**Section 23 Bonus The http client**

**Section 23: Lecture 293// Module Introduction**

1. The new http client is an alternative to the old client - behind the scenes it uses modern APIs.
2. The more important thing is it supports more features than the old client did most importantly - interstaters.

**Section 23: Lecture 294//The Documentation**

1. <https://angular.io> 🡪 Docs 🡪 Fundamentals 🡪 HttpClient

**Section 23: Lecture 295//Unlocking**

1. The new client gives us some new features like interceptors which can be nice.
2. app.module.ts:
3. import { BrowserModule } from '@angular/platform-browser';
4. import { NgModule } from '@angular/core';
5. import { HttpClientModule } from '@angular/common/http';
6. import { AppComponent } from './app.component';
7. import { AppRoutingModule } from './app-routing.module';
8. import { SharedModule } from './shared/shared.module';
9. import { ShoppingListModule } from './shopping-list/shopping-list.module';
10. import { AuthModule } from './auth/auth.module'
11. import { CoreModule } from './core/core.module';
12. @NgModule({
13. declarations: [
14. AppComponent
15. ],
16. imports: [
17. BrowserModule,
18. HttpClientModule,
19. AppRoutingModule,
20. SharedModule,
21. ShoppingListModule,
22. AuthModule,
23. CoreModule
24. ],
25. bootstrap: [AppComponent]
26. })
27. export class AppModule { }
28. data-storage.service.ts:
29. import { Injectable } from '@angular/core';
30. import 'rxjs/Rx';
31. import { RecipeService } from '../recipes/recipe.service';
32. import { Recipe } from '../recipes/recipe.model';
33. import { AuthService } from '../auth/auth.service';
34. import { HttpClient } from '@angular/common/http';
35. @Injectable()
36. export class DataStorageService {
37. constructor(private httpClient: HttpClient,
38. private recipeService: RecipeService,
39. private authService: AuthService) {
40. }
41. storeRecipes() {
42. const token = this.authService.getToken();
43. return this.httpClient.put('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token, this.recipeService.getRecipes());
44. }
45. getRecipes() {
46. const token = this.authService.getToken();
47. this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token)
48. .map(
49. (recipes) => {
50. //const recipes: Recipe[] = response.json();
51. for (let recipe of recipes) {
52. if (!recipe['ingredients']) {
53. recipe['ingredients'] = [];
54. }
55. }
56. return recipes;
57. }
58. )
59. .subscribe(
60. (recipes: Recipe[]) => {
61. this.recipeService.setRecipes(recipes);
62. }
63. );
64. }
65. }

**Section 23: Lecture 296//Request Configuration and Response**

1. data-storage.service.ts:
2. import { Injectable } from '@angular/core';
3. import 'rxjs/Rx';
4. import { RecipeService } from '../recipes/recipe.service';
5. import { Recipe } from '../recipes/recipe.model';
6. import { AuthService } from '../auth/auth.service';
7. import { HttpClient } from '@angular/common/http';
8. @Injectable()
9. export class DataStorageService {
10. constructor(private httpClient: HttpClient,
11. private recipeService: RecipeService,
12. private authService: AuthService) {
13. }
14. storeRecipes() {
15. const token = this.authService.getToken();
16. return this.httpClient.put('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token, this.recipeService.getRecipes());
17. }
18. getRecipes() {
19. const token = this.authService.getToken();
20. // this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token)
21. this.httpClient.get('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token,
22. {
23. observe: 'response',
24. responseType: 'text'
25. })
26. .map(
27. (recipes) => {
28. console.log(recipes);
29. //const recipes: Recipe[] = response.json();
30. // for (let recipe of recipes) {
31. // if (!recipe['ingredients']) {
32. // recipe['ingredients'] = [];
33. // }
34. // }
35. //return recipes;
36. return [];
37. }
38. )
39. .subscribe(
40. (recipes: Recipe[]) => {
41. this.recipeService.setRecipes(recipes);
42. }
43. );
44. }
45. }

**Section 23: Lecture 297//Requesting Events**

1. header.component.ts:
2. import { Component } from '@angular/core';
3. import { DataStorageService } from '../../shared/data-storage.service';
4. import { AuthService } from '../../auth/auth.service';
5. import { HttpEvent, HttpEventType } from '@angular/common/http';
6. @Component({
7. selector: 'app-header',
8. templateUrl: './header.component.html'
9. })
10. export class HeaderComponent {
11. constructor(private dataStorageService: DataStorageService,
12. private authService: AuthService) {
13. }
14. onSaveData() {
15. this.dataStorageService.storeRecipes()
16. .subscribe(
17. (response: HttpEvent<Object>) => {
18. console.log(response.type=== HttpEventType.Sent);
19. }
20. );
21. }
22. onFetchData() {
23. this.dataStorageService.getRecipes();
24. }
25. onLogout() {
26. this.authService.logout();
27. }
28. }
29. data-storage.service.ts:
30. import { Injectable } from '@angular/core';
31. import 'rxjs/Rx';
32. import { RecipeService } from '../recipes/recipe.service';
33. import { Recipe } from '../recipes/recipe.model';
34. import { AuthService } from '../auth/auth.service';
35. import { HttpClient } from '@angular/common/http';
36. @Injectable()
37. export class DataStorageService {
38. constructor(private httpClient: HttpClient,
39. private recipeService: RecipeService,
40. private authService: AuthService) {
41. }
42. storeRecipes() {
43. const token = this.authService.getToken();
44. return this.httpClient.put('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token, this.recipeService.getRecipes(), {
45. observe:'events'
46. });
47. }
48. getRecipes() {
49. const token = this.authService.getToken();
50. // this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token)
51. this.httpClient.get('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token,
52. {
53. observe: 'response',
54. responseType: 'text'
55. })
56. .map(
57. (recipes) => {
58. console.log(recipes);
59. //const recipes: Recipe[] = response.json();
60. // for (let recipe of recipes) {
61. // if (!recipe['ingredients']) {
62. // recipe['ingredients'] = [];
63. // }
64. // }
65. //return recipes;
66. return [];
67. }
68. )
69. .subscribe(
70. (recipes: Recipe[]) => {
71. this.recipeService.setRecipes(recipes);
72. }
73. );
74. }
75. }

