HttpClient){}

//where obj is the HttpClient object

//create the function

//function should make rest api call

public getCountries():Observable<any>{

return this.obj.get("https://restcountries.eu/rest/v2/all");

};

};

**countries.component.ts:**

//import Component

import { Component } from "@angular/core";

//import countriesService

//countriesService containes getCountries()

//getCountries() returning Observable

//subscribe() used to read the data from Observables

import { countriesService } from "../services/countries.service";

//import HttpErrorResponse

//HttpErrorResponse used to handle the Exceptions thrown by server

import { HttpErrorResponse } from "@angular/common/http";

//use Component

@Component({

selector:"countries",

templateUrl:"./countries.component.html"

})

//export the class

export class countriesComponent{

//decalre result variable

//result variable used to hold the result coming from server

private result:any;

//create the object to countriesService

//in general we will create objects by using constructor

//dependency injection

constructor(private obj:countriesService){}

//where obj is the service object

//ngOnInit() is the first life cycle hook

//ngOnInit() used to write the business logic

ngOnInit(){

this.obj.getCountries().subscribe((posRes)=>{

this.result = posRes;

},(errRes:HttpErrorResponse)=>{

console.log(errRes);

});

}; };

**countries.component.html**

<!--

initially we have JSON Array

"result" is the JSON Array

"result" array containes 250 JSON Objects

Each JSON Object containes following keys

@name

@capital

@region

@population

@flag

-->

<table border="1"

cellpadding="10px"

cellspacing="10px"

align="center">

<thead style="background-color: gray;">

<tr>

<th>SNO</th>

<th>Name</th>

<th>Capital</th>

<th>Region</th>

<th>Population</th>

<th>Flag</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let x of result;let i = index">

<td>{{i+1}}</td>

<td>{{x.name}}</td>

<td>{{x.capital}}</td>

<td>{{x.region}}</td>

<td>{{x.population}}</td>

<td><img width="100px" height="50px" src="{{x.flag}}"></td>

</tr>

</tbody>

</table>

**App.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { countriesComponent } from './components/countries.component';

import { HttpClientModule } from '@angular/common/http';

@NgModule({

declarations: [

AppComponent,countriesComponent

],

imports: [

BrowserModule,HttpClientModule

],

providers: [],

bootstrap: [countriesComponent]

})

export class AppModule { }

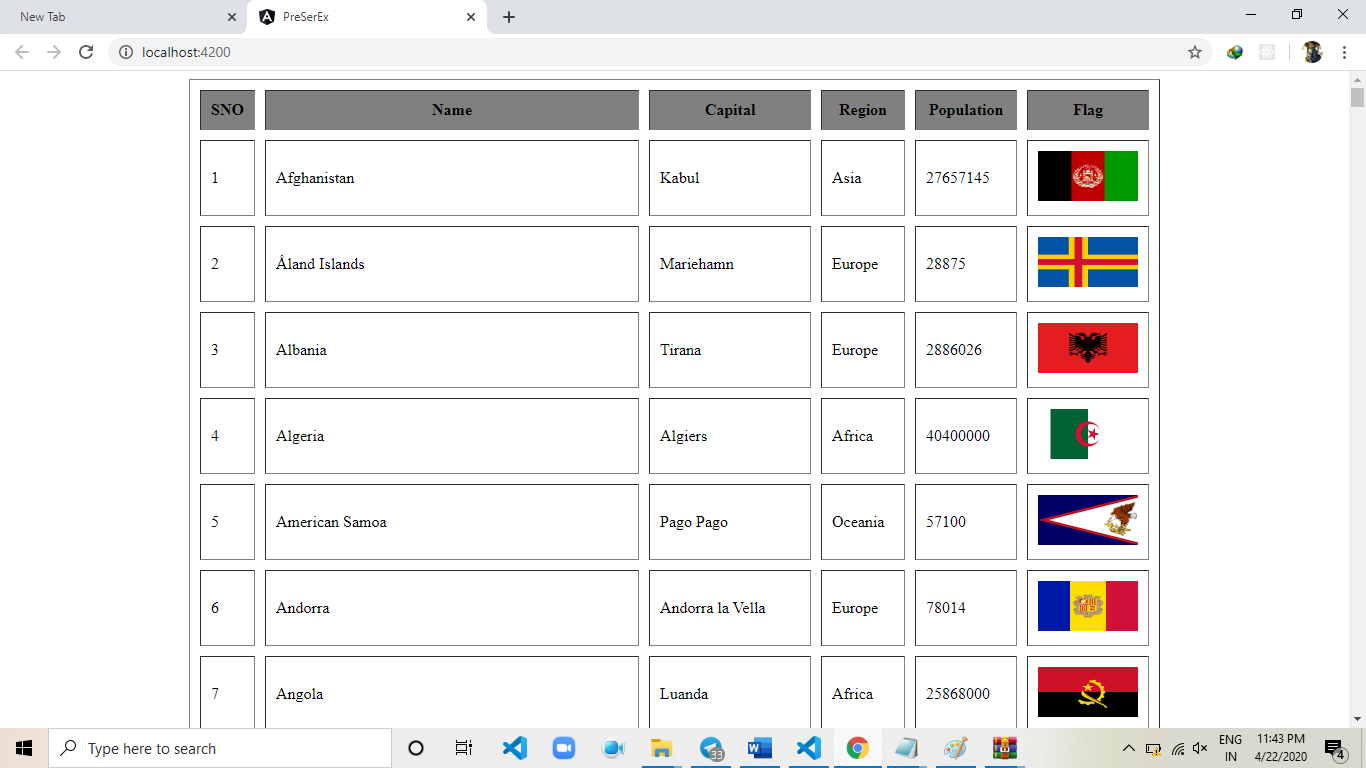
**index.html:**

<body>

<countries></countries>

</body>

**Result:**



**Chapter-3(Integration)**

**Series & Parallel Calls**

* Executing network calls "one by one" called as series calls.
* Executing network calls paralley called as Parallel Calls.
* to make parallel calls we need "Observable" class present in "rxjs-compat" package.
* we will download above library by using "yarn" tool.
* **Command**: yarn add rxjs-compat --save

**Java Integration**

* "EmployeeDetailRestResource" is the java webservices project.
* "EmployeeDetailRestResource" project will be deployed into "Tomcat" Server.
* This project gives the "XML" as Response.
* below url representing rest api url of java application.

**URL** : <http://localhost:9090/EmployeeDetailRestResource/api/empService/getAll>

**To execute java application we need following softwares**

-------------------------------------------------------

1) Tomcat

2) Ecilipse

3) jdk

4) "EmployeeDetailRestResource" project build

**Dot net Integration:**

* "MyFirstWebAPIService" is the dotnet web api application.
* we will deploy "MyFirstWebAPIService" application in "IIS" Server.
* below URL representing rest api url of dotnet web api application.
* URL : http://localhost:14741/api/Home
* above URL gives the xml as response.

**To execute dotnet application we need following softwares**

1) VisualStudio 2015

2) "MyFirstWebAPIService" Project Build

**Example**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

seriesAndParallelCallsEx

src

app

services

java.service.ts

dotnet.service.ts

components

series.component.ts

series.component.html

parallel.component.ts

parallel.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Commands:**

* yarn add rxjs-compat - - save
* ng g s services/java - -skipTests
* ng g s services/dotnet - -skipTests
* ng g c component/series - -skipTests -is - -selector=series - -flat true
* ng g c component/parallel - -skipTests -is - -selector=series - -flat true

**Java.service.ts:**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

providedIn: 'root'

})

export class JavaService {

constructor(private http:HttpClient) { }

public getEmployees():Observable<any>{

return this.http.get("http://localhost:9090/EmployeeDetailRestResource/api/empService/getAll");

}

}

**Dotnet.service.ts:**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

providedIn: 'root'

})

export class DotnetService {

constructor(private http:HttpClient) { }

public getEmployees():Observable<any>{

return this.http.get("http://localhost:14741/api/Home");

};

}

**Series.component.ts:**

import { Component, OnInit } from '@angular/core';

import { JavaService } from "../services/java.service";

import { DotnetService } from "../services/dotnet.service";

import { HttpErrorResponse } from "@angular/common/http";

@Component({

selector: 'series',

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

styles: []

})

export class SeriesComponent implements OnInit {

public javaResult:any;

public dotnetResult:any;

constructor(private java:JavaService,

private dotnet:DotnetService) { }

public errCallBack = (err:HttpErrorResponse)=>{

if(err.error instanceof Error){

console.log("client side error");

}else{

console.log("server side error");

}

};

ngOnInit() {

this.java.getEmployees().subscribe((posRes)=>{

this.javaResult = posRes;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

this.dotnet.getEmployees().subscribe((posRes)=>{

this.dotnetResult = posRes;

},this.errCallBack);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

},this.errCallBack);

}; }

**Series.component.html:**

<h4 style="color: red;">{{javaResult | json}}</h4>

<h4 style="color: royalblue;">{{dotnetResult | json}}</h4>

**app.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { SeriesComponent } from './components/series.component';

import { HttpClientModule } from '@angular/common/http';

import{ParallelComponent} from './components/parallel.component';

@NgModule({

declarations: [

AppComponent,

SeriesComponent,

ParallelComponent

],

imports: [

BrowserModule,HttpClientModule

],

providers: [],

bootstrap: [SeriesComponent]

})

export class AppModule { }

**Index.html:**

<body>

<series></series>

</body>

**Parallel.component.ts:**

import { Component, OnInit } from '@angular/core';

import { JavaService } from "../services/java.service";

import { DotnetService } from "../services/dotnet.service";

import { HttpErrorResponse } from "@angular/common/http";

import { Observable } from "rxjs-compat";

@Component({

selector: 'parallel',

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

styles: []

})

export class ParallelComponent implements OnInit {

public javaResult:any;

public dotnetResult:any;

constructor(private java:JavaService,

private dotnet:DotnetService) { }

public errCallBack = (err:HttpErrorResponse)=>{

if(err.error instanceof Error){

console.log("client side error");

}else{

console.log("server side error");

}

};

ngOnInit() {

Observable.forkJoin([

this.java.getEmployees(),

this.dotnet.getEmployees()

]).subscribe((posRes)=>{

this.javaResult = posRes[0];

this.dotnetResult = posRes[1];

},this.errCallBack);

}

};

**Parallel.component.html**

<h4>{{javaResult | json}}</h4>

<h4>{{dotnetResult | json}}</h4>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { SeriesComponent } from './components/series.component';

import { HttpClientModule } from '@angular/common/http';

import { ParallelComponent } from './components/parallel.component';

@NgModule({

declarations: [

AppComponent,

SeriesComponent,

ParallelComponent

],

imports: [

BrowserModule,HttpClientModule

],

providers: [],

bootstrap: [ParallelComponent]

})

