**Section11 Http Request in Angular**

**Notes: -**

**1-we will allow angular to access to store data (SQL or NoSQL) through server (REST or Graph QL) which is responsible to communicate between angular and data store**

**2-Anatomy of Http Request**

**A-Http Verb 🡪 POST, GET ,PUT , PATCH**

**B-URL (API Endpoint) 🡪 /posts/1**

**C-Header (Metadata) 🡪 {“Content-Type”:”application/json”}**

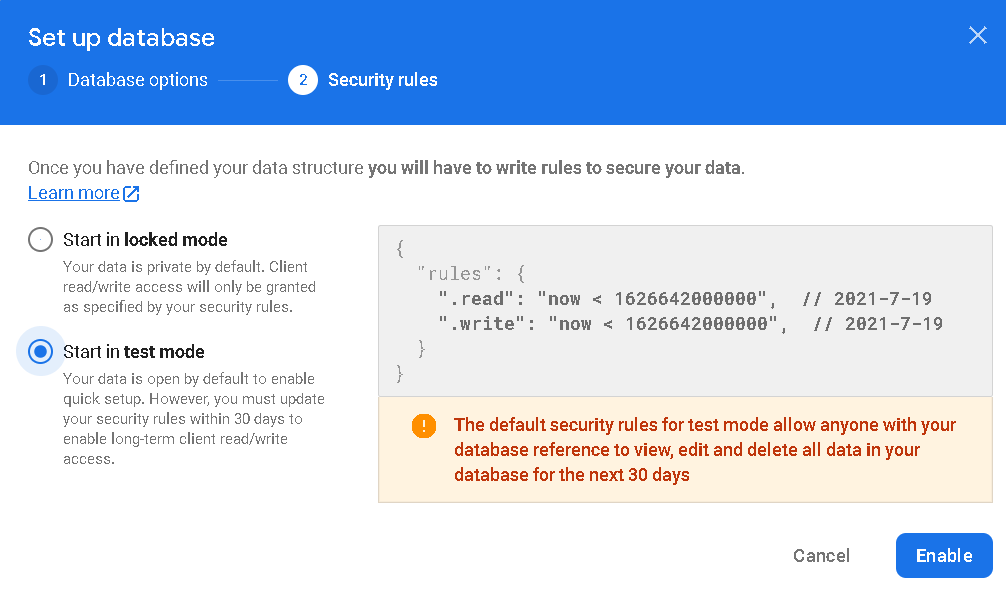
**D-Body (for POST, PUT,PATCH) 🡪 {“title”:”New POST”}**

**E-Params (for GET ,PUT) 🡪 /id=1&gender=male**

**3-we will sue firebase which is whole backend solution services which you can configure all backend API (not only just database)**

**Steps: -**

**1-on firebase > create new project > Firestore database > create database and choose test mode as below**



**(Angular takes the angular JavaScript and automatically convert it into json data)**

**(Angular using observable as perfect containers to handle async http request)**

**(When send post request from angular to firebase it will send two request one for check that if the request is allowed and the another to send payload post request)**

**(When send get request from angular to firebase it will send only one request)**

**2-we will use observable operator to handle with response when observer receive the data**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-md-6 col-md-offset-3">**

