Angular HTTPHeaders Example

1 Comment / 7 minutes of reading / September 1, 2021

← HttpParams/URL Params HTTP Interceptor →

In this guide let us explore how to add HTTP Headers to an HTTP request in Angular. There are two ways by which we can add the headers. One, we add the HTTP Headers while making a request. The second way is to use the HTTP interceptor to intercept all the Requests and add the Headers. In both cases, we use the httpHeaders configuration option provided by angular HttpClient to add the headers.

HTTP Headers let the client and the server share additional information about the HTTP request or response. For example, we use the content-type header to indicate the media type of the resource like JSON, text, blob, etc. Another important header is where you send the bearer token using the Authorization header

'Authorization', 'Bearer <yourTokenhere>'

Table of Contents

HttpHeaders

We add HTTP Headers using the HttpHeaders helper class. It is passed as one of the arguments to the GET, POST, PUT, DELETE, PATCH & OPTIONS request.

To use HttpHeaders in your app, you must import it into your component or service

```
1
2 import { HttpHeaders } from '@angular/common/http';
3
```

Then create an instance of the class

```
1
2 const headers= new HttpHeaders()
3   .set('content-type', 'application/json')
4   .set('Access-Control-Allow-Origin', '*');
5
```

And then call the httpClient.get method passing the headers as the argument

```
1
2 return this.httpClient.get(this.baseURL + 'users/' + userName + '/repos', { 'headers': hea
3
```

Note that `httpHeaders` are immutable. i.e every method on `HttpHeaders` object does not modify it but returns a new `HttpHeaders` object.

The `httpHeaders` class has several methods using which you can manipulate the headers.

set

```
set(name: string, value: string | string[]): HttpHeaders
```

The Sets method returns a new instance after modifying the given header. If the header already exists, its value is replaced with the given value in the returned object.

```
1
2 const headers = new HttpHeaders()
3    .set('content-type', 'application/json')
4    .set('Access-Control-Allow-Origin', '*');
5
```

httpHeaders are immutable

The HTTP headers are immutable. The following example does not work as each `set` method returns a new header and does not update the original header.

```

let headers = new HttpHeaders()
headers .set('content-type', 'application/json')
headers .set('Access-Control-Allow-Origin', '*')
console.log(headers);

```

To workaround, you can use the code as follows

```

const headers= new HttpHeaders()
  .set('content-type', 'application/json')
  .set('Access-Control-Allow-Origin', '*');

```

You can also use the following code

```

let headers = new HttpHeaders()
headers=headers.set('content-type','application/json')
headers=headers.set('Access-Control-Allow-Origin', '*');
console.log(headers)

```

append

append(name: string, value: string | string[]): HttpHeaders

The append method appends a new value to the existing set of values for a header and returns a new instance. The append method does not check if the value exists.

```

let headers = new HttpHeaders()

headers=headers.append('content-type','application/json')
headers=headers.append('Access-Control-Allow-Origin', '*')
headers=headers.append('content-type','application/x-www-form-urlencoded')

```

```
8 console.log(headers)
9
```

The above results in `content-type` header in the request header as

```
content-type: application/json,application/x-www-form-urlencoded
```

has

```
has(name: string): boolean
```

Returns true if the given header with the name already exists in the `HttpHeaders`. The following code checks if the `content-type` header present in the request header. If not it adds it.

```
1
2   let headers = new HttpHeaders()
3   headers=headers.append('Access-Control-Allow-Origin', '*')
4   if (!headers.has('content-type')) {
5     headers=headers.append('content-type','application/json')
6   }
7
```

get

```
get(name: string): string | null
```

Get the first value for the given header name, or null if it's not present.

```
let headers = new HttpHeaders()
   .set('content-type','application/json')
   .set('Access-Control-Allow-Origin', '*')

const h =headers.get('content-type')

if (h==null) {
  console.log('content type header not present')
} else {
  console.log(h)      //returns 'application/json'
}
```

getAll

getAll(name: string): string[] | null

Get all the headers for the given header name, or null if it's not present.

```
let headers = new HttpHeaders()
   .set('content-type','application/json')
   .set('Access-Control-Allow-Origin', '*')
   .append('content-type','application/x-www-form-urlencoded')

 const h =headers.getAll('content-type')
 console.log(h)

*** output
0: "application/json"
1: "application/x-www-form-urlencoded"
```