export class AppModule { }

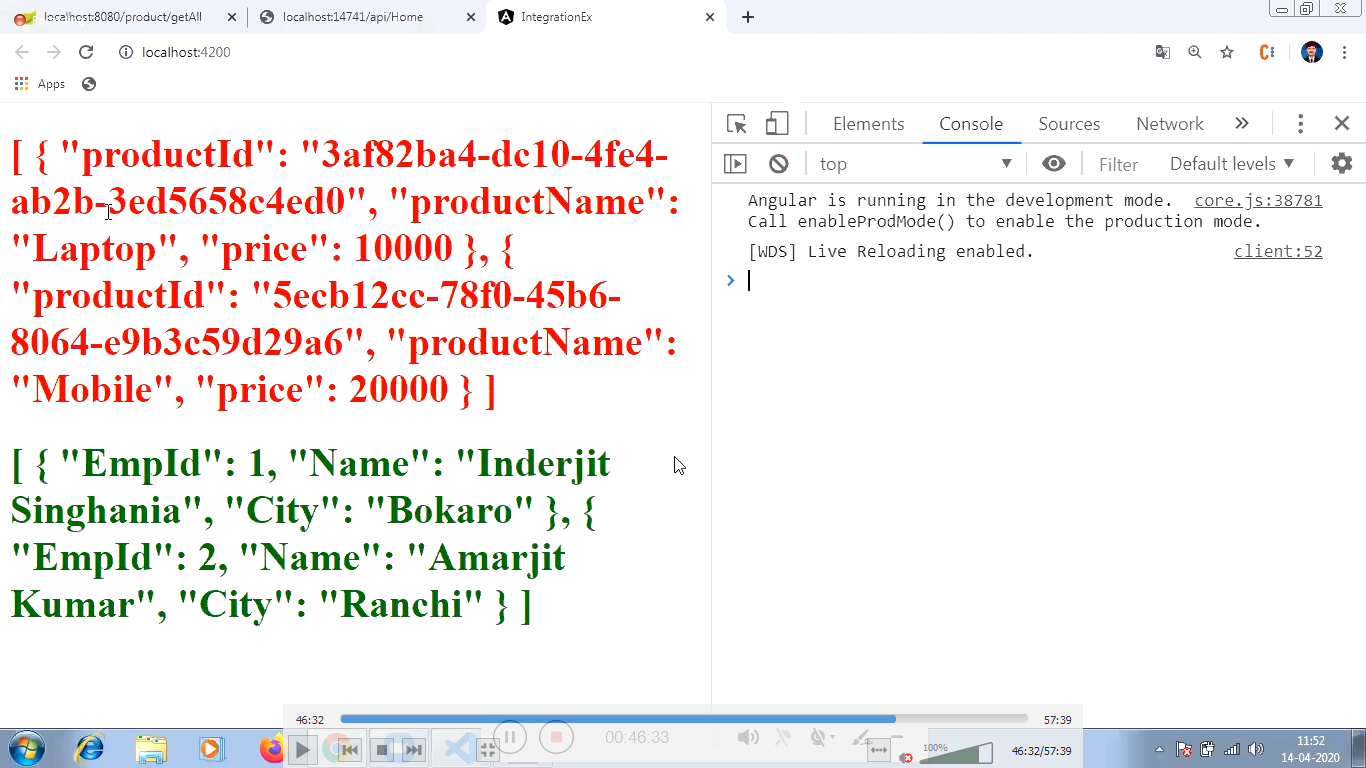
**Index.html**

<body>

<parallel></parallel>

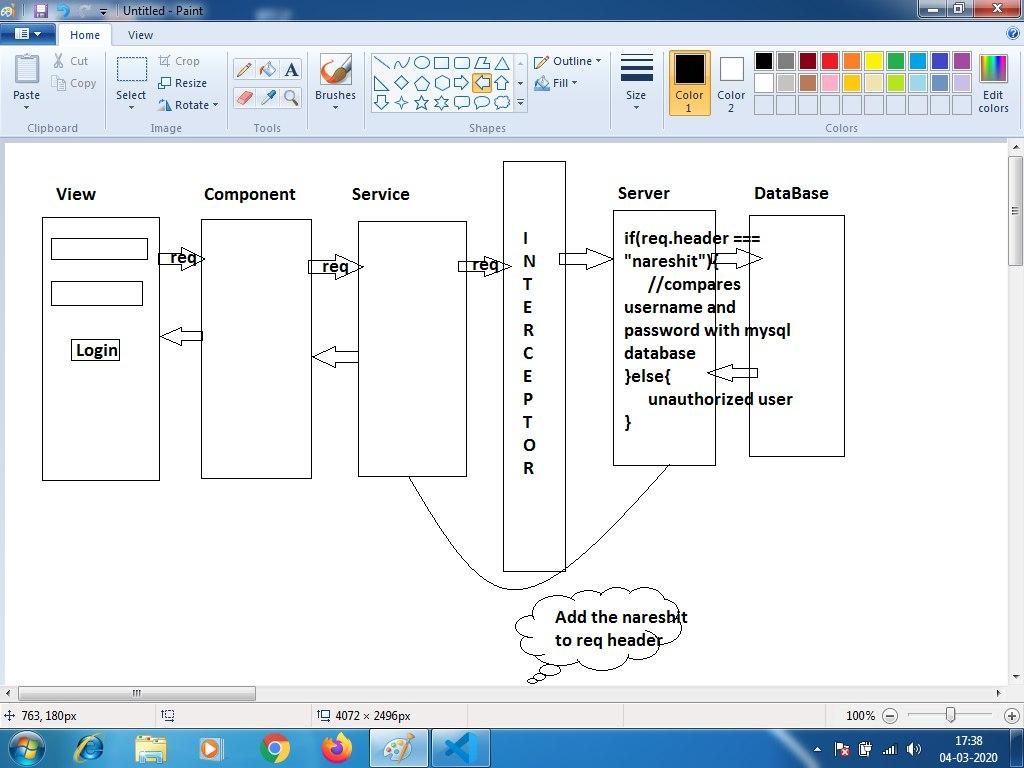
</body>

**Result:**



**Chapter-4(Interceptors)**

* Interceptors Authenticate the Http Requests.
* if Http Request is Authenticated,then req will bypass to server.
* In general, we will create Interceprors by using custom services.
* In general, we will register Interceprors in providers array in module file.



**steps to implement Interceptors Example**

---------------------------------------

**step 1.**

install SQLServer.

=> SQL Server 2014 Management Studio

**step 2.**

create the table in SQLServer.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

user : sa

password: 123

server : localhost

database: auth

table : login\_details

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**step 3.**

create the angular application

> ng new InterceptorsEx

**step 4.**

switch to angular application

> cd InterceptorsEx

**step 5.**

download the following node modules

=> express

=> mssql@6.0.1

=> body-parser

=> cors

* "express" module used to develop the rest apis
* "mssql@6.0.1" module used to interact with the SQLServer
* "body-parser" module used to read the client data.
* "cors" module used to enable the ports communication
* we will download above modules by using yarn tool.

**Command:** yarn add express mssql@6.0.1 body-parser cors --save

**step 6.**

develop the node server

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

interceprotsEx

server

server.js

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**step 7.**

start the node server

> cd server

> node server

**step 8.**

test the following rest api by using "Postman"

=> http://localhost:8080/login (POST)

**step 9.**

implement the Interceptor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

interceptorsEx

src

app

Interceptor

token.Interceptor.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "token.Interceptor.ts" used to add the nareshit as header to req.

after adding token we will send req to server.

**step 10.**

create the LoginService

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

interceprorsEx

src

app

services

login.service.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> ng g s services/login --skipTests

**step 11.**

create the component

> ng g c components/login --skipTests -is --selector=login --flat true

**step 12.**

register components and intercepror in app.module.ts file

**step 13.**

start the servers

Terminal-1

----------

> cd interceprorsEx/server

> node server

Terminal-2

----------

> cd interceprorsEx

> ng s -o

**Server.js:**

//import the modules

//require() function used to import the modules

let express = require("express");

let mssql = require("mssql");

let bodyparser = require("body-parser");

let cors = require("cors");

//create the rest object

let app = express();

//where "app" is the rest object

//"app" object used to develop the rest apis

//set the json as MIME Type

app.use(bodyparser.json());

//read the client data

app.use(bodyparser.urlencoded({extended:false}));

//enable the cors

app.use(cors());

//create the middleware function

//this middleware function used to check the headers

let checkHeaders = (req,res,next)=>{

let allHeaders = req.headers;

let str = allHeaders.token;

if(str === "nareshit"){

next();

}else{

res.send({"message":"unauthorized user"});

}

}

//create the rest api

app.post("/login",[checkHeaders],(req,res)=>{

mssql.connect({

user:"sa",

password:"123",

database:"auth",

server:"localhost"

},(err)=>{

if(err) throw err;

else{

let queryObj = new mssql.Request();

queryObj.query(`select \* from login\_details where uname='${req.body.uname}' and upwd='${req.body.upwd}'`,

(err,records)=>{

if(err) throw err;

else{

if(records.recordset.length>0){

res.send({"login":"success"});

}else{

res.send({"login":"fail"});

}

}

})

}

});

});

//assign the port no

app.listen(8080);

console.log("server listening the port no.8080");

**token.interceptor.ts:**

import { Injectable } from "@angular/core";

import { HttpRequest,

HttpHandler,

HttpEvent } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

providedIn:"root"

})

export class tokenInterceptor{

intercept(req:HttpRequest<any>,

handler:HttpHandler):Observable<HttpEvent<any>>{

const req1 = req.clone({

setHeaders:{

"token":"naresh"

}

});

return handler.handle(req1);

}

};

**Login.service.ts:**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

providedIn: 'root'

})

export class LoginService {

constructor(public http:HttpClient) { }

public authenticate(data:any):Observable<any>{

return this.http.post("http://localhost:8080/login",data);

};

};

**Login.component.ts:**

import { Component, OnInit } from '@angular/core';

import { LoginService } from "../services/login.service";

import { HttpErrorResponse } from "@angular/common/http";

@Component({

selector: 'login',

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

styles: []

})

export class LoginComponent implements OnInit {

public result:any;

constructor(public service:LoginService) { }

ngOnInit() {

}

public login(data:any):any{

this.service.authenticate(data)

.subscribe((posRes)=>{

this.result = posRes;

},(errRes:HttpErrorResponse)=>{

if(errRes.error instanceof Error){

console.log("client side error");

}else{

console.log("server side error");

}

});

};

};

**Login.component.html**

<fieldset>

<legend>Login</legend>

<input type="text"

name="uname"

placeholder="user name"

[(ngModel)]="uname">

<br><br>

<input type="password"

name="upwd"

placeholder="password"

[(ngModel)]="upwd">

<br><br>

<button (click)="login({'uname':uname,'upwd':upwd})">

Login

</button>

<h1>{{result | json}}</h1>

</fieldset>

**app.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { LoginComponent } from './components/login.component';

import { HttpClientModule, HTTP\_INTERCEPTORS } from '@angular/common/http';

import { FormsModule } from "@angular/forms";

import { tokenInterceptor } from './interceptor/token.Intercepror';

@NgModule({

declarations: [

AppComponent,

LoginComponent

],

imports: [

BrowserModule,HttpClientModule,FormsModule

],

providers: [{

provide:HTTP\_INTERCEPTORS,

useClass:tokenInterceptor,

multi:true

}],

bootstrap: [LoginComponent]

})

export class AppModule { }

**Index.html:**

<body>

<login></login>

</body>

**Chapter-5( Directives )**

* Directives enhances the view capabilities.
* We have two types of directives
  + Pre-defined directives
  + Custom directives
* The directives are given by angular framework is called predefined directives.
* The directives are doveloped by us based on application requirement called as custom directives

**=>Pre-defined directives**

1. ngFor
2. ngif
3. (click)
4. (dbclick)
5. [(ngmodel)]
6. (ngsubmit)
7. [ngclass]
8. [ngstyle]
9. [ngswitch]

* Directives are categorized into three types
  + Structural type directives
  + Event type directives
  + Attribute type directives
* Structural type directives have manipulate into dom
* Structural type directives starts with “\*”
* Based on the requirement we are adding or removing dom elements from browser memory.
* In order to handle events raised by dom ,we are using event type directives.
* Event type directives are serounder with “()”
* Attribute type directives serounder with ”[]”

**1) \*ngFor**

- this directive used to iterate the Array Elements.

**Syntax.**

\*ngFor= "let variable of array;constant1,constant2,...."

**constants**

**---------**

**1) index**

- it is used to get the indexes for each iteration.

2**) first**

- it is used to recognise the first element in array.

**3) last**

- it is used to recognise the last element in array.

**4) even**

- it will recognise even positions in array.

**5) odd**

- it will recognise odd positions in array.

**2) \*ngIf**

- this directive helps to write the conditions.

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

firstApp

src

app

first.component.ts

first.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**First.component.ts:**

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

@Component({

selector: 'app-root',

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

styleUrls: ['./first.component.css']

})

export class firstComponent {

title = 'first';

num:number = 0;

clickMe(arg1,arg2){

if(arg1 === "admin" && arg2 === "admin"){

alert("Login Success");

}else{

alert("Login Fail");

}

};

}

**First.component.html**

<!--

\*ngFor

------

- it is used to iterate the array elements.

-->

<!--

<div \*ngFor="let x of [10,20,30,40,50];

let i = index;

let f = first;

let l = last;

let e = even;

let o = odd;">

<span>{{x}}...{{i}}...{{f}}...{{l}}...{{e}}...{{o}}</span>

</div>

-->

<!--

[ngStyle]

- ngStyle directive used to apply the "CSS" to "DOM Elements".