**<form #postForm="ngForm" (ngSubmit)="onCreatePost(postForm.value)">**

**<div class="form-group">**

**<label for="title">Title</label>**

**<input type="text" class="form-control" id="title" required ngModel name="title" />**

**</div>**

**<div class="form-group">**

**<label for="content">Content</label>**

**<textarea class="form-control" id="content" required ngModel name="content"></textarea>**

**</div>**

**<button class="btn btn-primary" type="submit" [disabled]="!postForm.valid">Send Post</button>**

**</form>**

**</div></div>**

**<hr />**

**<div class="row">**

**<div class="col-xs-12 col-md-6 col-md-offset-3">**

**<button class="btn btn-primary" (click)="onFetchPosts()">Fetch Posts</button>**

**|**

**<button class="btn btn-danger" [disabled]="loadedPosts.length < 1"(click)="onClearPosts()">**

**Clear Posts</button>**

**</div></div>**

**<div class="row">**

**<div class="col-xs-12 col-md-6 col-md-offset-3">**

**<p \*ngIf="loadedPosts.length < 1 && !isLoading">No posts available!</p>**

**<ul class="list-group" \*ngIf="loadedPosts.length >= 1 && !isLoading">**

**<li class="list-group-item" \*ngFor="let post of loadedPosts">**

**<h1>{{post.title}}</h1>**

**<p>{{post.content}}</p>**

**</li></ul>**

**<p \*ngIf="isLoading">Loading...</p></div></div></div>**

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

**import { HttpClient } from '@angular/common/http';**

**import {map} from 'rxjs/operators';**

**import { PostModel } from './models/post-model';**

**@Component({**

**selector: 'app-root',**

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

**styleUrls: ['./app.component.css'],})**

**export class AppComponent implements OnInit {**

**loadedPosts:PostModel[] = [];**

**isLoading:boolean = false;**

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

**ngOnInit() {this.fetchPosts();}**

**onCreatePost(postData: PostModel) {console.log(postData);**

**this.http**

**.post('https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json',postData)**

**.subscribe((res) => {console.log(res);});}**

**onFetchPosts() {this.fetchPosts();}**

**onClearPosts() {}**

**private fetchPosts() {**

**this.isLoading = true;**

**//you can map in generate way as below**

**this.http**

**.get<{ [key: string]: PostModel }>(**

**'https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json')**

**.pipe(**

**map((responeData: { [key: string]: PostModel }) => {**

**const postsArray = [];**

**for (const key in responeData) {**

**if (responeData.hasOwnProperty(key)) {**

**postsArray.push({ ...responeData[key], id: key });}}**

**return postsArray;}))**

**.subscribe((posts) => {**

**this.isLoading = false;**

**this.loadedPosts = posts;**

**console.log(posts);});}}**

**Lesson02 Using Service for HTTP Request**

**Notes: -**

**1-we will use service to handle the DRY principle**

**(By return observables on the service layer and subscribe it on the component level)**

**2-we will pass header to the request such as authorization, content-type:application/json**

**and can also pass query parameters**

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

**import { Observable,throwError } from 'rxjs';**

**import { PostModel } from '../models/post-model';**

**import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';**

**import { catchError, map } from 'rxjs/operators';**

**@Injectable({providedIn: 'root',})**

**export class PostService {**

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

**post(model: PostModel):Observable<any> {**

**return this.http.post('https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json'**

**,model);}**

**get():Observable<PostModel[]> {**

**//we can pass parameter on the request URI as below**

**let searchParams = new HttpParams();**

**searchParams = searchParams.append('print','pretty').append('custom','key');**

**//we pass header on the request URI and pass URL params as below**

**return this.http.get<{ [key: string]: PostModel }>(**

**'https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json',{**

**headers:new HttpHeaders({'accept':  'application/json','custom-header':'hello'}),**

**params:searchParams})**

**//we are using pipe in order to manipulate to PostModel[] as below**

**.pipe(map((responeData: { [key: string]: PostModel }) => {**

**const postsArray: PostModel[] = [];**

**for (const key in responeData) {**

**if (responeData.hasOwnProperty(key)) {**

**postsArray.push({ ...responeData[key], id: key });}}**

**return postsArray;}),**

**//we can apply catchError in case there is some happen on the server side and we capture it also //on the observer side**