**Section 23: Lecture 298//Settings Headers**

1. In last lecture we saw that we can listen to the events and not only the response. By default we listen to the response and not event.
2. data-storage.service.ts:
3. import { Injectable } from '@angular/core';
4. import 'rxjs/Rx';
5. import { RecipeService } from '../recipes/recipe.service';
6. import { Recipe } from '../recipes/recipe.model';
7. import { AuthService } from '../auth/auth.service';
8. import { HttpClient, HttpHeaders } from '@angular/common/http';
9. @Injectable()
10. export class DataStorageService {
11. constructor(private httpClient: HttpClient,
12. private recipeService: RecipeService,
13. private authService: AuthService) {
14. }
15. storeRecipes() {
16. const token = this.authService.getToken();
17. const headers = new HttpHeaders().set('Authorization', 'Bearer djsjffajsdfhsj');
18. return this.httpClient.put('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token, this.recipeService.getRecipes(), {
19. observe:'body',
20. headers: headers
21. });
22. }
23. getRecipes() {
24. const token = this.authService.getToken();
25. // this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token)
26. this.httpClient.get('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token,
27. {
28. observe: 'response',
29. responseType: 'text'
30. })
31. .map(
32. (recipes) => {
33. console.log(recipes);
34. //const recipes: Recipe[] = response.json();
35. // for (let recipe of recipes) {
36. // if (!recipe['ingredients']) {
37. // recipe['ingredients'] = [];
38. // }
39. // }
40. //return recipes;
41. return [];
42. }
43. )
44. .subscribe(
45. (recipes: Recipe[]) => {
46. this.recipeService.setRecipes(recipes);
47. }
48. );
49. }
50. }

**Section 23: Lecture 299//Http parameters**

1. data-storage.service.ts:
2. import { Injectable } from '@angular/core';
3. import 'rxjs/Rx';
4. import { RecipeService } from '../recipes/recipe.service';
5. import { Recipe } from '../recipes/recipe.model';
6. import { AuthService } from '../auth/auth.service';
7. import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';
8. @Injectable()
9. export class DataStorageService {
10. constructor(private httpClient: HttpClient,
11. private recipeService: RecipeService,
12. private authService: AuthService) {
13. }
14. storeRecipes() {
15. const token = this.authService.getToken();
16. //const headers = new HttpHeaders().set('Authorization', 'Bearer djsjffajsdfhsj');
17. return this.httpClient.put('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token, this.recipeService.getRecipes(), {
18. observe:'body',
19. params: new HttpParams().set('auth',token)
20. //headers: headers
21. });
22. }
23. getRecipes() {
24. const token = this.authService.getToken();
25. // this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token)
26. this.httpClient.get('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json',
27. {
28. observe: 'response',
29. responseType: 'text'
30. })
31. .map(
32. (recipes) => {
33. console.log(recipes);
34. //const recipes: Recipe[] = response.json();
35. // for (let recipe of recipes) {
36. // if (!recipe['ingredients']) {
37. // recipe['ingredients'] = [];
38. // }
39. // }
40. //return recipes;
41. return [];
42. }
43. )
44. .subscribe(
45. (recipes: Recipe[]) => {
46. this.recipeService.setRecipes(recipes);
47. }
48. );
49. }
50. }

**Section 23: Lecture 300//Progress**

1. data-storage.service.ts:
2. import { Injectable } from '@angular/core';
3. import 'rxjs/Rx';
4. import { RecipeService } from '../recipes/recipe.service';
5. import { Recipe } from '../recipes/recipe.model';
6. import { AuthService } from '../auth/auth.service';
7. import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';
8. import { HttpRequest } from 'selenium-webdriver/http';
9. @Injectable()
10. export class DataStorageService {
11. constructor(private httpClient: HttpClient,
12. private recipeService: RecipeService,
13. private authService: AuthService) {
14. }
15. storeRecipes() {
16. const token = this.authService.getToken();
17. //const headers = new HttpHeaders().set('Authorization', 'Bearer djsjffajsdfhsj');
18. // return this.httpClient.put('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token, this.recipeService.getRecipes(), {
19. // observe:'body',
20. // params: new HttpParams().set('auth',token)
21. // //headers: headers
22. // });
23. const req = new HttpRequest('PUT','https://ng-recipe-book-3adbb.firebaseio.com/recipes.json',
24. this.recipeService.getRecipes(),{
25. reportProgress: true,
26. params: new HttpParams().set('auth', token)
27. })
28. return this.httpClient.request(req);
29. }
30. getRecipes() {
31. const token = this.authService.getToken();
32. // this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json?auth=' + token)
33. this.httpClient.get('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json',
34. {
35. observe: 'response',
36. responseType: 'text'
37. })
38. .map(
39. (recipes) => {
40. console.log(recipes);
41. //const recipes: Recipe[] = response.json();
42. // for (let recipe of recipes) {
43. // if (!recipe['ingredients']) {
44. // recipe['ingredients'] = [];
45. // }
46. // }
47. //return recipes;
48. return [];
49. }
50. )
51. .subscribe(
52. (recipes: Recipe[]) => {
53. this.recipeService.setRecipes(recipes);
54. }
55. );
56. }
57. }

**Section 23: Lecture 301//Interceptors**

1. Interceptors are used to do something on every request.