<h1 [ngStyle]="{'color':'red'}">Hello</h1>

<h1 [ngStyle]="{'color':title==='firstApp'?'green':'red'}">Welcome</h1>

<div \*ngFor="let x of [10,20,30,40,50]">

<div [ngSwitch]="x">

<div \*ngSwitchCase="10" [ngStyle]="{'color':'red'}">{{x}}</div>

<div \*ngSwitchCase="20" [ngStyle]="{'color':'green'}">{{x}}</div>

<div \*ngSwitchCase="30" [ngStyle]="{'color':'blue'}">{{x}}</div>

<div \*ngSwitchCase="40" [ngStyle]="{'color':'pink'}">{{x}}</div>

<div \*ngSwitchDefault [ngStyle]="{'color':'yellow'}">{{x}}</div>

</div>

</div>

-->

<!--

[ngClass]

- it is used to apply the bootstarp to DOM Elements

-->

<!--

<h1 [ngClass]="{'text-success':true}">Hello</h1>

<h1 [ngClass]="{'text-danger':title==='firstApp'}">Welcome</h1>

<div \*ngFor="let x of [10,20,30,40,50]">

<div [ngSwitch]="x">

<div \*ngSwitchCase="10" [ngClass]="{'text-success':true}">{{x}}</div>

<div \*ngSwitchCase="20" [ngClass]="{'text-info':true}">{{x}}</div>

<div \*ngSwitchCase="30" [ngClass]="{'text-primary':true}">{{x}}</div>

<div \*ngSwitchCase="40" [ngClass]="{'text-danger':true}">{{x}}</div>

<div \*ngSwitchDefault [ngClass]="{'text-default':true}">{{x}}</div>

</div>

</div> -->

<!--

<div class="container">

<br><br>

<button class="glyphicon glyphicon-plus btn-success btn-sm"

(dblclick)="num=num+1"></button>

<button class="btn btn-primary">{{num}}</button>

<button class="glyphicon glyphicon-minus btn-success btn-sm"

(dblclick)="num=num-1"></button>

</div>

-->

<fieldset>

<legend>Login</legend>

<input type="text"

placeholder="user name"

#uname>

<br><br>

<input type="password"

placeholder="user password"

#upwd>

<br><br>

<button (click)="clickMe(uname.value,upwd.value)">Login</button>

</fieldset>

**App.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FirstComponent } from './first.component';

@NgModule({

declarations: [

FirstComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [FirstComponent]

})

export class AppModule { }

**Index.html:**

<body>

<first></first>

</body>

**=>Custom Directives:**

Creating our own directives based on application requirement is called as custom directives.

We can create two types of custom directives.

* Attribute type custom directives
* Structurl type custom directives

**Attribute type custom directives**

Directive is the predefined class used to dovelop the custom directives

**“elementref”** is the predefined class used to manipulate the dom elements in custom directives

**“input”** is the predefined class used to apply the data to directive form component

**“Hostlistener”** class helps to apply the mouse events to dom elements

**Command**: ng g d mydir --skipTests

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CustDirex

src

app

my.directive.ts

app.component.ts

app.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.html**

<h1 [var\_one]="color\_one.value" [var\_two]="color\_two.value" myDir>hello</h1>

<br><br>

<input type="color" #color\_one>

<input type="color" #color\_two>

**my.directive.ts**

import { Directive,HostListener,Input,ElementRef } from '@angular/core';

@Directive({

selector: '[myDir]'

})

export class myDirective {

@Input() var\_one;

@Input() var\_two;

constructor(public \_el:ElementRef) { }

@HostListener("mouseenter") onmouseenter(){

this.changeColor(this.var\_one);

};

@HostListener("mouseleave") onmouseleave(){

this.changeColor(this.var\_two);

};

public changeColor(arg1){

this.\_el.nativeElement.style.backgroundColor=arg1;

}

}

**app.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { myDirective } from './my.directive';

@NgModule({

declarations: [

AppComponent,

myDirective

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Index.html:**

<body>

<app></app>

</body>

**Structural Type Custom Directives**

* Structural Directive prefixed with "\*".
* Structural Directive have the capability to "manipulate the DOM".
* Based on Requirement DOM Element "added/removed" from browser memory.
* "Directive" is the predefined class, used to create the "Custom Directive".
* "TemplateRef" is the predefined class, used to manipulate the "DOM".
* "ViewContainerRef" is the predefined class, used to "add/remove" the DOM Elements from browser memory.
* "Input" is the predefined class used to pass the data from Component to Directive.

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CustDirex

src

app

strl.directive.ts

app.component.ts

app.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.html**

<h1 \*hello="false">Welcome</h1>

**strl.directive.ts**

import { Directive,TemplateRef,ViewContainerRef,Input } from '@angular/core';

@Directive({

selector: '[hello]'

})

export class StrlDirective {

constructor(public \_templateRef:TemplateRef<any>,

public \_viewContainerRef:ViewContainerRef) { }

@Input() set hello(arg1:boolean){

//if arg1 is "true" , add "\_templateRef" to "browser memory" with the help of "\_viewContainerRef"

if(arg1){ this.\_viewContainerRef.createEmbeddedView(this.\_templateRef)

}else{

this.\_viewContainerRef.clear();

}; } };

**App.module.ts:**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { StrlDirective } from './strl.directive';

@NgModule({

declarations: [

AppComponent,

StrlDirective

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Index.html:**

<body>

<app-root></app-root>

</body>

**Chapter-6(Communication between components)**

* As a angular developer we can create morethan one component
* We can provide communication between components
* In angular we can provide communication in four ways
* @Input
* @Output
* @viewchild
* @viewchildren

**@Input**

This directive used to store the data from parent component to child component.

**@Output**

This directive used to store the data from child component to parent component.

**Steps to store the data from parent component to child component**

----------------------------------------------------------------

**Step-**1) create the childComponent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

child.component.ts

child.component.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step-**2) create the parentComponent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

parent.component.ts

parent.component.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step-**3)map the parentcomponent data to childcomponent properties.

**Step-**4) bootstrap the parentComponent

**Steps to store the data from parent component to child component**

**Step-1.**Create the child component

**Step-2.**Fire the eventEmiter object

**Step-3.**Map the childcomponent data to parentcomponent

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Combtcom

src

app

child.component.ts

child.component.html

parent.component.ts

parent.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**child.component.ts**

import { Component,Input, Output, EventEmitter } from "@angular/core";

@Component({

selector:"child",

templateUrl:"./child.component.html"

})

export class childComponent{

@Input() p\_id;

@Input() p\_name;

@Input() p\_cost;

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

clickMe():any{

this.send.emit(this.p\_id+"...."+this.p\_name+"...."+this.p\_cost)

};

};

**child.component.html**

<h2>Product ID:<span style="color: red;">{{p\_id}}</span></h2>

<h2>Product Name:<span style="color: red;">{{p\_name}}</span></h2>

<h2>Product Cost:<span style="color: red;">{{p\_cost}}</span></h2>

<button (click)="clickMe()">Send</button>

<hr>

**parent.component.ts**

import { Component } from "@angular/core";

@Component({

selector:"parent",

templateUrl:"./parent.component.html"

})

export class parentComponent{

private products:Array<any> = [

{p\_id:111,p\_name:"p\_one",p\_cost:10000},

{p\_id:222,p\_name:"p\_two",p\_cost:20000},

{p\_id:333,p\_name:"p\_three",p\_cost:30000},

{p\_id:444,p\_name:"p\_four",p\_cost:40000},

{p\_id:555,p\_name:"p\_five",p\_cost:50000}

];

public myFun(data:any){

alert(data);

}

};

**parent.component.html**

<child

[p\_id]="x.p\_id"

[p\_name]="x.p\_name"

[p\_cost]="x.p\_cost"

(send)="myFun($event)"

\*ngFor="let x of products"></child>

app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { parentComponent } from './parent.component';

import { childComponent } from './child.component';

@NgModule({

declarations: [

AppComponent,parentComponent,childComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [parentComponent]

})

export class AppModule { }

**index.html**

<body>

<parent></parent>

</body>

**@Viewchild() and @Viewchildren:**

If we want to store the data between components without relationship between them parent and child then we will use @viewchild() and @viewchildren()

**Steps to Implement the Application by using ViewChild and ViewChildren**

----------------------------------------------------------------

1) create the secondComponent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

second.component.ts

second.component.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2) create the firstComponent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

first.component.ts

first.component.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3) bootstrap the firstComponent

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Combtcom

src

app

first.component.ts

first.component.html

second.component.ts

second.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**first.component.ts**

import { Component,ViewChild, ViewChildren, QueryList } from "@angular/core";

import { secondComponent } from "./second.component";

@Component({

selector:"first",

templateUrl:"./first.component.html"

})

export class firstComponent{

/\*

@ViewChild(secondComponent,{static:true})

private second:secondComponent;

clickMe(){

this.second.var\_one = "welcome\_1";

this.second.var\_two = "welcome\_2";

};

\*/

@ViewChildren(secondComponent)

private obj:QueryList<secondComponent> = new QueryList();

private arr:Array<any> = [];

ngAfterViewInit(){

this.arr = this.obj.toArray();

};

clickMe(){

this.arr.forEach((element,index)=>{

element.var\_one = "welcome\_1";

element.var\_two = "welcome\_2";

});

};

//QueryList is the utility class helps to create the map object based on target occurances.

//we must convert datastructure to equalent array.

//in order to convert "one data structure" to "another data structure" we will use ngAfterViewInit() life cycle hook

};

**first.component.html**

<second></second>

<second></second>

<second></second>

<button (click)="clickMe()">Change</button>

**second.component.ts**

import { Component } from "@angular/core";

@Component({

selector:"second",

templateUrl:"./second.component.html"

})

export class secondComponent{

public var\_one:string;

public var\_two:string;

constructor(){

this.var\_one = "hello\_1";

this.var\_two = "hello\_2";

};

};

**second.component.html**

<h1 style="color: red;">{{var\_one}}</h1>

<h1 style="color: green;"><marquee>{{var\_two}}</marquee></h1>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { firstComponent } from './first.component';

import { secondComponent } from './second.component';

@NgModule({

declarations: [

AppComponent,firstComponent,secondComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [firstComponent]

})

export class AppModule { }

**index.html**

<body>

<first></first>

</body>

**Note:** Viewchild() can reflect the changes on Target component if any one existing occurs to “To overcome limitation we will use @viewchildren.

**Chapter-7( Pipes )**

* Pipes are used to manipulate the data based on Application Requirement.
* we have two types of pipes.
  + predefined pipes
  + custom pipes
* the pipes given by angular framework called as predefined pipes.
* the pipes developed by us based on Application Requirement called as custom Pipe.

**=>predefined pipes**

* uppercase
* lowercase
* titlecase
* currency
* json
* slice
* number
* percent
* async
* date

**1) uppercase**

- it is used to convert the lowercase characters to uppercase characters.

**2) lowercase**

- it is used to convert the uppercase characters to lowercase characters.

**3) titlecase**

- it is used to create the camelcase words.

**4) currency**

- it is used to append the currencies symbols to numerical values.

**5) json**

- it will convert "JSON Object" to "JSON String".

**6) slice**

- it is used to manipulate the arrays.

**7) number/decimal**

- it is used to manipulate the numerical values.

**8) percent**

- used to convert the fractions to equalent percentages.

**9) async**

- it is used to display the asynchronous data on webpages.

**10) date**

- it is used to manipulate the "date" accroding to application requirement.

**Command:** ng g p reverse -–skipTests

ng g p message --skipTests

**=>Custom Pipes**

- creating our own pipes based on application requirement called as custom pipe.

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pipesex

src

app

reverse.pipe

message.pipe

app.component.ts

app.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.html**

<h1>{{“hello” | reverse}}</h1>

<h1>{{“hello” | reverse}}</h1>

//where reverse,message are custom pipes

**reverse.pipe**

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

name: 'reverse'

})

export class ReversePipe implements PipeTransform {

transform(value: any, ...args: any[]): any {

return Array.from(value).reverse().join("");

}

}

**message.pipe**

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

name: 'message'

})

export class MessagePipe implements PipeTransform {

transform(value: any, ...args: any[]): any {

return args[1]+" "+args[0]+" "+value;

}

}

**app.component.html**

<h1 style="color: green;">

{{"Angular9" | message:"to":"welcome"}}

</h1>

<h1 style="color: red;">{{"hello" | reverse}}</h1>

<!--

where "reverse" is the custom pipe

-->

<h1>{{var\_ten | async}}</h1>

<h1>{{var\_nine | date:"fullDate"}}</h1>

<h1>{{var\_nine | date:"medium"}}</h1>

<h1>{{var\_nine | date:"short"}}</h1>

<h1>{{var\_nine | date:"dd-MMM-yyyy"}}</h1>

<h1>{{var\_nine | date:"dd-MM-yy"}}</h1>

<h1>{{var\_eigth | percent}}</h1>

<h1>{{var\_seven | number:"4.1-2"}}</h1>

<h1>{{var\_seven | number:"3.2-3"}}</h1>

<h1>{{var\_six | slice:2:-3}}</h1>

<h1>{{var\_six | slice:2:-1}}</h1>

<h1>{{var\_six | slice:2:5}}</h1>

<h1>{{var\_six | slice:2:4}}</h1>

<h1>{{var\_five | json}}</h1>

<h1>{{var\_four | currency:"INR"}}</h1>

<h1>{{var\_four | currency:"EUR"}}</h1>

<h1>{{var\_four | currency:"GBP"}}</h1>

<h1>{{var\_four | currency}}</h1>

<h1>{{var\_three | titlecase}}</h1>

<h1>{{var\_two | lowercase}}</h1>

<h1>{{var\_one | uppercase}}</h1>

**app.component.ts**

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

private var\_one:string="hello";

private var\_two:string = "HELLO";

private var\_three:string="naresh it";

private var\_four:number=100;

private var\_five:any={

p\_id:111,

p\_name:"p\_one",

p\_cost:10000

};

private var\_six:Array<number>=[

10,20,30,40,50

];

private var\_seven:number=100.12345;

private var\_eigth:number = 0.9;

private var\_nine:Date = new Date();

private var\_ten:any;

constructor(){

this.var\_ten = new Promise((resolve,reject)=>{

setTimeout(()=>{

resolve("Success");

},5000);

});

};

}

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { ReversePipe } from './reverse.pipe';

import { MessagePipe } from './message.pipe';

@NgModule({

declarations: [

AppComponent,

ReversePipe,

MessagePipe

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<body>

<app-root></app-root>

</body>

**Chapter-8 ( Lifecycle hooks )**

1) ngOnChanges()

2) ngOnInit()

3) ngDoCheck()

4) ngAfterContentInit()

5) ngAfterContentChecked()

6) ngAfterViewInit()

7) ngAfterViewChecked()

8) ngOnDestroy()

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

lifecyclehoks

src

app

app.component.ts

app.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.ts**

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

constructor(){

//constructor will execute at booting time

//constructor used to initilize the instant members

//mainly we are using constructor for dependency injection purpose

console.log("--in constructor--");

};

ngOnChanges(){

//when ever change detected in "@Input" binding properties automatically this life cycle hook will execute.

