**Back-end set up**

Use fire base, we send the request to firebase service and then they will access automatically to database and your data will be get stored there or fetched from here

**Sending Request**

Create a service to centralize tasks

storeServers(servers: any[]) {

// in there I want to reach out where I want to send such a request

// and reach out to my server to store my service there

this.http.post('https://udemy-ng-http-d2e84.firebaseio.com/',servers);

// but actual angular is using observables behind the scene

}

We have a method with name storeServers to call http request. Angular actually is using observables behind the scene that a reason it doesn’t call the method as above. The reason for this is since we use an observable to also then subscribe to it and react to any response

return this.http.post('https://udemy-ng-http-d2e84.firebaseio.com/data.json',servers);

have data.json otherwise you will get a error

onSave() {

this.serverService.storeServers(this.servers).subscribe(

// when receive some data

(response) => {console.log(response)},

(error) => console.log(error)

);

}

**Ajusting request header**

Back end need to send some specific headers with the request. Here we didn’t configure any specific header

return this.http.post('https://udemy-ng-http-d2e84.firebaseio.com/data.json',servers);

storeServers(servers: any[]) {

// in there I want to reach out where I want to send such a request

// and reach out to my server to store my service there

const headers = new Headers({'Content-Type': 'application/json'});

return this.http.post('https://udemy-ng-http-d2e84.firebaseio.com/data.json',

servers, {headers: headers});

// but actual angular is using observables behind the scene

}

Header is object and property headers in http request is only optional

**Sending GET request**

getServers() {

return this.http.get('https://udemy-ng-http-d2e84.firebaseio.com/data.json');

}

<button class="btn btn-primary" (click)="onGet()">Get Servers</button>

onGet() {

this.serverService.getServers().subscribe(

(response: any) => {

const data = response.json();

for(let responseItem in data) {

this.servers.push(...data[responseItem]);

}

},

(error) => console.log(error)

)

}

We use response.json to parse Response object to Json

**Sending put request**

updateServers(servers: any[]) {

const headers = new Headers({'Content-Type': 'application/json'});

return this.http.put('https://udemy-ng-http-d2e84.firebaseio.com/data.json',

servers, {headers: headers});

}

I will not append our service to object with unique keys instead of it will override existing data. These unique IDs are gone and we just have 0 and 1 our services

**Transform Responses Easily with Observable Operation**

onGet() {

this.serverService.getServers().subscribe(

(response: any) => {

const data = response.json();

for(let responseItem in data) {

this.servers.push(...data[responseItem]);

}

},

(error) => console.log(error)

)

}

In above code you can see we transform data from Response Object to json at here, but we can central transform it in service request by using observable operation.

**Using Returned Data**

getServers() {

return this.http.get('https://udemy-ng-http-d2e84.firebaseio.com/data.json')

.map(

(response) => {

const data = response.json();

for(let item of data) {

item.name = 'FETCHED\_' + item.name;

}

return data;

}

)

;

}

Use the response data from server and then append prefix “FETCHED\_” before the item name

**Catching Http error**

getServers() {

return this.http.get('https://udemy-ng-http-d2e84.firebaseio.com/data')

.map(

(response) => {

const data = response.json();

for(let item of data) {

item.name = 'FETCHED\_' + item.name;

}

return data;

}

)

.catch(

(error: Response) => {

return Observable.throw("SOME THING WRONG!!!");

}

)

;

}

**Using the “async” Pipe with Http Requests**

It help us transform data which we get asynchronously and output it in the template

getAppName() {

return this.http.get('https://udemy-ng-http-d2e84.firebaseio.com/data/Test2.json')

.map(

(response) => {

return response.json();

}

)

}

appName;

ngOnInit() {

this.appName = this.serverService.getAppName();

}

If only as above code => appName is [object object] because getAppName() method will send a request to server and response value , but at this time don’t observable => above code finished . If app Name have any update from response then angular don’t know that, because nothing subscribe the changing. We have two way to do that:

this.serverService.getAppName().subscribe(

(name: any) => {

this.appName = name;

}

);

Or

<h1>{{ appName | async }}</h1>

Using async pipe on UI