**Section18 Making Http Requests**

**Section 18: Lecture 235//MUST READ: Angular 6 and Http**

Angular 6 is currently the latest version of Angular and it deprecates the Http-access method taught in this module.

What does this mean?

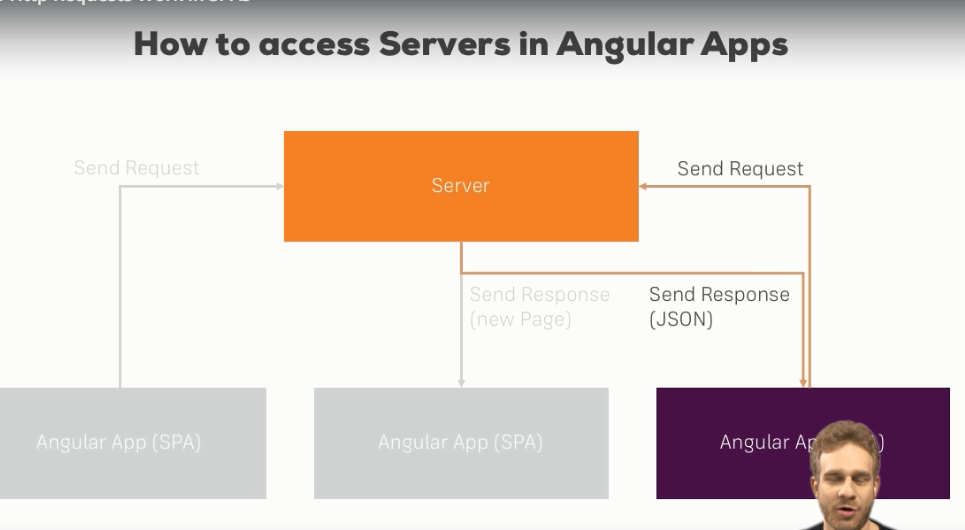
It means that the method still works, still is secure - you can use it! But there is a better Http module to use now: HttpClient.

I added a module (section 23) on that new client months ago, even before Angular 5 was released. You'll meet it later in the course and we'll easily update all our Http calls with the new client there.

So for now, follow along with this module here - the core concepts taught here will still apply (i.e. how it works etc).

And later in the course, we'll revisit this solution and update it to HttpClient.

**Section 18: Lecture 236//Introduction and how http Requests work in SPAs**

1. Here we will learn – how to make HTTP requests in an angular app. As we do need to reach out server sometimes i.e. to store something in the database.
2. Angular itself cannot access the database. So we need to reach out some server which does this for us and it gives use back the data if we need it. So, connecting to servers is important.
3. If we send request to the server, we are not getting back a new page. The request in angular is sent via Ajax.
4. 

**Section 18: Lecture 237//Example App and Backend Setup**

1. In this module we will use firebase a backed by google. <https://firebase.google.com>. This will provide us a ready to use backend.
2. Here we will store the servers in the backend and fetch them from there.

**Section 18: Lecture 238//Sending Requests (Example: POST Request)**

1. Not technically required but we will create our separate service for this i.e. POST requests.
2. @Injectable decorator is required if we want to inject a service into a service.
3. We will inject the angular built in HTTP service, which gives us some methods which we need for sending requests. Now, with this we inject the angular HTTP service into a constructor.
4. Here when we call the post method on http service – angular is using observables behind the scene. But if we don’t subscribe the observable – no request will get sent.
5. Here, we will return the observable that we are creating in the service. We will subscribe it in the component.
6. servers.service.ts:
7. import { Injectable } from "@angular/core";
8. import { Http } from "@angular/http";
9. @Injectable()
10. export class ServerService {
11. constructor(private http: Http){
13. }
14. storeServers(servers: any[]){
15. return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json', servers);
16. }
17. }

7. app.module.ts:

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

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

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

import { HttpModule } from '@angular/http';

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

import { ServerService } from './servers.service';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule,

HttpModule

],

providers: [ServerService],

bootstrap: [AppComponent]

})

export class AppModule { }

8. app.component.ts:

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

import { ServerService } from './servers.service';

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

servers = [

{

name: 'Testserver',

capacity: 10,

id: this.generateId()