//ngOnChanges() will execute immidiately after constructor

console.log("--in ngOnChanges--");

};

ngOnInit(){

//ngOnInit() will execute after first successful execution of ngOnChanges()

//ngOnInit() also called as first life cycle hook of component.

//ngOnInit() will execute only once.

//ngOnInit() helps to maintain the main business logic.

//Ex. making the service calls

console.log("--in ngOnInit--");

};

public num:number = 100;

public increment():number{

return this.num+=100;

};

public decrement():number{

return this.num-=100;

};

ngDoCheck(){

//when ever change detected in Application Model(num), automatically this life cycle hook will execute.

console.log("--in ngDoCheck--");

};

ngAfterContentInit(){

//if framework identifies the memory for component with the help of browser engine, automatically this life cycle hook will execute

console.log("--in ngAfterContentInit--");

};

ngAfterConetentChecked(){

//if browser engine allots the memory for component then this life cycle hook will execute.

console.log("--in ngAfterContentChecked--");

};

ngAfterViewInit(){

//if component loaded successfully, then this life cycle hook will execute.

console.log("--in ngAfterViewInit--");

};

ngAfterViewChecked(){

//if data populated successfully, then this life cycle hook will execute

console.log("--in ngAfterViewCheck--");

};

ngOnDestroy(){

//ngOnDestroy() will execute by framework, before kiling the component by framework.

//in general we will use this life cycle hook to maintain cleanup code

console.log("--in ngOnDestroy--");

};

};

**app.component.html**

<h1>{{num}}</h1>

<button (click)="increment()">Increment</button>

<button (click)="decrement()">Decrement</button>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<body>

<app-root></app-root> </body>

**Chapter-9 ( Forms )**

* Angular supports two types of forms.

1) Template Driven Forms (TDF)

2) Model Driven Forms (MDF)

* "Template Driven Forms" mainly on Application Design.
* "Template Driven Forms" may not support Framework facilities.
* "Model Driven Forms" mainly on "Application Model".
* "Model Driven Forms" also called as Reactive Forms.
* "Model Driven Forms" provides the facilities upto Framework Level Forms Design.

**=>Template Driven Forms**

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tdfex

src

app

components

tdf.component.ts

tdf.component.html

app.module.ts

index.html

**tdf.component.ts**

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'tdf',

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

styleUrls: ['./tdf.component.css']

})

export class TdfComponent implements OnInit {

constructor() { }

ngOnInit() {

}

register(data:any){

console.log(data);

}}

**tdf.component.html**

<!--

Directives in TDF

1) ngForm

- this directive helps assign the logical name to forms.

2) ngModel

- this directive behaves like one way data binding directive.

- this directive saves the application data (Form Field Data).

3) ngModelGroup

- this directive helps to create the subgroups.

[A group inside another group called as subgroup]

4) to handle form submition we will use (ngSubmit) directive.

-->

<body>

<form #profileData="ngForm"

(ngSubmit)="register(profileData.value)">

<table>

<tr>

<td>User Name</td>

<input type="text" name="uname" ngModel>

</tr>

<tr>

<td>Password</td>

<input type="password" name="upwd" ngModel>

</tr>

<tr>

<td>Age</td>

<input type="number" name="age" ngModel>

</tr>

<tr>

<td>Gender</td>

<td><input type="radio"

name="gender"

value="male"

ngModel>Male</td>

<td><input type="radio"

name="gender"

value="female"

ngModel>Female</td>

</tr>

<tr ngModelGroup="addr">

<td>City</td>

<td>

<input type="text"

name="ucity"

ngModel>

</td>

</tr>

<tr>

<td>Country</td>

<td><select name="ucountry" ngModel>

<option value="india">India</option>

<option value="usa">USA</option>

<option value="canada">Canada</option>

</select></td>

</tr>

<tr>

<td></td>

<td><input type="submit"></td>

</tr>

</table>

</form>

</body>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from "@angular/forms";

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

import { TdfComponent } from './components/tdf.component';

@NgModule({

declarations: [

AppComponent,

TdfComponent

],

imports: [

BrowserModule,FormsModule

],

providers: [],

bootstrap: [TdfComponent]

})

export class AppModule { }

**index.html**

<body>

<tdf></tdf>

</body>

**=>Model Driven Forms**

* Model Driven forms provides the more flexibility to developers to handle "validations".
* Model Driven Forms also called as "Reactive Forms".
* Reactive Forms present in "ReactiveFormsModule".
* [formGroup] is the directive used to assign the logical name to Forms.
* "formControlName" is the directive used to save the forms data(form fields data).
* "formGroupName" is the directive used to create the SubGroups.

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mdfex

src

app

app.component.ts

app.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.ts**

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

import { FormGroup, FormControl, Validators } from "@angular/forms";

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

profileData:FormGroup;

constructor(){

this.profileData = new FormGroup({

uname : new FormControl("Naresh",[Validators.required,

Validators.minLength(3),

Validators.maxLength(6)]),

addr : new FormGroup({

address : new FormControl()

}),

gender : new FormControl(),

country : new FormControl()

});

}

register():any{

console.log(this.profileData.value);

};

};

**app.component.html**

<div class="container mt-5">

<form [formGroup]="profileData"

(ngSubmit)="register()">

<div class="form-group">

<label>Uname</label>

<input type="text"

name="uname"

class="form-control"

formControlName="uname"

required>

</div>

<div \*ngIf="profileData.controls['uname']

.hasError('required')"

class="alert alert-danger">

\*\*\*\* can't left blank \*\*\*\*

</div>

<div \*ngIf="profileData.controls['uname']

.hasError('minlength')"

class="alert alert-danger">

\*\*\*\* minimum 3 characters are required \*\*\*\*

</div>

<div \*ngIf="profileData.controls['uname'].hasError('maxlength')"

class="alert alert-danger">

\*\*\*\* maximum 6 characters are allowed \*\*\*\*

</div>

<div class="form-group" formGroupName="addr">

<div class="form-group">

<label>Address</label>

<textarea

cols="4"

rows="5"

name="address"

formControlName="address"

class="form-control"></textarea>

</div>

</div>

<div class="form-group">

<label>Gender</label>

<input type="radio"

class="form-control"

name="gender"

value="male"

formControlName="gender">Male

<input type="radio"

class="form-control"

name="gender"

value="female"

formControlName="gender">Female

</div>

<div class="form-group">

<label>Country</label>

<select name="country"

class="form-control"

formControlName="country">

<option value="india">India</option>

<option value="usa">USA</option>

<option value="canada">Canada</option>

<option value="japan">Japan</option>

<option value="china">China</option>

</select>

</div>

<div class="form-group" align="center">

<input type="submit"

class="btn btn-success">

</div>

</form>

</div>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { ReactiveFormsModule } from '@angular/forms';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,ReactiveFormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<body>

<app-root></app-root>

</body>

**Chapter-10 ( Angular Material )**

* Angular Material is the library provided by google.
* Angular Material library used to develop the Rich UI.
* we will add Angular Material by using following command.

> ng add @angular/material

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

angmatex

src

app

app.component.ts

app.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.html**

<br><br>

<div class="mat-elevation-z8">

<table mat-table [dataSource]="data" style="width: 100%;" matSort>

<ng-container matColumnDef="p\_id">

<th mat-header-cell \*matHeaderCellDef mat-sort-header>p\_id</th>

<td mat-cell \*matCellDef="let row">{{row.p\_id}}</td>

</ng-container>

<ng-container matColumnDef="p\_name">

<th mat-header-cell \*matHeaderCellDef mat-sort-header>p\_name</th>

<td mat-cell \*matCellDef="let row">{{row.p\_name}}</td>

</ng-container>

<ng-container matColumnDef="p\_cost">

<th mat-header-cell \*matHeaderCellDef mat-sort-header>p\_cost</th>

<td mat-cell \*matCellDef="let row">{{row.p\_cost}}</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row;columns:displayedColumns"></tr>

</table>

<mat-paginator [pageSizeOptions]="[1, 2, 3, 5]"></mat-paginator>

</div>

**app.component.ts**

import { Component, ViewChild } from '@angular/core';

//prepare data, which is suitable to "Material Table"

import { MatTableDataSource,MatPaginator,MatSort } from "@angular/material";

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

@ViewChild(MatPaginator,{static:true})

public paginator:MatPaginator;

@ViewChild(MatSort,{static:true})

public sort:MatSort;

public displayedColumns:string[] = ["p\_id","p\_name","p\_cost"];

public data:MatTableDataSource<any>;

constructor(){

this.data = new MatTableDataSource([

{"p\_id":111,"p\_name":"p\_one","p\_cost":10000},

{"p\_id":555,"p\_name":"p\_five","p\_cost":50000},

{"p\_id":222,"p\_name":"p\_two","p\_cost":20000},

{"p\_id":444,"p\_name":"p\_four","p\_cost":40000},

{"p\_id":333,"p\_name":"p\_three","p\_cost":30000}

]); };

ngOnInit(){

this.data.paginator = this.paginator;

this.data.sort = this.sort; };}

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { MatTableModule,MatPaginatorModule, MatSortModule } from "@angular/material";

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

BrowserAnimationsModule,

MatTableModule,

MatPaginatorModule,

MatSortModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<body>

<app-root></app-root> </body>

**Chapter-11( Unit-Test Cases )**

* Testing particular functionality with assumptions called as Unit Testing.
* "karma" is the automation tool, helps to write the unit test cases.
* "karma" is the inbuilt tool of angular.
* unit Testing files should have the ".spec.ts" extension.
* we will execute unit test cases by using following command.