Keys

keys(): string[]

Get all the headers for this request.

```
let headers = new HttpHeaders()
    .set('content-type','application/json')
    .set('Access-Control-Allow-Origin', '*')
    .append('content-type','application/x-www-form-urlencoded')

const h =headers.keys()
console.log(h)

***output
0: "content-type"
1: "Access-Control-Allow-Origin"
```

delete

delete(name: string, value?: string | string[]): HttpHeaders

Deletes the header and returns the new headers. You can delete using the header name or by using the name & value.

```
let headers = new HttpHeaders()
    .set('content-type','application/json')
    .set('Access-Control-Allow-Origin', '*')
    .append('content-type','application/x-www-form-urlencoded')
```

```
headers=headers.delete("content-type","application/json")  //delete content-type='applicat

headers=headers.delete("content-type")   //delete all content-type headers

```

HttpHeaders from object

The following code shows how you can create `HttpHeaders` from an object.

```

let headers = new HttpHeaders({ 'Access-Control-Allow-Origin': '*','content-type': 'applicati
console.log(headers)

```

Using HTTP Interceptor

Most headers we add to the HTTP Request in the entire application are likely to remain the same. Adding them to every `GET`, `POST`, `PUT`, etc requests are cumbersome. Instead, you can make use of the HTTP Interceptors to intercept every request and add the commonly used headers. Refer to our tutorial on how to set HttpHeaders using HTTP Interceptors

HttpHeaders Example

Refer to our tutorial on HTTP Post example.

The code requires you to set up a fake backend server using `json-server`. Install JSON server using the following command.

```

npm install -g json-server

```

Create a db.json file with some data.

```
{
  "people": [
    {
      "id": 1,
      "name": "Don Bradman"
    },
    {
      "id": 2,
      "name": "Sachin Tendulkar"
    }
  ]
}
```

Start the server with the following command. The server will run on the port http://localhost:3000/

```
json-server --watch db.json
```

Code

person.ts

```
export class Person {
  id:number
  name:string
}
```

app.module.ts

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

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';




@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }


```

app.component.ts

```
import { Component, OnInit } from '@angular/core';
```

```
import { ApiService } from './api.service';
import { Person } from './person';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

  title = 'http Headers Example';
  people:Person[];
  person = new Person();

  constructor(private apiService:ApiService) {}

  ngOnInit() {
    this.refreshPeople()
  }

  refreshPeople() {
    this.apiService.getPeopleFromObject()
      .subscribe(data => {
        this.people=data;
      })

  }

  addPerson() {
    this.apiService.addPerson(this.person)
      .subscribe(data => {
          this.person = new Person();
        this.refreshPeople();
      })

  }

}


```

app.component.html

```

<h1>{{title}}</h1>

```

```
<div>
  <div>
    <label>Name: </label>
    <input [(ngModel)]="person.name" />
  </div>
  <div>
    <button (click)="addPerson()">Add</button>
  </div>
</div>

<button (click)="refreshPeople()">Refresh</button>

<table class='table'>
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let person of people;">
      <td>{{person.id}}</td>
      <td>{{person.name}}</td>
    </tr>
  </tbody>
</table>

```

app.routing.module.ts

```

import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';


const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }


```

app.service.ts

```
1
2  import { HttpClient, HttpHeaders } from '@angular/common/http';
3  import { Person } from './person';
4  import { Injectable } from '@angular/core';
5  import { Observable } from 'rxjs';
6
7  @Injectable({ providedIn: 'root' })
8  export class ApiService {
9
10   baseURL: string = "http://localhost:3000/";
11
12   constructor(private http: HttpClient) {
13   }
14
15
16   getPeople(): Observable<Person[]> {
17     console.log('getPeople ' + this.baseURL + 'people')
18     return this.http.get<Person[]>(this.baseURL + 'people')
19   }
20
21   //Adding headers
22   getPeopleWithHeaders(): Observable<Person[]> {
23     const headers = { 'content-type': 'application/json'}
24     console.log(headers)
25     return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
26   }
27
28   //Set method
29   getPeopleWithSet(): Observable<Person[]> {
30     const headers = new HttpHeaders()
31       .set('content-type','application/json')
32       .set('Access-Control-Allow-Origin', '*');
33     console.log(headers)
34     return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
35   }
36
37   //This wont work
38   getPeopleWithImmutable(): Observable<Person[]> {
39     const headers = new HttpHeaders()
40     headers.set('content-type','application/json')
41     headers.set('Access-Control-Allow-Origin', '*');
42
43     console.log(headers)
44     return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
45   }
```