},

{

name: 'Liveserver',

capacity: 100,

id: this.generateId()

}

];

constructor(private serverService: ServerService){}

onAddServer(name: string) {

this.servers.push({

name: name,

capacity: 50,

id: this.generateId()

});

}

onSave(){

this.serverService.storeServers(this.servers)

.subscribe(

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

(error)=>console.log(error)

);

}

private generateId() {

return Math.round(Math.random() \* 10000);

}

}

**Section 18: Lecture 239//Adjusting request headers**

1. In the last lecture we sent some data to our server as we implemented this save servers button and there we fired a method in serversService - which generates this observable for us which holds this pre-configured request and then we sent that request by subscribing to that observable.

1. So, here we will configure the headers – here we will not configure any special headers because the default ones are what we need.
2. Then we will pass these headers in the post request as the 3rd argument.
3. servers.service.ts:
4. import { Injectable } from "@angular/core";
5. import { Http, Headers } from "@angular/http";
6. @Injectable()
7. export class ServerService {
8. constructor(private http: Http){
10. }
11. storeServers(servers: any[]){
12. const headers = new Headers({'Content-Type':'application/json'});
13. return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
14. servers,
15. {headers: headers});
16. }
17. }

**Section 18: Lecture 240//Sending GET requests**

1. Now, we will add a new method named getService in our service will get the stored services from the server.
2. Here we are using response.json() method to extract the data from the response by our observable.
3. app.component.ts:
4. import { Component } from '@angular/core';
5. import { ServerService } from './servers.service';
6. import {Response} from '@angular/http';
7. @Component({
8. selector: 'app-root',
9. templateUrl: './app.component.html',
10. styleUrls: ['./app.component.css']
11. })
12. export class AppComponent {
13. servers = [
14. {
15. name: 'Testserver',
16. capacity: 10,
17. id: this.generateId()
18. },
19. {
20. name: 'Liveserver',
21. capacity: 100,
22. id: this.generateId()
23. }
24. ];
25. constructor(private serverService: ServerService){}
26. onAddServer(name: string) {
27. this.servers.push({
28. name: name,
29. capacity: 50,
30. id: this.generateId()
31. });
32. }
33. onSave(){
34. this.serverService.storeServers(this.servers)
35. .subscribe(
36. (response)=> console.log(response),
37. (error)=>console.log(error)
38. );
39. }
40. onGet(){
41. this.serverService.gerServers()
42. .subscribe(
43. //(response)=> console.log(response),
44. (response: Response)=>{
45. const data = response.json();
46. console.log(data);
47. },
48. (error)=>console.log(error)
49. );
50. }
51. private generateId() {
52. return Math.round(Math.random() \* 10000);
53. }
54. }
55. servers.service.ts:
56. import { Injectable } from "@angular/core";
57. import { Http, Headers } from "@angular/http";
58. @Injectable()
59. export class ServerService {
60. constructor(private http: Http){
62. }
63. storeServers(servers: any[]){
64. const headers = new Headers({'Content-Type':'application/json'});
65. return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
66. servers,
67. {headers: headers});
68. }
69. gerServers(){
70. return this.http.get('http://udemy-ng-http-81cda.firebaseapp.com/data.json');
71. }
72. }

**Section 18: Lecture 241//Sending a PUT request**

1. servers.service.ts:
2. import { Injectable } from "@angular/core";
3. import { Http, Headers } from "@angular/http";
4. @Injectable()
5. export class ServerService {
6. constructor(private http: Http){
8. }
9. storeServers(servers: any[]){
10. const headers = new Headers({'Content-Type':'application/json'});
11. // return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
12. // servers,
13. // {headers: headers});
14. return this.http.put('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
15. servers,
16. {headers: headers});
17. }
18. gerServers(){
19. return this.http.get('http://udemy-ng-http-81cda.firebaseapp.com/data.json');
20. }
21. }

**Section 18: Lecture 242//RxJS 6 without rxjs-compat**

Don't forget - if you're using Angular (and therefore also RxJS 6+) and you're NOT using rxjs-compat  (npm install --save rxjs-compat  - you may ignore this lecture then, use the code as shown in the videos!), you have to use operators like map()  differently:

Instead of

....map(...)

use

....pipe(map(...))