> ng test

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unittestex

src

app

app.component.ts

app.component.html

calc.spec

calc

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.ts**

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'unitTestCasesEx';

}

**calc.spec**

import { Calculator } from "./calc";

/\*

karma with jasmine starts the execution from describe()

\*/

describe("calculator testing",()=>{

let obj:Calculator;

/\*

//it will execute before each describe() function

beforeEach(()=>{

obj = new Calculator();

});

\*/

/\*

//it will execute only once globally

\*/

beforeAll(()=>{

obj = new Calculator();

});

/\*

these describe() functions used to write the unit test cases to particular functions

\*/

describe("add function testing",()=>{

/\*

it() function used to write the test suits

\*/

it("10+10 should be equal to 20",()=>{

const result = obj.add(10,10);

/\*

expect() function used for assertions

\*/

expect(result).toBe(20);

});

});

describe("sub function testing",()=>{

it("10-10 should be equal to 0",()=>{

const result = obj.sub(10,10);

expect(result).toBe(0);

});

});

describe("array testing",()=>{

it("check 30 in my\_array",()=>{

expect(obj.my\_array).toContain(30);

});

});

});

**Calc:**

export class Calculator{

public add(num1:number,

num2:number):number{

return num1+num2;

};

public sub(num1:number,

num2:number):number{

return num1-num2;

};

public my\_array:Array<number> = [10,20,30,40,50]; };

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

BrowserAnimationsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<body>

<app-root></app-root>

</body>

**Chapter-12( BehaviorSubject )**

* BehaviorSubject used to sync the data between components.
* BehaviorSubject is the predefined service available in "rxjs" package

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bahaviourSubEx

src

app

services

test.service.ts

components

first.component.ts

first.components.html

second.component.ts

second.component.html

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**test.service.ts**

import { Injectable } from "@angular/core";

import { BehaviorSubject } from "rxjs";

@Injectable({

providedIn:"root"

})

export class testService{

private data = new BehaviorSubject<string>("Angular9");

public cast = this.data.asObservable();

public changeData(arg1:string){

this.data.next(arg1);

};

};

**first.component.ts**

import { Component } from "@angular/core";

import { testService } from "../services/test.service";

@Component({

selector:"first",

templateUrl:"./first.component.html"

})

export class firstComponent{

private result:string;

constructor(private service:testService){}

ngOnInit(){

this.service.cast.subscribe((posRes)=>{

this.result = posRes;

});

};

clickMe(arg1){

this.service.changeData(arg1);

};

};

**first.components.html**

<h1>{{result}}</h1>

<input type="text" #msg>

<button (click)="clickMe(msg.value)">Change</button>

**second.component.ts**

import { Component } from "@angular/core";

import { testService } from "../services/test.service";

@Component({

selector:"second",

templateUrl:"./second.component.html"

})

export class secondComponent{

private result:string;

constructor(private service:testService){}

ngOnInit(){

this.service.cast.subscribe((posRes)=>{

this.result = posRes;

});

};

};

**second.component.html**

<h1>{{result}}</h1>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { firstComponent } from './components/first.component';

import { secondComponent } from './components/second.component';

@NgModule({

declarations: [

AppComponent,firstComponent,secondComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<body>

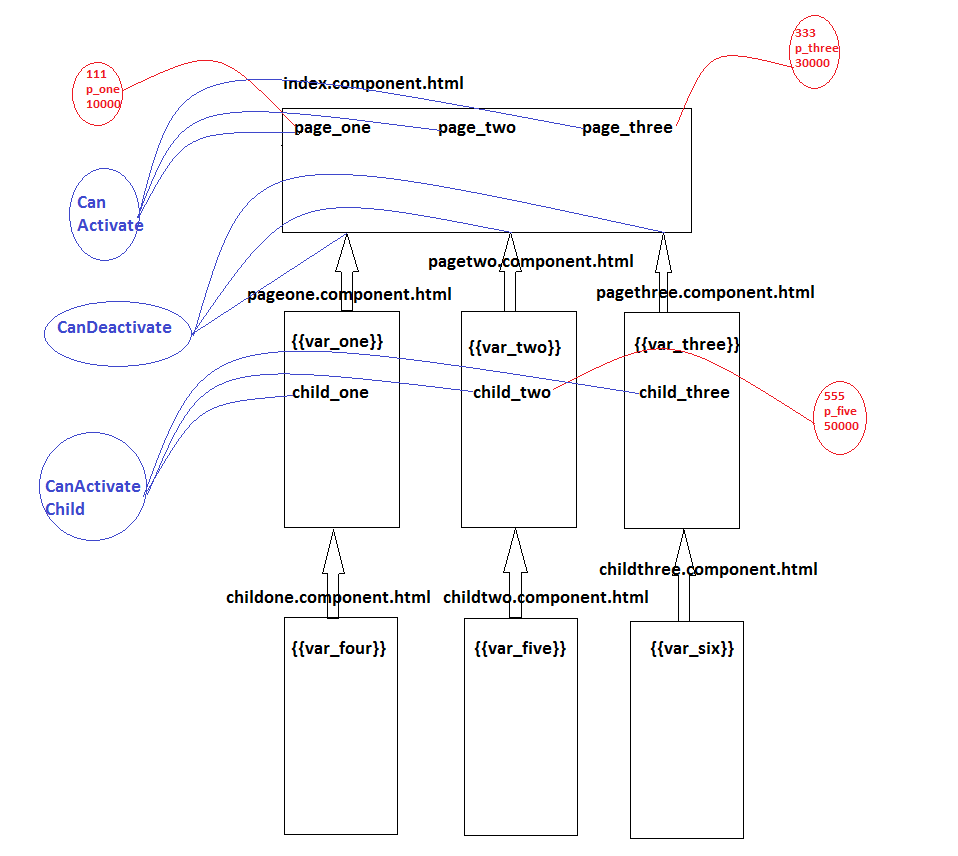
<app-root></app-root>

</body>

**Chapter-13(Single Page Applications)**

* loading one template to another template without refreshing the whole webpage called as single page application.
* loading one webpage to another webpage in single page application called as routing.
* we will implement the "Routing" in single page application by using "Routes" class.
* we will load "Routes" into framwork by using "RouterModule"
* both "Routes" and "RouterModule" present in "@angular/router" package

**Diagram:**



**step 1.**

create the components

* ng g c components/index --skipTests -is --selector=index

--flat true

* ng g c components/pageone --skipTests -is --selector=pageone

--flat true

* ng g c components/pagetwo --skipTests -is --selector=pagetwo

--flat true

* >ng g c components/pagethree --skipTests -is -- selector=pagethree --flat true
* where "IndexComponent" is the main component.
* where "PageoneComponent", "PagetwoComponent" and "PagethreeComponent" are target components in single page application

**step 2.**

implement the business logic in target components

**step 3.**

create the router links

**step 4.**

implement the routing

src

app

routes

app.routes.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**step 5.**

load "appRoutes" into framwork by using "RouterModule"

**implementation of child routing**

-------------------------------

**step 6.**

create the components

* ng g c components/childone --skipTests -is --selector=childone --flat true
* ng g c components/childtwo --skipTests -is --selector=childtwo --flat true
* ng g c components/childthree --skipTests -is --selector=childthree --flat true

**step 7.**

implement the business logic in target components

**step 8.**

create the hyperlinks

- we must create following hyperlinks

=> /child\_one

=> /child\_two

=> /child\_three

**step 9.**

implement the child routing

**step 10.**

**Passing Routing Parameters in Single Page Applications**

------------------------------------------------------

"ActivatedRoute" is the predefined class in Angular, helps to read the Routing Parameters.

"snapshot" is the predefined property(utility property) helps to ActivatedRoute in order to read Routing Parameters.

**step 11.**

**Routing Guards**

--------------

- Routing Guards helps to perform the authentication in single page applications.

**1) CanActivate**

- authentication before entering into main routes.

**2) CanDeactivate**

- authentication before leaving main routes.

**3) CanActivateChild**

- authentication before entering into child routes.

we will implement authentication Guards by using custom services

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

spademoex

src

app

components

page\_one.component.html

page\_one.component.ts

page\_two.component.html

page\_two.component.ts

page\_three.component.html

page\_three.component.ts

child\_one.component.html

child\_one.component.ts

child\_two.component.html

child\_two.component.ts

child\_three.component.html

child\_three.component.ts

index.component.html

index.component.ts

guards

auth.guards.ts

routings

app.routes.ts

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**page\_one.component.html**

<p>page-one works!</p>

<h1 style="color: black;margin-right:100px;">{{var\_one}}</h1>

<a [routerLink]="['childone']" ><b>Child\_one</b></a>

<router-outlet></router-outlet>

**page\_one.component.ts**

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

@Component({

selector: 'pageone',

templateUrl: './page-one.component.html',

styles: []

})

export class PageOneComponent implements OnInit {

private var\_one:any;

constructor(public route:ActivatedRoute) {

this.var\_one=this.route.snapshot.params["p\_id"]+"....."+

this.route.snapshot.params["p\_name"]+"...."+

this.route.snapshot.params["p\_cost"];

}

ngOnInit() {

}

}

**page\_two.component.html**

<p>page-two works!</p>

<h1 style="color: blue;margin-right:100px;">{{var\_two}}</h1>

<a [routerLink]="['childtwo']" ><b>Child\_two</b></a>

<router-outlet></router-outlet>

**page\_two.component.ts**

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'pagetwo',

templateUrl: './page-two.component.html',

styles: []

})

export class PageTwoComponent implements OnInit {

private var\_two:any;

constructor() {

this.var\_two="Welcome to pagetwo............!"

}

ngOnInit() {

}}

**page\_three.component.html**

<p>page-two works!</p>

<h1 style="color: blue;margin-right:100px;">{{var\_two}}</h1>

<a [routerLink]="['childtwo']" ><b>Child\_two</b></a>

<router-outlet></router-outlet>

**page\_three.component.ts**

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'pagethree',

templateUrl: './page-three.component.html',

styles: []

})

export class PageThreeComponent implements OnInit {

private var\_three:any;

constructor() {

this.var\_three="welcome to page three.......!";

}

ngOnInit() {

}}

**Index.component.html**

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<nav class="navbar navbar-expand-sm bg-light navbar-light">

<ul class="navbar-nav">

<li class="nav-item active">

<a [routerLink]="['/pageone',111,'p\_one',1000]" style="margin-right: 100px;">PageOne</a>

</li>

<li class="nav-item">

<a [routerLink]="['/pagetwo']" style="margin-right: 100px;">Pagetwo</a>

</li>

<li class="nav-item">

<a [routerLink]="['/pagethree']" style="margin-right: 100px;">Pagethree</a>

</li>

</ul>

</nav>

<!--

<a [routerLink]="['/pageone']" style="margin-right: 100px;">PageOne</a>

<a [routerLink]="['/pagetwo']" style="margin-right: 100px;">Pagetwo</a>

<a [routerLink]="['/pagethree']" style="margin-right: 100px;">Pagethree</a>

-->

<router-outlet></router-outlet>

</body>

</html>

**child\_one.component.html**

<h1 style="color: springgreen;">{{var\_four}}</h1>

**child\_one.component.ts**

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

@Component({

selector:"childone",

templateUrl:'./child-one.component.html'

})

export class childonecomponent

{

private var\_four:any;

constructor(){

this.var\_four="Welcome to child\_one component";

}

}

**child\_two.component.html**

<h1 style="color: tomato;">{{var\_five}}</h1>

**child\_two.component.ts**

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

import { ThrowStmt } from '@angular/compiler';

@Component({

selector:"childtwo",

templateUrl:"./child-two.component.html"

})

export class childtwocomponent

{

private var\_five:any;

constructor(){

this.var\_five="Welcome to child\_two component";

}

}

**child\_three.component.html**

<h1 style="color: violet;">{{var\_six}}</h1>

**child\_three.component.ts**

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

@Component({

selector:"childthree",

templateUrl:"./child-three.component.html"

})

export class childthreecomponent

{

private var\_six:any;

constructor(){

this.var\_six="Welcome to child\_two component";

}}

**app.routes.ts**

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

import { PageOneComponent } from '../components/page-one.component';

import { PageTwoComponent } from '../components/page-two.component';

import { PageThreeComponent } from '../components/page-three.component';

import { childonecomponent } from '../components/child-one.component';

import { childtwocomponent } from '../components/child-two.component';

import { childthreecomponent } from '../components/child-three.component';

import { authGuards } from '../guards/auth.guards';

export const appRoutes:Routes = [

{path:"pageone/:p\_id/:p\_name/:p\_cost",component:PageOneComponent,

children:[{path:"childone",component:childonecomponent}],

canActivate:[authGuards]},

{path:"pagetwo",component:PageTwoComponent,

children:[{path:"childtwo",component:childtwocomponent}],

canDeactivate:[authGuards]},

{path:"pagethree",component:PageThreeComponent,

children:[{path:"childthree",component:childthreecomponent}],

canActivateChild:[authGuards]}

];

**auth.guards.ts**

import { Injectable } from "@angular/core";

@Injectable({

providedIn:"root"

})

export class authGuards{

canActivate():boolean{

return confirm("do you want to enter into first page ??");

};

canDeactivate():boolean{

return confirm("do you want to leave second page ??");

};

canActivateChild():boolean{

return confirm("do you want to enter into 3rd child ??");

};};

**App.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { PageOneComponent } from './components/page-one.component';

import { IndexComponent } from './components/index.component';

import { PageTwoComponent } from './components/page-two.component';

import { PageThreeComponent } from './components/page-three.component';

import { RouterModule } from '@angular/router';

import { appRoutes } from './routes/app.routes';import { childonecomponent } from './components/child-one.component';

import { childtwocomponent } from './components/child-two.component';

import { childthreecomponent } from './components/child-three.component';

@NgModule({

declarations: [

AppComponent,

PageOneComponent,

IndexComponent,

PageTwoComponent,

PageThreeComponent,

childonecomponent,

childtwocomponent,

childthreecomponent

],

imports: [

BrowserModule,RouterModule.forRoot(appRoutes)

],

providers: [],

bootstrap: [IndexComponent]

