**Angular 7 By Example: HTTP GET Requests with HttpClient (Services, Async Pipe and Observables)**

How to send GET requests to REST API servers in your Angular 7 application using HttpClient. We'll also see how to use Angular services, RxJS Observables, models and the async pipe.

Front end applications, built using frameworks like Angular communicate with backend servers through REST APIs (which are based on the HTTP protocol) using either the XMLHttpRequest interface or the fetch() API.

Angular HttpClient makes use of the XMLHttpRequest interface which also supports old browsers.

Throughout this tutorial, we are going to see practical examples of how to use HttpClient, which is available from the @angular/common/http package, to make HTTP GET requests using the get()method.

* How to generate a fake and complete working REST API,
* How to create Angular services,
* How to subscribe to Observables,
* How to use the async pipe in templates to iterate over Observable data.

**Prerequisites**

Before getting started you need a few requirements. You need to have the following tools installed on your development machine:

* Node.js and npm. You can install both from the [official website](https://www.nodejs.org/).
* Angular CLI 7 (You can install it from npm using: npm install -g @angular/cli)

You also need to create an Angular 7 project.

In case, this is your first time using the Angular CLI, simply open your terminal and run the following command to generate a project:

$ ng new AngularHttpClientGetDemo

*Please note that we are using HttpClient which is an improved version of HTTP Client API and available starting with Angular version*[*4.3.0-rc.0*](https://github.com/angular/angular/blob/master/CHANGELOG.md#430-rc0-2017-07-08)

**Setting up HttpClient**

The first step is to include HttpClientModule in your main application module. Open the src/app/app.module.ts file and update it accordingly:

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

HttpClientModule

],

bootstrap: [AppComponent]

})

export class AppModule { }

That's all, we are now ready to use the HttpClient in our project.

**Setting up a Fake REST API**

To work with HttpClient we need a REST API server, you can either use an external API service, create a real Rest API server or create a fake API using json-server. In this example we'll use the last approach because it's less time consuming.

So head over to your terminal and start by installing json-server from npm:

$ npm install -g json-server

Next define your data in a db.json file:

{

"products": [

{

"id": 1,

"name": "Product001",

"cost": 10.0,

"quantity": 1000,

"locationId" : 1,

"familyId" : 1

},

{

"id": 2,

"name": "Product002",

"cost": 20.0,

"quantity": 2000,

"locationId" : 1,

"familyId" : 2

},

{

"id": 3,

"name": "Product003",

"cost": 30.0,

"quantity": 3000,

"locationId" : 3,

"familyId" : 2

},

{

"id": 4,

"name": "Product004",

"cost": 40.0,

"quantity": 4000,

"locationId" : 2,

"familyId" : 3

}

],

"locations":[

{

"id": 1,

"name": "Location001"

},

{

"id": 2,

"name": "Location002"

},

{

"id": 3,

"name": "Location003"

}

],

"families":[

{

"id": 1,

"name": "FM001"

},

{

"id": 2,

"name": "FM002"

},

{

"id": 3,

"name": "FM003"

}

],

"transactions":[

{

"id": 1,

"cost":11,

"quantity":10,

"productId":1

},

{

"id": 2,

"cost":12,

"quantity":100,

"productId":2

},

{

"id": 3,

"cost":15,

"quantity":101,

"productId":3

}

]

}

Next, you can run a REST server using the following command:

$ json-server --watch db.json

**The HttpClient get() Method**

The HttpClient get() method is designed to send HTTP GET requests. The syntax is as follows:

get(url: string, options: {

headers?: HttpHeaders;

observe: 'response';

params?: HttpParams;

reportProgress?: boolean;

responseType?: 'json';

withCredentials?: boolean;

}): Observable<HttpResponse<Object>>;

It takes a REST API endpoint and an optional options object and returns an Observable instance.

Now let's take a real world example. Let's presume you want to access the API endpoints we created above:

First you need to import HttpClient in your component.