map also needs to be imported:

Instead of

import 'rxjs/Rx';

use

import { map } from 'rxjs/operators';

**Section 18: Lecture// Transform Responses Easily with Observable Operators(map())**

1. Here we will do data extraction for the get method in the servers.service.ts.
2. The map observable will get data from the form the old observable and wrap it to extract data and transform it - it will also warp this transformed data into another observable.
3. Here we will still get back an observable.
4. app.component.ts:
5. import { Component } from '@angular/core';
6. import { ServerService } from './servers.service';
7. import {Response} from '@angular/http';
8. @Component({
9. selector: 'app-root',
10. templateUrl: './app.component.html',
11. styleUrls: ['./app.component.css']
12. })
13. export class AppComponent {
14. servers = [
15. {
16. name: 'Testserver',
17. capacity: 10,
18. id: this.generateId()
19. },
20. {
21. name: 'Liveserver',
22. capacity: 100,
23. id: this.generateId()
24. }
25. ];
26. constructor(private serverService: ServerService){}
27. onAddServer(name: string) {
28. this.servers.push({
29. name: name,
30. capacity: 50,
31. id: this.generateId()
32. });
33. }
34. onSave(){
35. this.serverService.storeServers(this.servers)
36. .subscribe(
37. (response)=> console.log(response),
38. (error)=>console.log(error)
39. );
40. }
41. onGet(){
42. this.serverService.gerServers()
43. .subscribe(
44. //(response)=> console.log(response),
45. (servers: any[])=> console.log(servers),
46. // {
47. // const data = response.json();
48. // console.log(data);
49. // },
50. (error)=>console.log(error)
51. );
52. }
53. private generateId() {
54. return Math.round(Math.random() \* 10000);
55. }
56. }

5. servers.service.ts:

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

import { Http, Headers, Response } from "@angular/http";

import 'rxjs/Rx';

@Injectable()

export class ServerService {

constructor(private http: Http){

}

storeServers(servers: any[]){

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

// return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',

// servers,

// {headers: headers});

return this.http.put('http://udemy-ng-http-81cda.firebaseapp.com/data.json',

servers,

{headers: headers});

}

gerServers(){

return this.http.get('http://udemy-ng-http-81cda.firebaseapp.com/data.json')

.map(

(response: Response)=>{

const data = response.json();

return data;

}

);

}

}

**Section 18: Lecture 224//Using the Returned Data**

1. Here we will demonstrate how we will transform the data.
2. servers.service.ts:
3. import { Injectable } from "@angular/core";
4. import { Http, Headers, Response } from "@angular/http";
5. import 'rxjs/Rx';
6. @Injectable()
7. export class ServerService {
8. constructor(private http: Http){
10. }
11. storeServers(servers: any[]){
12. const headers = new Headers({'Content-Type':'application/json'});
13. // return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
14. // servers,
15. // {headers: headers});
16. return this.http.put('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
17. servers,
18. {headers: headers});
19. }
20. gerServers(){
21. return this.http.get('http://udemy-ng-http-81cda.firebaseapp.com/data.json')
22. .map(
23. (response: Response)=>{
24. const data = response.json();
25. for(const server of data){
26. server.name = 'FETCHED\_'+server.name;
27. }
28. return data;
29. }
30. );
31. }
32. }

**Section 18: Lecture 225//Catching errors without rxjs-compat**

Are you using Angular 6 (and therefore RxJS 6+) and you're NOT using rxjs-compat  (npm install --save rxjs-compat  - you may ignore this lecture then, use the code as shown in the videos!)?

You then have to use the catch()  operator you'll see in the next lecture a bit differently.