**catchError(err => {**

**return throwError(err);}))}**

**delete(){**

**return this.http.delete(**

**'https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json');}}**

**//on the component level side we handle the error message as below**

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

**import { HttpClient } from '@angular/common/http';**

**import { map } from 'rxjs/operators';**

**import { PostModel } from './models/post-model';**

**import { PostService } from './services/post.service';**

**import { NgForm } from '@angular/forms';**

**import { Subscription } from 'rxjs';**

**@Component({**

**selector: 'app-root',**

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

**styleUrls: ['./app.component.css'],})**

**export class AppComponent implements OnInit, OnDestroy {**

**loadedPosts: PostModel[] = [];**

**isLoading: boolean = false;**

**error = null;**

**@ViewChild('postForm') form: NgForm;**

**//we using subscription to register the observer subscription to unsubscribe onDestroy**

**sub: Subscription;**

**constructor(private http: HttpClient, private postService: PostService) {}**

**ngOnInit() {this.fetchPosts();}**

**ngOnDestroy(): void {this.sub.unsubscribe();}**

**onCreatePost(postData: PostModel) {**

**this.sub = this.postService.post(postData).subscribe(**

**(res) => {console.log(res);},**

**(error) => {this.error = error.message;});}**

**onFetchPosts() {this.fetchPosts();}**

**onClearPosts() {**

**this.form.reset();**

**this.sub = this.postService.delete().subscribe(**

**() => {this.loadedPosts = [];},**

**(error) => {this.error = error.message;});}**

**private fetchPosts() {**

**this.isLoading = true;**

**this.error = null;**

**this.sub = this.postService**

**.get()**

**.subscribe(**

**(posts) => {**

**this.isLoading = false;**

**this.loadedPosts = posts;**

**console.log(posts);},**

**(error) => {**

**this.isLoading = false;**

**this.error = error.message;});}**

**onHandleError(){this.fetchPosts();}}**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-md-6 col-md-offset-3">**

**<form #postForm="ngForm" (ngSubmit)="onCreatePost(postForm.value)">**

**<div class="form-group">**

**<label for="title">Title</label>**

**<input type="text" class="form-control" id="title" required ngModel name="title" />**

**</div>**

**<div class="form-group">**

**<label for="content">Content</label>**

**<textarea class="form-control" id="content" required ngModel name="content"></textarea></div>**

**<button class="btn btn-primary" type="submit" [disabled]="!postForm.valid">Send Post</button>**

**</form></div></div>**

**<hr />**

**<div class="row">**

**<div class="col-xs-12 col-md-6 col-md-offset-3">**

**<button class="btn btn-primary" (click)="onFetchPosts()">Fetch Posts</button>|**

**<button class="btn btn-danger" [disabled]="loadedPosts.length < 1" (click)="onClearPosts()">**

**Clear Posts**

**</button></div></div>**

**<div class="row">**

**<div class="col-xs-12 col-md-6 col-md-offset-3">**

**<div \*ngIf="!error">**

**<p \*ngIf="loadedPosts.length < 1 && !isLoading">No posts available!</p>**

**<ul class="list-group" \*ngIf="loadedPosts.length >= 1 && !isLoading">**

**<li class="list-group-item" \*ngFor="let post of loadedPosts">**

**<h1>{{post.title}}</h1><p>{{post.content}}</p>**

**</li></ul>**

**<p \*ngIf="isLoading">Loading...</p>**

**</div>**

**<div class="alert alert-danger" \*ngIf="error">**

**<h1>An Error Occured!</h1>**

**<p>{{error}}</p>**

**<button class="btn btn-danger" (click)="onHandleError()">Okay</button>**

**</div></div></div></div>**

**Lesson03 observing different types of responses**

**Notes: -**

**1-by default observables return response body, you can control to return the whole response as below**

**//on the service level**

**post(model: PostModel):Observable<any> {**

**return this.http**

**.post(**

**'https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json',**

**model,{observe:'response'});}**

**//on the component level**

**onCreatePost(postData: PostModel) {**

**this.sub = this.postService.post(postData).subscribe(**

**(res) => {console.log(res.body);},**

**(error) => {this.error = error.message;});}**

**2-you can return the response type of the event to manipulate the type of response**

**delete(){**

**return this.http.delete(**

**'https://enbehpro-default-rtdb.europe-west1.firebasedatabase.app/posts.json',{observe:'events'}).pipe(**

**tap(event => {**

**console.log(event);**

**if(event.type === HttpEventType.Sent){console.log(event);}**

**if(event.type === HttpEventType.Response){console.log(event.body);}}));}**

**3-the default response type is json , you can convert to text / blob on demand you want , but keep in minds that you have to handle it on the service response return and view**

**responseType:'json'**

**responseType:'text'**

**responseType:'blob'**

**Lesson04 Interceptor in Angular**

**Notes: -**

**1-we can with interceptor modify any request outgoing your app and append some header value and make business logic on center place**

**2-we can apply multi-interceptors but be aware that the order is important in executing**

**3-we can access to all properties of the request URL before send and after income from the Web API by using observable operators such as map , tab ,etc..**

**Example: -**

**We are using two interceptors and apply business logic and modify the http request as below**

**import { HttpEvent, HttpEventType, HttpHandler, HttpInterceptor, HttpRequest } from "@angular/common/http";**

**import { Observable } from "rxjs";**

**import { tap } from "rxjs/operators";**

**export class AuthInterceptorService implements HttpInterceptor{**

**//next its the code run after create http request and before leave your app**

**//(we put authorization data)**

**intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {**

**//we can check the request method type as below**

**if(req.method == 'GET'){console.log('Request is on its way');}**

**//we can modify the request by adding**

**const modifiedRequest = req.clone({headers: req.headers.append('fullname', 'm93enbah'),});**

**//it will return to request and then request go to the API**

**//we can handle the response return from the API we can other operators**

**//such as using of tap , map , etc..**

**return next.handle(modifiedRequest).pipe(tap(event => {**

**if(event.type == HttpEventType.Response){**

**console.log('Response arribed , body data : '+event.body);}}));}}**

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

**import {HttpRequest,HttpHandler,HttpEvent,HttpInterceptor,HttpEventType,}**

**from '@angular/common/http';**

**import { Observable } from 'rxjs';**

**import { tap } from 'rxjs/operators';**

**@Injectable()**

**export class LoggingInterceptor implements HttpInterceptor {**

**constructor() {}**

**intercept(**

**request: HttpRequest<unknown>,**

**next: HttpHandler**

**): Observable<HttpEvent<unknown>> {**

**console.log('Outgoing request');**

**console.log(request.url);**

**return next.handle(request).pipe(**

**tap((event) => {**

**if (event.type === HttpEventType.Response) {**

**console.log('Incoming Response');**

**console.log(event.body);}}));}}**

**we can register multi-interceptors as below**

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

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

**import { FormsModule } from '@angular/forms';**

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

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

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

**import { AuthInterceptorService } from './interceptors/auth-interceptor-service';**

**import { LoggingInterceptor } from './interceptors/logging.interceptor';**

**import { PostService } from './services/post.service';**

**@NgModule({**

**declarations: [AppComponent],**

**imports: [BrowserModule, AppRoutingModule, FormsModule, HttpClientModule],**

**//to inject the authorization interceptor service among all the services**

**//we can apply multi interceptor and the order is important**

**providers: [**

**PostService,{**

**provide: HTTP\_INTERCEPTORS,**

**useClass: AuthInterceptorService,**

**multi: true,},**

**{provide: HTTP\_INTERCEPTORS,**

**useClass: LoggingInterceptor,**

**multi: true,}],**

**bootstrap: [AppComponent],})**

**export class AppModule {}**