<https://alligator.io/angular/httpclient-intro/>

Basic Usage

Making basic GET, POST, PUT, PATCH or DELETE requests is very similar to what you’re used to with the old Http API. One major difference is that a JSON response is expected by default, so there’s no need to explicitly parse the JSON response anymore.

Here’s a sample GET request:

// ...

constructor(private http: HttpClient) {}

getData() {

this.http.get(this.url).subscribe(res => {

this.posts = res;

});

}

If you expect something else than JSON as the response, you can specify the expected response type using an object with the responseType key:

getData() {

this.http.get(this.url, { responseType: 'text' }).subscribe(res => {

this.data = res;

});

}

You can also define an interface for the shape of the response and type-check against that interface:

interface Post {

title: string;

body: string;

};

// ...

constructor(private http: HttpClient) {}

getData() {

this.http.get<Post>(this.url).subscribe(res => {

this.postTitle = res.title;

});

}

By default the HttpClient returns the body of the response. You can pass-in an object with an observe key set to a value of ‘response’ to get the full response. This can be useful to inspect for certain headers:

getData() {

this.http.get<Post>(this.url, { observe: 'response' }).subscribe(res => {

this.powered = res.headers.get('X-Powered-By');

this.postTitle = res.body.title;

});

}

Post, put and patch requests

Making a POST, PUT or PATCH request is just as easy:

postData() {

this.http.post(this.url, this.payload).subscribe();

}

Notice how we still have to subscribe in order for the request to be made. Without the subscribe call, the request is cold. You’ll obviously probably want to handle any response back or error:

postData() {

this.http.post(this.url, this.payload).subscribe(

res => {

console.log(res);

},

(err: HttpErrorResponse) => {

console.log(err.error);

console.log(err.name);

console.log(err.message);

console.log(err.status);

}

);

}

A request error is of type HttpErrorResponse and contains, among others, an error name, error message and server status.

Options for passing-in headers or query parameters can also be added to a POST, PUT or PATCH request using the headers or params keys in the object passed-in as the 3rd argument:

updatePost() {

this.http

.put(this.url, this.payload, {

params: new HttpParams().set('id', '56784'),

headers: new HttpHeaders().set('Authorization', 'some-token')

})

.subscribe(...);

}

Notice here the use of the HttpParams and HttpHeaders classes. You’ll need to import these from @angular/common/http as well.

Progress Events

A great new feature with HttpClient is the ability to listen for progress events. This can be done on any type of request and different information will be available during the lifecycle of the request event. Here’s a full example with a GET request:

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

import {

HttpClient,

HttpRequest,

HttpEvent,

HttpEventType

} from '@angular/common/http';

@Injectable()

export class DataService {

url = '/some/api';

constructor(private http: HttpClient) {}

getData() {

const req = new HttpRequest('GET', this.url, {

reportProgress: true

});

this.http.request(req).subscribe((event: HttpEvent<any>) => {

switch (event.type) {

case HttpEventType.Sent:

console.log('Request sent!');

break;

case HttpEventType.ResponseHeader:

console.log('Response header received!');

break;

case HttpEventType.DownloadProgress:

const kbLoaded = Math.round(event.loaded / 1024);

console.log(`Download in progress! ${ kbLoaded }Kb loaded`);

break;

case HttpEventType.Response:

console.log('😺 Done!', event.body);

}

});

}

}

* We first need to build a request object by creating an instance of the HttpRequest class and using the reportProgress option.
* We then subscribe to our request object to initiate the request and listen to the different event types over the life of the request. We can react appropriately depending on the event type. The available event types are Sent, UploadProgress, ResponseHeader, DownloadProgress, Response and User.
* In the above example, we get the amount of data downloaded so far from the GET response, and in the case of something like a POST or PUT request we could also get the percentage of the payload uploaded by using something like 100 \* event.loaded / event.total. This makes it very easy to show a progress bar to the user.

🤓 This post covered the basics of HttpClient, and next will tackle the use of interceptors, the killer feature of HttpClient. You can also head over to the [official docs](https://angular.io/guide/http) to go more in-depth.

Top of Form

✨💌

Bottom of Form

GET

{

**"args"**: {},

**"headers"**: {

**"Accept"**: "\*/\*",

**"Accept-Encoding"**: "gzip, deflate, sdch",

**"Accept-Language"**: "en-US,en;q=0.8",

**"Cache-Control"**: "no-cache",

**"Host"**: "httpbin.org",

**"Origin"**: "<http://evil.com/>",

**"Pragma"**: "no-cache",

**"Referer"**: "<http://run.plnkr.co/ODORtE0cj708R2eO/>",

**"User-Agent"**: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.80 Safari/537.36"

},

**"origin"**: "2.139.81.90",

**"url"**: "<http://httpbin.org/get>"

}