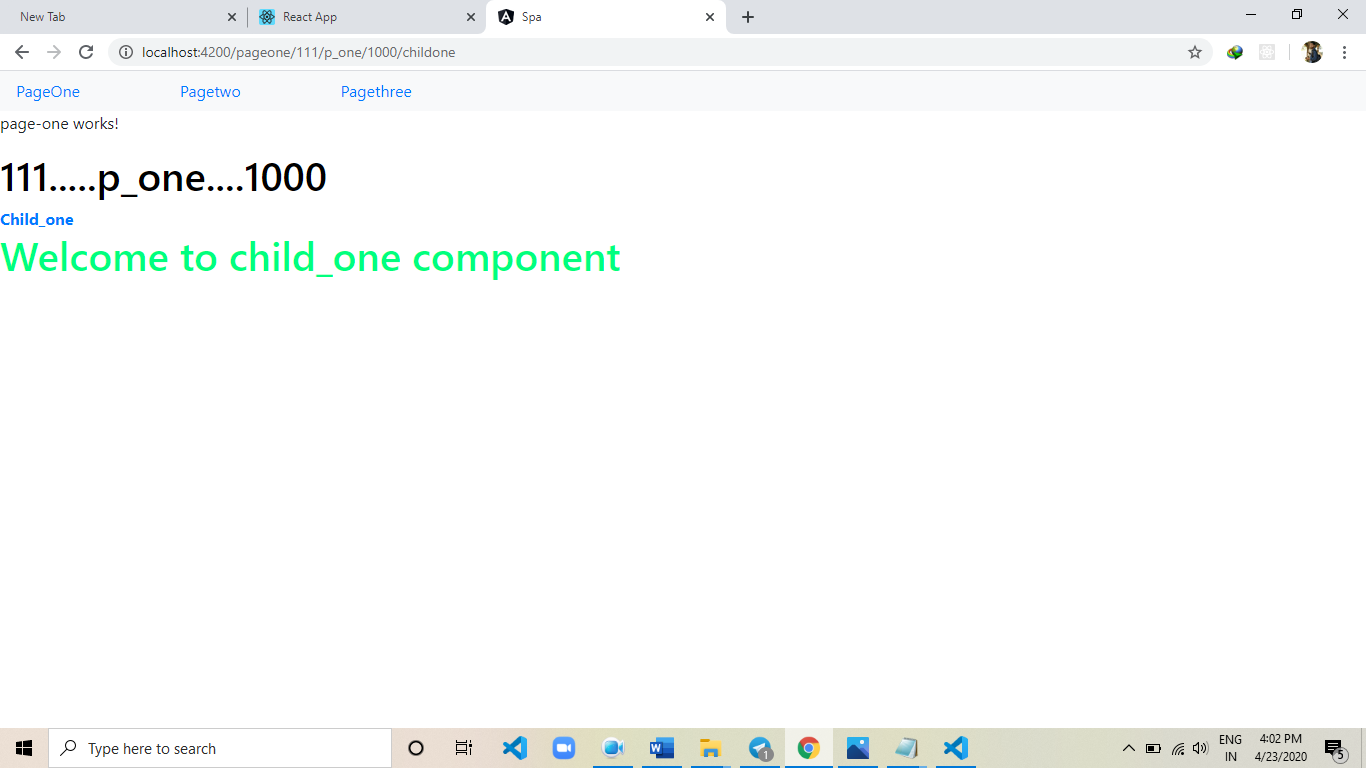
})

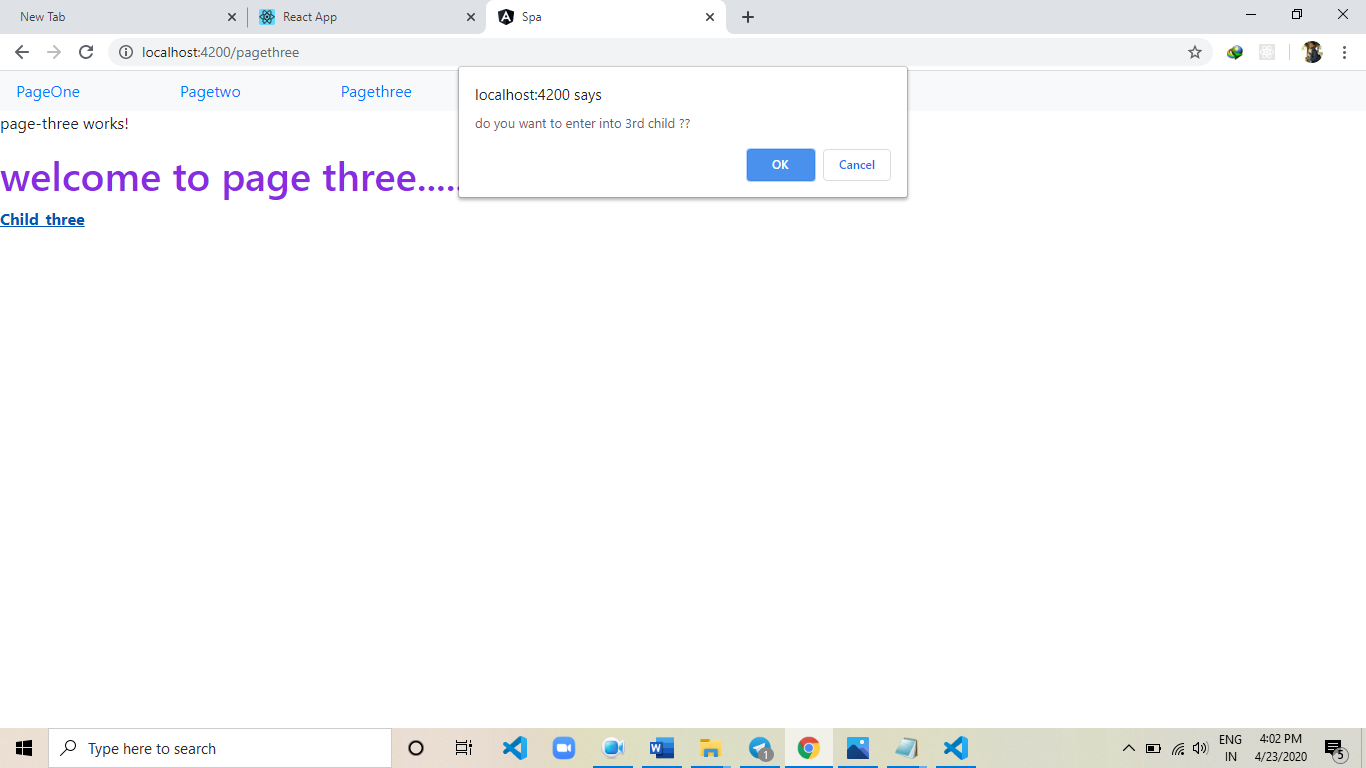
export class AppModule { }

**index.html**

<body> <index></index> </body>

**Result:**





**Chapter-14(Crud Operations)**

**CRUD Operations:**

**Example:**

**Directory Structure:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

crudApp

server

server.js

src

app

components

crud.component.html

crud.component.ts

services

FetchService

InsertService

UpdateService

DeleteService

app.module.ts

index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1) make the MySQL DataBase Ready for CRUD Operations.**

Default Password : root

> create schema angular7am;

- automatically "angular7am" DataBase will create.

> use angular7am;

- we can switch to angular7am DataBase.

> create table products(p\_id integer, p\_name varchar(20),

p\_cost integer);

- automatically "products" table will create.

> insert into products values(111,"p\_one",10000);

- automatically record will be inserted.

> select \* from products;

- we can fetch the data from products table.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

host : localhost

user : root

Password: admin

database: angular7am

table : products

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**2) create the angular project**

> ng new crudApp

**3) switch to angular application**

> cd crudApp

**4) download the following node modules**

=> express

=> mysql

=> cors

=> body-parser

* "express" module used to develop the rest apis.
* "mysql" module used to interact with the mysql database.
* "cors" module used to enable the ports communication.
* "body-parser" module used to read the post parameters.

- we will download above modules by using "yarn" tool.

**> yarn add express mysql cors body-parser --save**

**5) develop rest apis by using nodejs.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

crudApp

server

server.js

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**server.js:**

//import the modules

//require() function used to import the modules in nodejs

let express = require("express");

let mysql = require("mysql");

let cors = require("cors");

let bodyparser = require("body-parser");

//create the rest object

let app = express();

//enable the cors

app.use(cors());

//set the JSON as MIME Type

app.use(bodyparser.json());

//read the JSON

app.use(bodyparser.urlencoded({extended:false}));

//create the connection object

let connection=mysql.createConnection({

    host:"localhost",

    user:"root",

    password:"admin",

    database:"angular7am"

});

//connect to database

connection.connect();

//create the get request

app.get("/fetch",(req,res)=>{

    connection.query(`select \* from products`,(err,records,fields)=>{

        if(err) throw err;

        else{

            res.send(records);

        }

    });

});

//create the post request

app.post("/insert",(req,res)=>{

    connection.query(`insert into products values(${req.body.p\_id},'${req.body.p\_name}',${req.body.p\_cost})`,(err,result)=>{

        if(err) throw err;

        else{

           res.send({insert:"success"});

        }

    });

});

//create the put request

app.post("/update",(req,res)=>{

    connection.query(`update products set p\_name='${req.body.p\_name}',p\_cost=${req.body.p\_cost} where p\_id=${req.body.p\_id}`,

                    (err,result)=>{

        if(err) throw err;

        else{

           res.send({update:"success"});

        }

    });

});

//delete request

app.post("/delete",(req,res)=>{

    connection.query(`delete from products where p\_id=${req.body.p\_id}`,(err,result)=>{

        if(err) throw err;

        else{

          res.send({delete:"success"});

        }

    });

});

//assign the port no

app.listen(8080);

console.log("server listening the port no.8080");

**6) start the node server**

> cd server

> node server

**7) test the rest apis by using Postman**

=> http://localhost:8080/fetch (GET)

=> http://localhost:8080/insert (POST)

=> http://localhost:8080/update (PUT)

=> http://localhost:8080/delete (DELETE)

**8) create the services**

=> FetchService

=> InsertService

=> UpdateService

=> DeleteService

> ng g s services/fetch --skipTests

> ng g s services/insert --skipTests

> ng g s services/update --skipTests

> ng g s services/delete --skipTests

=> "FetchService" is ready with "getProducts()" function.

=> "InsertService" is ready with "insertRecord(data)" function.

=> "UpdateService" is ready with "updateRecord(data)" function.

=> "DeleteService" is ready with "deleteRecord(data)" function.

**=> Fetch.Service.ts**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

  providedIn: 'root'

})

export class FetchService {

  constructor(private http:HttpClient) { }

  public getProducts():Observable<any>{

    return this.http.get("http://localhost:3306/fetch");

  }; };

**=> Insert.Service.ts**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

  providedIn: 'root'

})

export class InsertService {

  constructor(private http:HttpClient) { }

  public insertRecord(data:any):Observable<any>{

    return this.http.post("http://localhost:3306/insert",data);

  };

}

**=> Update.Service.ts**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

  providedIn: 'root'

})

export class UpdateService {

  constructor(private http:HttpClient) { }

  public updateRecord(data:any):Observable<any>{

     return this.http.post("http://localhost:3306/update",data);

  };

};

**=> Delete.Service.ts**

import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

  providedIn: 'root'

})

export class DeleteService {

  constructor(private http:HttpClient) { }

  public deleteRecord(data:any):Observable<any>{

      console.log(data);

      return this.http.post("http://localhost:3306/delete",data);

  }; };

**9) create the component.**

> ng g c components/crud --skipTests -is --selector=crud --flat true

**Crud.component.ts**

import { Component, OnInit } from '@angular/core';

import { FetchService } from "../services/fetch.service";

import { InsertService } from "../services/insert.service";

import { UpdateService } from "../services/update.service";

import { DeleteService } from "../services/delete.service";

import { HttpErrorResponse } from "@angular/common/http";

import { ModalService } from "../\_modal/modal.service";

@Component({

  selector: 'crud',

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

  styles: []

})

export class CrudComponent implements OnInit {

  private result:any;

  private update\_pid:number;

  private update\_pname:string;

  private update\_pcost:number;

  bodyText:any;

  constructor(private fetch:FetchService,

              private insert:InsertService,

              private update:UpdateService,

              private remove:DeleteService,

              private service:ModalService) { }

  private errCallBack = (errRes:HttpErrorResponse)=>{

      if(errRes.error instanceof Error){

        console.log("client side error");

      }else{

        console.log("server side error");

      }

  };

  ngOnInit() {

    this.fetch.getProducts().subscribe((posRes)=>{

        this.result = posRes;

    },this.errCallBack);

  }

  deleteRecord(data):any{

      console.log(data);

      this.remove.deleteRecord({p\_id:data})

          .subscribe((posRes)=>{

            if(posRes.delete === "success"){

              let i =

                this.result

                  .findIndex((element,index)=>{

                  return element.p\_id === data;

               });

               this.result.splice(i,1);

            }else{

              alert("delete fail");

            }

      },this.errCallBack);

  }

  openUpdateModal(id: string,data:any) {

    this.bodyText = data;

    console.log(this.bodyText);

    this.update\_pid=data.p\_id;

    this.update\_pname=data.p\_name;

    this.update\_pcost=data.p\_cost;

    this.service.open(id);