Instead of

1. ....catch(error => {
2. return Observable.throw(...)
3. })

use

1. ....pipe(catchError(error => {
2. return throwError(...)
3. }))

And make sure to import it:

Instead of

import 'rxjs/Rx';

and

import { Observable } from 'rxjs/Observable';

use

import { catchError } from 'rxjs/operators';

and

import { throwError } from 'rxjs';

**Section 18: Lecture 246//Catching Http Errors**

1. Now, we will add the catch operator.
2. services.service.ts:
3. import { Injectable } from "@angular/core";
4. import { Http, Headers, Response } from "@angular/http";
5. import 'rxjs/Rx';
6. import { Observable } from "rxjs/Observable";
7. @Injectable()
8. export class ServerService {
9. constructor(private http: Http){
11. }
12. storeServers(servers: any[]){
13. const headers = new Headers({'Content-Type':'application/json'});
14. // return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
15. // servers,
16. // {headers: headers});
17. return this.http.put('http://udemy-ng-http-81cda.firebaseapp.com/data.json',
18. servers,
19. {headers: headers});
20. }
21. gerServers(){
22. return this.http.get('http://udemy-ng-http-81cda.firebaseapp.com/data.json')
23. .map(
24. (response: Response)=>{
25. const data = response.json();
26. for(const server of data){
27. server.name = 'FETCHED\_'+server.name;
28. }
29. return data;
30. }
31. )
32. .catch(
33. (error: Response)=>{
34. console.log(error);
35. return Observable.throw('Something went wrong!');
36. }
37. );
38. }
39. }

**Section 18: Lecture 247//Using the “async” Pipe with Http Requests**

* 1. We will use something here that we used in the pipes module i.e. the Async Pipe.
  2. app.component.html:

1. <div class="container">
2. <div class="row">
3. <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
4. <h1>{{ appName | async }}</h1>
5. <input type="text" #serverName>
6. <button class="btn btn-primary" (click)="onAddServer(serverName.value)">Add Server</button>
7. <br><br>
8. <button class="btn btn-primary" (click)="onSave()">Save Servers</button>
9. <button class="btn btn-primary" (click)="onGet()">Get Server</button>
10. <hr>
11. <ul class="list-group" \*ngFor="let server of servers">
12. <li class="list-group-item">{{ server.name }} (ID: {{ server.id }})</li>
13. </ul>
14. </div>
15. </div>
16. </div>
17. App.component.ts:
18. import { Component } from '@angular/core';
19. import { ServerService } from './servers.service';
20. import {Response} from '@angular/http';
21. @Component({
22. selector: 'app-root',
23. templateUrl: './app.component.html',
24. styleUrls: ['./app.component.css']
25. })
26. export class AppComponent {
27. appName = this.serverService.getAppName();
28. servers = [
29. {
30. name: 'Testserver',
31. capacity: 10,
32. id: this.generateId()
33. },
34. {
35. name: 'Liveserver',
36. capacity: 100,
37. id: this.generateId()
38. }
39. ];
40. constructor(private serverService: ServerService){}
41. onAddServer(name: string) {
42. this.servers.push({
43. name: name,
44. capacity: 50,
45. id: this.generateId()
46. });
47. }
48. onSave(){
49. this.serverService.storeServers(this.servers)
50. .subscribe(
51. (response)=> console.log(response),
52. (error)=>console.log(error)
53. );
54. }
55. onGet(){
56. this.serverService.gerServers()
57. .subscribe(
58. //(response)=> console.log(response),
59. (servers: any[])=> console.log(servers),
60. // {
61. // const data = response.json();
62. // console.log(data);
63. // },
64. (error)=>console.log(error)
65. );
66. }
67. private generateId() {
68. return Math.round(Math.random() \* 10000);
69. }
70. }

5. servers.service.ts:

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

import { Http, Headers, Response } from "@angular/http";

import 'rxjs/Rx';

import { Observable } from "rxjs/Observable";

@Injectable()

export class ServerService {

constructor(private http: Http){

}

storeServers(servers: any[]){

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

// return this.http.post('http://udemy-ng-http-81cda.firebaseapp.com/data.json',

// servers,

// {headers: headers});

return this.http.put('https://udemy-ng-http-ba4db.firebaseio.com/data.json',

servers,

{headers: headers});

}

gerServers(){

return this.http.get('https://udemy-ng-http-ba4db.firebaseio.com/data.json')

.map(

(response: Response)=>{

const data = response.json();

for(const server of data){

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

}

return data;

}

)

.catch(

(error: Response)=>{

console.log(error);

return Observable.throw('Something went wrong!');

}

);

}

getAppName(){

return this.http.get('https://udemy-ng-http-ba4db.firebaseio.com/appName.json')

.map(

(response: Response)=>{

return response.json();

}

)

;

}

}

========================END OF MODULE==================================