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

Next you need to inject HttpClient via the component's constructor

constructor(private httpClient: HttpClient){}

Next, add a method where you can call HttpClient.get(ENDPOINT\_URL):

get\_products(){

this.httpClient.get(this.baseUrl + '/products').subscribe((res)=>{

console.log(res);

});

}

When called, this method will make a GET request to the /products endpoint then subscribe to the returned Observable. It will then log the array of products to the console.

Now let's make a button to callthe get\_products() method:

<button (click)="get\_products()">GET /products</button>

Now, If you want to show the products on the component template.

First, add a products array:

private products = [];

Next change the get\_products() method as follows:

get\_products(){

this.httpClient.get(this.baseUrl + '/products').subscribe((res : any[])=>{

console.log(res);

this.products = res;

});

}

We simply assing the returned products to the products array.

Next, use the ngFor directive in your component template to loop through the products array:

<ul>

<li \*ngFor="let product of products" >

-- id: {{product.id}}

-- name: {{product.name}}

-- cost: {{product.cost}}

-- quantity: {{product.quantity}}

</li>

</ul>

**The async pipe and Observables**

In our example, We can access the data returned by the get() method in two ways.

Subscribe to the returned Observable, i.e:

get\_products(){

this.httpClient.get(this.baseUrl + '/products').subscribe((res : any[])=>{

console.log(res);

this.products = res;

});

}

Or use the async pipe with the returned Observable and iterate directly over data in the template. Let's see how in more details.

First, you need to create an Observable using the following:

private productsObservable : Observable<any[]> ;

Next, call the get() method and assign the result to productsObservable:

this.productsObservable = this.httpClient.get(this.baseUrl + '/products');

Finally, in your template:

<li \*ngFor="let product of productsObservable | async" >

-- id: {{product.id}}

-- name: {{product.name}}

-- cost: {{product.cost}}

-- quantity: {{product.quantity}}

</li>

**Using Angular Services**

Using code that access data directly in your components is against the separation of concerns rule so let's refactor our code to use an Angular service which makes HTTP GET requests then returns the result back to our component(s).

Using Angular CLI generate a new service:

$ ng generate service data

Next move the data access code to this service. Open the src/app/data.service.ts file and update it accordingly:

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

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

@Injectable({

providedIn: 'root'

})

export class DataService {

baseUrl:string = "[http://localhost:3000](http://localhost:3000/)";

constructor(private httpClient : HttpClient) {}

get\_products(){

return this.httpClient.get(this.baseUrl + '/products');

}

get\_families(){

return this.httpClient.get(this.baseUrl + '/families');

}

get\_locations(){

return this.httpClient.get(this.baseUrl + '/locations');

}

get\_transactions(){

return this.httpClient.get(this.baseUrl + '/families');

}

}

Next, change the src/app/app.component.ts file as follows:

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

import { Observable } from 'rxjs';

import { DataService } from './data.service';

/\* .... \*/

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

private products = [];

private families = [];

private locations = [];

private transactions = [];

private productsObservable : Observable<any[]> ;

constructor(private dataService: DataService){

this.productsObservable = this.dataService.get\_products();

this.dataService.get\_families().subscribe((res : any[])=>{

this.families = res;

});

this.dataService.get\_locations().subscribe((res : any[])=>{

console.log(res);

this.locations = res;

});

this.dataService.get\_transactions().subscribe((res : any[])=>{

console.log(res);

this.transactions = res;

});

}

}

Instead of injecting HttpClient directly in our component we inject our data service and call its methods to make GET requests to our REST API server.

**Creating Models**

Now let's further refactor our code to use models for products, families, locations and transactions.