  }

  closeUpdateModal(id: string) {

    this.update.updateRecord({"p\_id":this.update\_pid,"p\_name":this.update\_pname,"p\_cost":this.update\_pcost})

              .subscribe((posRes)=>{

                if(posRes.update === "success"){

                  let i = this.result.findIndex((element,index)=>{

                      return element.p\_id == this.update\_pid;

                  });

                  this.result[i].p\_name = this.update\_pname;

                  this.result[i].p\_cost = this.update\_pcost;

                  this.service.close(id);

                }

              },this.errCallBack);

  }

  cancel(id:string){

    this.service.close(id);

  }

  insertRecord(){

     this.service.open("insert");

  };

  insertR(id,data:any){

     this.insert.insertRecord(data)

                .subscribe((posRes)=>{

        if(posRes.insert === "success"){

            this.result.push(data);

        }else{

            alert("Insert Fail");

        }

        this.service.close(id);

     },this.errCallBack);

  }

  removeR(id){

    this.service.close(id);

  }

**Crud.component.html**<button class="glyphicon glyphicon-plus

               btn btn-success"

        (click)="insertRecord()"

        style="position: absolute;

               right: 0;

               padding: 10px;"></button>

<table border="1"

       cellpadding="30px"

       cellspacing="30px"

       align="center"

       style="font-size: 20px;

       text-align: center;">

    <thead style="background-color: gray;">

        <tr>

            <th>SNO</th>

            <th>P\_ID</th>

            <th>P\_NAME</th>

            <th>P\_COST</th>

            <th>EDIT</th>

            <th>DELETE</th>

        </tr>

    </thead>

    <tbody>

        <tr \*ngFor="let x of result;let i=index">

            <td>{{i+1}}</td>

            <td>{{x.p\_id}}</td>

            <td>{{x.p\_name}}</td>

            <td>{{x.p\_cost}}</td>

            <td><button

                 class="glyphicon glyphicon-edit"

                 (click)="openUpdateModal('edit',x)"></button></td>

            <td><button

                class="glyphicon glyphicon-trash"

                (click)="deleteRecord(x.p\_id)"></button></td>

        </tr>

    </tbody>

</table>

<jw-modal id="edit">

    <h1>Edit</h1>

    <p>P\_ID: <input type="number"

                    [(ngModel)]="update\_pid" /></p>

    <p>P\_NAME: <input type="text"

                    [(ngModel)]="update\_pname" /></p>

    <p>P\_COST: <input type="number"

                    [(ngModel)]="update\_pcost" /></p>

    <button (click)="closeUpdateModal('edit');">Update</button>

    <button (click)="cancel('edit');">Cancel</button>

</jw-modal>

<jw-modal id="insert">

    <h1>Insert</h1>

    <p>P\_ID: <input type="number"

                    [(ngModel)]="insert\_pid" /></p>

    <p>P\_NAME: <input type="text"

                    [(ngModel)]="insert\_pname" /></p>

    <p>P\_COST: <input type="number"

                    [(ngModel)]="insert\_pcost" /></p>

    <button (click)="insertR('insert',

                             {'p\_id':insert\_pid,

                            'p\_name':insert\_pname,

                            'p\_cost':insert\_pcost});">Insert</button>

    <button (click)="removeR('insert');">Cancel</button>

</jw-modal>

}

**Displaying the Model Popup to Perform "Update" and "Insert" Operations**

**1) download 3rd party library and place in "app"**

Directory

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

src

app

\_model

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- where "\_model" is the 3rd party directory.

- This directory containes "Popup" design.

**2) add the dependency with the help of "app.module.ts" file.**

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { CrudComponent } from './components/crud.component';

import { HttpClientModule } from '@angular/common/http';

import { FormsModule } from "@angular/forms";

import { ModalModule  } from "./\_modal/modal.module";

@NgModule({

  declarations: [

    AppComponent,

    CrudComponent

  ],

  imports: [

    BrowserModule,HttpClientModule,FormsModule,ModalModule

  ],

  providers: [],

  bootstrap: [CrudComponent]

})

export class AppModule { }

**3) design the Popup by using HTML Custom Element.**

**4) handle the Popup Events**

**=> show the Popup with "Edit" icon event.**

<button class="glyphicon glyphicon-edit"

(click)="openUpdateModal('edit',x)"></button>

openUpdateModal(id: string,data:any) {

this.bodyText = data;

console.log(this.bodyText);

this.update\_pid=data.p\_id;

this.update\_pname=data.p\_name;

this.update\_pcost=data.p\_cost;

this.service.open(id);

}

closeUpdateModal(id: string) {

this.update.updateRecord({"p\_id":this.update\_pid,

"p\_name":this.update\_pname,

"p\_cost":this.update\_pcost})

.subscribe((posRes)=>{

if(posRes.update === "success"){

let i = this.result

.findIndex((element,index)=>{

return element.p\_id ==

this.update\_pid;

});

this.result[i].p\_name =

this.update\_pname;

this.result[i].p\_cost =

this.update\_pcost;

this.service.close(id);

}

},this.errCallBack);

}

cancel(id:string){

this.service.close(id);

}

**implementation of insert functionality**

--------------------------------------

=> we have following functions

1) **insertRecord()** function used to show the popup

2) **insert(-,-)** used to insert the data into database.

3) **remove(-)** used to remove the popup.

**Index.html**

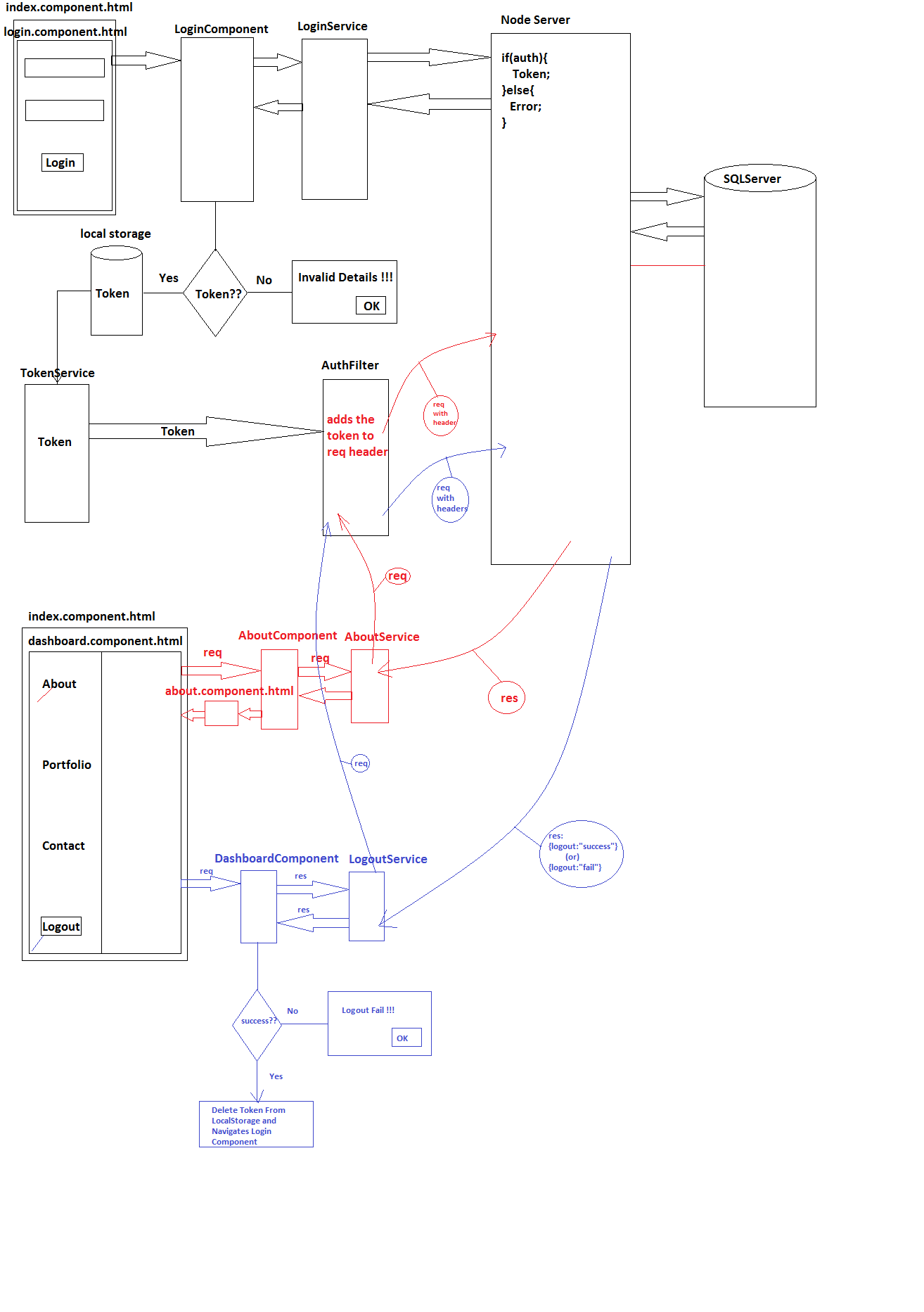
<body>

    <crud></crud>

</body>

**Result:**

**MiniProject Implementation**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Frontend : Angular

Backend : NodeJS

Database : SQLServer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1) make the SQLServer Ready for MiniProject Implementation.**

- we need four tables for Implementation.

=> login\_details

=> about

=> portfolio

=> contact

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server : localhost

user : sa

password : 123

database : miniproject

tables : login\_details

about

portfolio

contact

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**2) create the angular application**

> ng new miniproject

**3) switch to miniproject**

> cd miniproject

**4) download following node modules**

=> express

=> mssql

=> body-parser

=> cors

=> jwt-simple

- "express" module used to develop the rest apis.

- "mssql" module used to interact with the SQLServer.

- "body-parser" module used to read the post parameters.

- "cors" module used to enable the ports communication.

- "jwt-simple" module used to generate the token.

- we will download above modules by using "yarn" tool.

**Command:**

**> yarn add express mssql body-parser cors jwt-simple --save**

**5) develop rest apis by using nodejs**

**Directory structure**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

miniproject

server

config

db\_properties.js

token.js

generateToken.js

auth.js

login

login.js

about

about.js

portfolio

portfolio.js

contact

contact.js

logout

logout.js

server.js

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-"db\_properties.js" file used to maintain the database properties(SQLServer)

- "token.js" file used to store the server side token.

- "generateToken.js" file used to generate the tokens.

- "auth.js" file used to compare the server side tokens.

- "login.js" file used to generate the "login rest api" with module.

- "about.js","portfolio.js","contact.js" and "logout.js" files are used to develop the coresponding rest apis.

- "server.js" file is the main node server.

**db\_properties.js**

const obj = {

server : "localhost",

user : "sa",

password : "123",

database : "miniproject"

};

module.exports = obj;

**token.js**

let obj = {

token : ""

};

module.exports = obj;n.js

**generateToken.js**

//converting readable data to unreadable data with custom password called as token.

let jwt = require("jwt-simple");

let genToken = (data,password)=>{

return jwt.encode(data,password);

};

module.exports = genToken;

**auth.js**

//import token.js file

//it containes server side token

let obj = require("./token");

//create the function

let auth = (req,res,next)=>{

//read headers

let allHeaders = req.headers;

let c\_token = allHeaders.token;

//compare the tokens

if(c\_token === obj.token){

next();

}else{

res.send({'message':'unauthorized user'});

}

};

module.exports = auth;

**login.js**

//it is used to create and export the module

//import mssql module

let mssql = require("mssql");

let login = require("express").Router()

.post("/",(req,res)=>{

mssql.connect(require("../config/db\_properties"),(err)=>{

if(err) throw err;

else{

let queryObj = new mssql.Request(); queryObj.query(`select \* from login\_details where uname='${req.body.uname}' and upwd='${req.body.upwd}'`, (err,records)=>{

if(err) throw err;

else{

if(records.recordset.length>0){

let token = require("../config/generateToken")({'uname':req.body.uname,

'upwd':req.body.upwd},"hr@tcs.com");

require("../config/token").token = token;

res.send({'login':'success','token':token});

}else{

res.send({'login':'fail'});

}

}

mssql.close();

});

}

});

});

module.exports = login;

**about.js**

let mssql = require("mssql");

let about = require("express").Router().get("/",

[require("../config/auth")],(req,res)=>{

mssql.connect(require("../config/db\_properties"),

(err)=>{

if(err) throw err;

else{

let queryObj = new mssql.Request();

queryObj.query(`select \* from about`,

(err,records)=>{

if(err) throw err;

else{

res.send(records);

}

mssql.close();

});

}

});

});

module.exports = about;