```
46
47  getPeopleWithImmutable1(): Observable<Person[]> {
48    let headers = new HttpHeaders()
49    headers=headers.set('content-type','application/json')
50    headers=headers.set('Access-Control-Allow-Origin', '*');
51
52    console.log(headers)
53    return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
54  }
55
56
57  getPeopleAppend(): Observable<Person[]> {
58    let headers = new HttpHeaders()
59    headers=headers.append('content-type','application/json')
60    headers=headers.append('Access-Control-Allow-Origin', '*')
61    headers=headers.append('content-type','application/x-www-form-urlencoded')
62    headers=headers.append('customer-header', 'custom')
63    console.log(headers)
64    return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
65  }
66
67  getPeopleHas(): Observable<Person[]> {
68    let headers = new HttpHeaders()
69    //headers=headers.append('content-type','application/json')
70    headers=headers.append('Access-Control-Allow-Origin', '*')
71    if (!headers.has('content-type')) {
72      headers=headers.append('content-type','application/json')
73    }
74
75    console.log(headers)
76    return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
77  }
78
79
80  getPeopleGet(): Observable<Person[]> {
81    let headers = new HttpHeaders()
82      .set('content-type','application/json')
83      .set('Access-Control-Allow-Origin', '*')
84
85    const h =headers.get('content-type')
86    if (h==null) {
87      console.log('content type header not present')
88    } else {
89      console.log(h)
90    }
91
92
93    return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
94  }
```

```
95
96  getPeopleGetAll(): Observable<Person[]> {
97    let headers = new HttpHeaders()
98      .set('content-type','application/json')
99      .set('Access-Control-Allow-Origin', '*')
100     .append('content-type','application/x-www-form-urlencoded')
101
102   const h =headers.getAll('content-type')
103   console.log(h)
104
105
106   return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
107 }
108
109 getPeopleKeys(): Observable<Person[]> {
110   let headers = new HttpHeaders()
111     .set('content-type','application/json')
112     .set('Access-Control-Allow-Origin', '*')
113     .append('content-type','application/x-www-form-urlencoded')
114
115   const h =headers.keys()
116   console.log(h)
117
118
119   return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
120 }
121
122 getPeopleDelete(): Observable<Person[]> {
123   let headers = new HttpHeaders()
124     .set('content-type','application/json')
125     .set('Access-Control-Allow-Origin', '*')
126     .append('content-type','application/x-www-form-urlencoded')
127
128
129   headers=headers.delete('content-type','application/json')
130
131   //headers=headers.delete("content-type")
132
133
134   console.log(headers)
135
136
137   return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
138 }
139
140
141 getPeopleFromObject(): Observable<Person[]> {
142
143   let headers = new HttpHeaders({ 'Access-Control-Allow-Origin': '*','content-type': 'app
```

```
144
145     console.log(headers)
146
147
148     return this.http.get<Person[]>(this.baseURL + 'people',{'headers':headers})
149   }
150
151
152   addPerson(person:Person): Observable<Person> {
153     const headers = { 'content-type': 'application/json'}
154     const body=JSON.stringify(person);
155     console.log(body)
156     return this.http.post<Person>(this.baseURL + 'people', body,{'headers':headers})
157   }
158 }
159
```

Summary

We learned how to add/modify the HTTP Headers using the `HttpHeaders` in Angular.

← HttpParams/URL Params HTTP Interceptor →

Related Posts

Best Resources to Learn Angular

1 Comment / Angular / By TekTutorialsHub

Introduction to Angular | What is Angular?

1 Comment / Angular / By TekTutorialsHub