In the root of Angular project, create these models:

src/app/product.ts

export interface Product {

id: number;

name: string;

cost: number;

quantity: number;

locationId: number;

familyId: number;

}

src/app/family.ts

export interface Family {

id: number;

name: string;

}

src/app/location.ts

export interface Location {

id: number;

name: string;

constructor() { }

}

src/app/transaction.ts

export interface Transaction {

id: number;

cost: number;

productId: number;

quantity: number;

}

Next update your the src/app/app.component.ts file to use the new models:

import { Product } from './product';

import { Family } from './family';

import { Location } from './location';

import { Transaction } from './transaction';

private products : Product[] = [];

private families : Family[] = [];

private locations : Location[] = [];

private transactions : Transaction[] = [];

private productsObservable : Observable<Product[]> ;

constructor(private dataService: DataService){

this.productsObservable = this.dataService.get\_products();

this.dataService.get\_families().subscribe((res : Family[])=>{

this.families = res;

});

this.dataService.get\_locations().subscribe((res : Location[])=>{

this.locations = res;

});

this.dataService.get\_transactions().subscribe((res : Transaction[])=>{

this.transactions = res;

});

}

DEMO

Step 1 :

The first step is to include HttpClientModule in your main application module. Open the src/app/app.module.ts file and update it accordingly:

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

HttpClientModule

],

bootstrap: [AppComponent]

})

export class AppModule { }

That's all, we are now ready to use the HttpClient in our project.

## Setting up a Fake REST API

To work with HttpClient we need a REST API server, you can either use an external API service, create a real Rest API server or create a fake API using json-server. In this example we'll use the last approach because it's less time consuming.

So head over to your terminal and start by installing json-server from npm:

$ npm install -g json-server

Next define your data in a db.json file:

{

"products": [

{

"id": 1,

"name": "Product001",

"quantity": 1000,

"locationId" : 1,

"familyId" : 1

},

{

"id": 2,

"name": "Product002",

"cost": 20.0,

"quantity": 2000,

"locationId" : 1,

"familyId" : 2

},

{

"id": 3,

"name": "Product003",

"cost": 30.0,

"quantity": 3000,

"locationId" : 3,

"familyId" : 2

},

{

"id": 4,

"name": "Product004",

"cost": 40.0,

"quantity": 4000,

"locationId" : 2,

"familyId" : 3

}

],

"locations":[

{

"id": 1,

"name": "Location001"

},

{

"id": 2,

"name": "Location002"

},

{

"id": 3,

"name": "Location003"

}

],

"families":[

{

"id": 1,

"name": "FM001"

},

{

"id": 2,

"name": "FM002"

},

{

"id": 3,

"name": "FM003"

}

],

"transactions":[

{

"id": 1,

"cost":11,

"quantity":10,

"productId":1

},

{

"id": 2,

"cost":12,

"quantity":100,

"productId":2

},

{

"id": 3,

"cost":15,

"quantity":101,

"productId":3

}

]

}

Next, you can run a REST server using the following command:

$ json-server --watch db.json

you need to import HttpClient in your component.

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

Next you need to inject HttpClient via the component's constructor

constructor(private httpClient: HttpClient){}

Next, add a method where you can call HttpClient.get(ENDPOINT\_URL):

get\_products(){

this.httpClient.get(this.baseUrl + '/products').subscribe((res)=>{

console.log(res);

});

<button (click)="get\_products()">GET /products</button>

To Get All Records in an Array

private products = [];

baseUrl : string ="http://localhost:3000";

get\_products()

{

this.httpClient.get(this.baseUrl+ '/products' ).subscribe((res :any[])=> {

console.log(res);

this.products = res;

});

To Display all Records

<p>api-list-comp works!</p>

<button (click) = "get\_products()"> Get All Products </button>

<ul>

<li \*ngFor="let product of products" >

-- id: {{product.id}}

-- name: {{product.name}}

-- cost: {{product.cost}}

-- quantity: {{product.quantity}}

</li>

</ul>

<https://www.techiediaries.com/angular-by-example-httpclient-get/>