**portfolio.js**

let mssql = require("mssql");

let portfolio = require("express").Router().get("/",

[require("../config/auth")],(req,res)=>{

mssql.connect(require("../config/db\_properties"),

(err)=>{

if(err) throw err;

else{

let queryObj = new mssql.Request();

queryObj.query(`select \* from portfolio`,

(err,records)=>{

if(err) throw err;

else{

res.send(records);

}

mssql.close();

});

}

});

});

module.exports = portfolio;

**contact.js**

let mssql = require("mssql");

let contact = require("express").Router().get("/",

[require("../config/auth")],(req,res)=>{

mssql.connect(require("../config/db\_properties"),

(err)=>{

if(err) throw err;

else{

let queryObj = new mssql.Request();

queryObj.query(`select \* from contact`,

(err,records)=>{

if(err) throw err;

else{

res.send(records);

}

mssql.close();

});

}

});

});

module.exports = contact;

**logout.js**

let logout =

require("express").Router()

.get("/",[require("../config/auth")],(req,res)=>{

require("../config/token").token = "";

let obj = require("../config/token");

if(obj.token === ""){

res.send({logout:"success"});

}else{

res.send({logout:"fail"});

} });

module.exports = logout;

**server.js**

let express = require("express");

let bodyparser = require("body-parser");

let cors = require("cors");

let app = express();

app.use(cors());

app.use(bodyparser.json());

app.use(bodyparser.urlencoded({extended:false}));

app.use("/login",require("./login/login"));

app.use("/about",require("./about/about"));

app.use("/portfolio",require("./portfolio/portfolio"));

app.use("/contact",require("./contact/contact"));

app.use("/logout",require("./logout/logout"));

app.listen(8080);

console.log("server listening the port no.8080");

**6) start the node server**

> cd miniproject

> cd server

> node server

**7) test the rest apis by using Postman.**

=> http://localhost:8080/login (POST)

=> http://localhost:8080/about (GET)

=> http://localhost:8080/portfolio (GET)

=> http://localhost:8080/contact (GET)

=> http://localhost:8080/logout (GET)

**8) divide the application into modules**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AppModule(Main Module)(BootStrap)

LoginModule DashboardModule

- AboutModule

- PortfolioModule

- ContactModule

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**ContactModule**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

src

app

contact

services

contact.service.ts

components

contact.component.ts

contact.component.html

module

contact.module.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**contact.service.ts**

import { Injectable } from "@angular/core";

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn:"root"

})

export class ContactService{

constructor(private http:HttpClient){}

public getData():Observable<any>{

return this.http.get("http://localhost:8080/contact");

};

};

**contact.component.ts**

import { Component } from "@angular/core";

import { ContactService } from '../services/contact.service';

import errCallBack from 'src/app/config/errCallBack';

@Component({

selector:"contact",

templateUrl:"./contact.component.html"

})

export class ContactComponent{

private result:any;

constructor(private service:ContactService){}

ngOnInit(){

this.service.getData().subscribe((posRes)=>{

this.result = posRes;

},errCallBack);

}

};

**contact.component.html**

<h1>{{result | json}}</h1>

**contact.module.ts**

import { NgModule } from "@angular/core";

import { ContactComponent } from '../components/contact.component';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { TokenModule } from 'src/app/token/token.module';

import { ContactService } from '../services/contact.service';

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

export const appRoutes:Routes = [

{path:"",component:ContactComponent}

];

@NgModule({

declarations:[ContactComponent],

imports:[CommonModule,

HttpClientModule,

TokenModule,

RouterModule.forChild(appRoutes)],

providers:[ContactService],

exports:[ContactComponent]

})

export class ContactModule{}

**TokenModule**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

src

app

token

gettoken.service.ts

auth.interceptor.ts

token.module.ts

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**gettoken.service.ts**

//this service used to fetch the token from local storage.

import { Injectable } from "@angular/core";

@Injectable({

providedIn:"root"

})

export class FetchTokenService{

public getToken():string{

let str = window.localStorage.getItem("login\_details");

let obj = JSON.parse(str);

return obj.token;

};

};

**auth.interceptor.ts**

import { Injectable } from "@angular/core";

import { FetchTokenService } from './gettoken.service';

import { HttpRequest,HttpHandler,HttpEvent } from "@angular/common/http";

import { Observable } from "rxjs";

@Injectable({

providedIn:"root"

})

export class authInterceptor{

constructor(private service:FetchTokenService){}

intercept(req:HttpRequest<any>,handler:HttpHandler)

:Observable<HttpEvent<any>>{

if(req.url == "http://localhost:8080/login"){

return handler.handle(req);

}else{

const req1 = req.clone({

setHeaders:{

token:this.service.getToken()

}

});

return handler.handle(req1);

}

};

};

**token.module.ts**

import { NgModule } from "@angular/core";

import { CommonModule } from '@angular/common';

import { FetchTokenService } from './gettoken.service';

import { HTTP\_INTERCEPTORS } from '@angular/common/http';

import { authInterceptor } from './auth.interceptor';

@NgModule({

imports:[CommonModule],

providers:[FetchTokenService,{

provide:HTTP\_INTERCEPTORS,

useClass:authInterceptor,

multi:true

}]

})

export class TokenModule{}

**PortfolioModule**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

src

app

portfolio

services

portfolio.service.ts

components

portfolio.component.ts

portfolio.component.html

module

portfolio.module.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**portfolio.service.ts**

import { Injectable } from "@angular/core";

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn:"root"

})

export class PortfolioService{

constructor(private http:HttpClient){}

public getData():Observable<any>{

return this.http.get("http://localhost:8080/portfolio");

};

};

**portfolio.component.ts**

import { Component } from "@angular/core";

import { PortfolioService } from '../services/portfolio.service';

import errCallBack from 'src/app/config/errCallBack';

@Component({

selector:"portfolio",

templateUrl:"./portfolio.component.html"

})

export class PortfolioComponent{

private result:any;

constructor(private service:PortfolioService){}

ngOnInit(){

this.service.getData().subscribe((posRes)=>{

this.result = posRes;

},errCallBack);

}

};

**portfolio.component.html**

<h1>{{result | json}}</h1>

**portfolio.module.ts**

import { NgModule } from "@angular/core";

import { PortfolioComponent } from '../components/portfolio.component';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { TokenModule } from 'src/app/token/token.module';

import { PortfolioService } from '../services/portfolio.service';

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

export const appRoutes:Routes = [

{path:"",component:PortfolioComponent}

];

@NgModule({

declarations:[PortfolioComponent],

imports:[CommonModule,

HttpClientModule,

TokenModule,

RouterModule.forChild(appRoutes)],

providers:[PortfolioService],

exports:[PortfolioComponent]

})

export class PortfolioModule{}

**AboutModule**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

src

app

about

services

about.service.ts

components

about.component.ts

about.component.html

module

about.module.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**about.service.ts**

import { Injectable } from "@angular/core";

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn:"root"

})

export class AboutService{

constructor(private http:HttpClient){}

public getData():Observable<any>{

return this.http.get("http://localhost:8080/about");

};

};

**about.component.ts**

import { Component } from "@angular/core";

import { AboutService } from '../services/about.service';

import errCallBack from 'src/app/config/errCallBack';

@Component({

selector:"about",

templateUrl:"./about.component.html"

})

export class AboutComponent{

private result:any;

constructor(private service:AboutService){}

ngOnInit(){

this.service.getData().subscribe((posRes)=>{

this.result = posRes;

},errCallBack);

}

};

**about.component.html**

<h1>{{result | json}}</h1>

**about.module.ts**

import { NgModule } from "@angular/core";

import { AboutComponent } from '../components/about.component';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { TokenModule } from 'src/app/token/token.module';

import { AboutService } from '../services/about.service';

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

export const appRoutes:Routes = [

{path:"",component:AboutComponent}

];

@NgModule({

declarations:[AboutComponent],

imports:[CommonModule,

HttpClientModule,

TokenModule,

RouterModule.forChild(appRoutes)],

providers:[AboutService],

exports:[AboutComponent]

})

export class AboutModule{}

**DashboardModule**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

dashboard

services

logout.service.ts

components

dashboard.component.ts

dashboard.component.html

module

dashboard.module.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**logout.service.ts**

import { Injectable } from "@angular/core";

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn:"root"

})

export class logoutService{

constructor(private http:HttpClient){}

public logout():Observable<any>{

return this.http.get("http://localhost:8080/logout")

};

};

**dashboard.component.ts**

import { Component } from "@angular/core";

import { logoutService } from "../services/logout.service";

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

import errCallBack from "../../config/errCallBack";

@Component({

selector:"dashboard",

templateUrl:"./dashboard.component.html"

})

export class dashboardComponent{

constructor(private router:Router,

private service:logoutService){}

logout():any{

this.service.logout().subscribe((posRes)=>{

if(posRes.logout == "success"){

window.localStorage.removeItem("login\_details");

this.router.navigate(["/"]);

}

},errCallBack);

};};

**dashboard.component.html**

<a [routerLink]="['about']" style="margin-right: 100px;">

<b>About</b>

</a>

<a [routerLink]="['portfolio']" style="margin-right: 100px;">

<b>Portfolio</b>

</a>

<a [routerLink]="['contact']" style="margin-right: 100px;">

<b>Contact</b>

</a>

<button (click)="logout()">Logout</button>

<br><br>

<router-outlet></router-outlet>

**dashboard.module.ts**

import { NgModule } from "@angular/core";

import { dashboardComponent } from '../components/dashboard.component';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { TokenModule } from 'src/app/token/token.module';

import { logoutService } from '../services/logout.service';

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

export const appRoutes:Routes = [

{path:"",component:dashboardComponent,

children:[{path:"about",loadChildren:"src/app/about/module/about.module#AboutModule"},

{path:"portfolio",loadChildren:"src/app/portfolio/module/portfolio.module#PortfolioModule"},

{path:"contact",loadChildren:"src/app/contact/module/contact.module#ContactModule"}]}

];

@NgModule({

declarations:[dashboardComponent],

imports:[CommonModule,

HttpClientModule,

TokenModule,

RouterModule.forChild(appRoutes)],

providers:[logoutService],

exports:[dashboardComponent]

})

export class DashboardModule{}

**LoginModule**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

login

services

login.service.ts

components

login.component.ts

login.component.html

module

login.module.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**login.service.ts**

import { Injectable } from "@angular/core";

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn:"root"

})

export class loginService{

constructor(private http:HttpClient){}

public login(data:any):Observable<any>{

return this.http.post("http://localhost:8080/login",data);

};

};

**login.component.ts**

import { Component } from "@angular/core";

import { loginService } from "../services/login.service";

import errCallBack from "../../config/errCallBack";

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

@Component({

selector:"login",

templateUrl:"./login.component.html"

})

export class loginComponent{

constructor(private service:loginService,

private router:Router){}

public login(data:any){

this.service.login(data).subscribe((posRes)=>{

if(posRes.login == "success"){

let str = JSON.stringify(posRes);

window.localStorage.setItem("login\_details",str);

this.router.navigate(["/dashboard"]);

}else{

alert("Login Fail");

}

},errCallBack);

}; };

**login.component.html**

<fieldset>

<legend>Login</legend>

<input type="text" [(ngModel)]="uname" placeholder="User Name">

<br><br>

<input type="password" [(ngModel)]="upwd" placeholder="User Password">

<br><br>

<button (click)="login({'uname':uname,

'upwd':upwd})">Login</button>

</fieldset>

**login.module.ts**

import { NgModule } from "@angular/core";

import { loginComponent } from '../components/login.component';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { FormsModule } from "@angular/forms";

import { loginService } from '../services/login.service';

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

export const appRoutes:Routes = [

{path:"",component:loginComponent}

];

@NgModule({

declarations:[loginComponent],

imports:[CommonModule,

HttpClientModule,

FormsModule,

RouterModule.forChild(appRoutes)],

providers:[loginService],

exports:[loginComponent]

})

export class LoginModule{}

**AppModule**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

app.component.ts

app.component.html

app.module.ts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**app.component.ts**

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'miniproject';

}

**app.component.html**

<router-outlet></router-outlet>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { LoginModule } from './login/module/login.module';

import { DashboardModule } from './dashboard/module/dashboard.module';

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

export const appRoutes:Routes = [

{path:"",loadChildren:"./login/module/login.module#LoginModule"}, {path:"dashboard",

loadChildren:"./dashboard/module/dashboard.module#DashboardModule"}

];

@NgModule({

declarations: [

AppComponent

],

Imports:[ BrowserModule,

LoginModule,

DashboardModule,

RouterModule.forRoot(appRoutes)

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**9) execute the miniproject**

**Terminal-1**

**----------**

> cd miniproject

> cd server

> node server

**Terminal-2**

**----------**

> cd miniproject

> ng s -o

**Result:**

