Difference between HTTP and HTTPClient in angular 4?

214

62

I want to know which one to use to build a mock web service to test the Angular program?

angular http httpclient

edited Jul 25 '17 at 6:03

asked Jul 16 '17 at 14:40

Aiyoub Amini
2,646 6 18 31

7 *"HttpClient is an evolution of the existing Angular HTTP API, which exists alongside of it in a separate package..."*. – jonrsharpe Jul 16 '17 at 14:47

1 I actually wrote about some of its new features on my blog yesterday: blog.jonrshar.pe/2017/Jul/15/angular-http-client.html – jonrsharpe Jul 16 '17 at 14:53

4 angular.io/guide/http – yurzui Jul 16 '17 at 14:55

6 The tutorial uses HttpModule and angular.io/guide/http uses HttpClientModule and neither explains when one or the other should be used or what version of Angular is needed to use what. – Mickey Segal Sep 19 '17 at 15:29

5 Answers

321

Use the `HttpClient` class from `HttpClientModule` if you're using Angular 4.3.x and above:

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

@NgModule({
 imports: [
   BrowserModule,
   HttpClientModule
```

It's an upgraded version of `http` from `@angular/http` module with the following improvements:

> - Interceptors allow middleware logic to be inserted into the pipeline
> - Immutable request/response objects
> - Progress events for both request upload and response download

You can read about how it works in [Insider's guide into interceptors and HttpClient mechanics in Angular](#).

> - Typed, synchronous response body access, including support for JSON body types
> - JSON is an assumed default and no longer needs to be explicitly parsed
> - Post-request verification & flush based testing framework

Going forward the old http client will be deprecated. Here are the links to the [commit message](#) and [the official docs](#).

Also pay attention that old http was injected using `Http` class token instead of the new `HttpClient` :

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

@NgModule({
 imports: [
   BrowserModule,
   HttpModule
 ],
 ...

 class MyService() {
    constructor(http: Http) {...}
```

Also, new `HttpClient` seem to require `tslib` in runtime, so you have to install it `npm i tslib` and update `system.config.js` if you're using `SystemJS` :

```
map: {
     ...
```

edited Jan 9 '18 at 16:02

answered Jul 16 '17 at 14:47

Max Koretskyi aka Wizard

54.8k 23 163 285

1 I am trying to import HttpClientModule. But '@angular/common/http' is not present in node_modules directory which I installed using "npm start" command. Can you help? – Dheeraj Kumar Jul 27 '17 at 7:27

1 @DheerajKumar, which version are you using? it's only available in 4.3.0 and up – Max Koretskyi aka Wizard Jul 27 '17 at 7:31

I downloaded angular quick start from git. and In package.json, "@angular/common": "^4.3.0" is present. but there is no @angular/common/http. – Dheeraj Kumar Jul 27 '17 at 7:35

remove `node_modules` folder and run `npm install` again – Max Koretskyi aka Wizard Jul 27 '17 at 7:44

5 I've run into this very same issue (I am using System.js). One thing that is missing from this answer is that you'll also need to map the new module in system.js as follows: `'@angular/common/http': 'npm:@angular/common/bundles/common-http.umd.js',` – Tyler O Aug 8 '17 at 18:02

40

Don't want to be repetitive, but just to summarize in other way:

- Automatic conversion from JSON to an object
- Response type definition
- Event firing
- Simplified syntax for headers
- Interceptors

I wrote an article, where I covered the difference between old "http" and new "HttpClient". The goal was to explain it in the easiest way possible.

Simply about new HttpClient in Angular

edited Sep 5 '18 at 14:21

answered Dec 5 '17 at 2:40

16

This is a good reference, it helped me switch my http requests to httpClient

https://blog.hackages.io/angular-http-httpclient-same-but-different-86a50bbcc450

It compares the two in terms of differences and gives code examples.

This is just a few differences I dealt with while changing services to httpclient in my project (borrowing from the article I mentioned) :

Importing

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

Requesting and parsing response:

@angular/http

```
this.http.get(url)
     // Extract the data in HTTP Response (parsing)
     .map((response: Response) => response.json() as GithubUser)
     .subscribe((data: GithubUser) => {
       // Display the result
       console.log('TJ user data', data);
     });
```

@angular/common/http

```
this.http.get(url)
     .subscribe((data: GithubUser) => {
       // Data extraction from the HTTP response is already done
       // Display the result
       console.log('TJ user data', data);
     });
```

Making the GET HTTP request with `responseType` option:

```
this.http.get(url, {responseType: 'blob'})
     .subscribe((data) => {
       // Data extraction from the HTTP response is already done
       // Display the result
       console.log('TJ user data', data);
     });
```

Adding Interceptor

I also used interceptors for adding the token for my authorization to every request:

This is a good reference: https://offering.solutions/blog/articles/2017/07/19/angular-2-new-http-interface-with-interceptors/

like so:

```
@Injectable()
export class MyFirstInterceptor implements HttpInterceptor {

    constructor(private currentUserService: CurrentUserService) { }

    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

        // get the token from a service
        const token: string = this.currentUserService.token;

        // add it if we have one
        if (token) {
            req = req.clone({ headers: req.headers.set('Authorization', 'Bearer ' +
token) });
        }

        // if this is a login-request the header is
        // already set to x/www/formurl/encoded.
        // so if we already have a content-type, do not
        // set it, but if we don't have one, set it to
        // default --> json
```

```
            req = req.clone({ headers: req.headers.set('Accept', 'application/json') });
            return next.handle(req);
        }
    }
```

Its a pretty nice upgrade!

edited Feb 15 at 18:40

Tarik
37.8k 68 209 310

answered May 3 '18 at 12:31

abann sunny
360 3 10

You need to include the relevant information in your answer and not just as a link – Michael May 3 '18 at 12:53

0

There is a library which allows you to **use HttpClient with strongly-typed callbacks**.

The data and the error are available directly via these callbacks.

When you use HttpClient with Observable, you have to use **.subscribe(x=>...)** in the rest of your code.

This is because **Observable< `HttpResponse` < `T` >>** is tied to **HttpResponse**.

This **tightly couples** the **http layer** with the **rest of your code**.

This library encapsulates the **.subscribe(x => ...)** part and exposes only the data and error through your Models.

With strongly-typed callbacks, you only have to deal with your Models in the rest of your code.

The library is called **angular-extended-http-client**.

Sample usage

The strongly-typed callbacks are

Success:

- **IObservable< `T` >**
- **IObservableHttpResponse**
- **IObservableHttpCustomResponse< `T` >**

Failure:

- **IObservableError< `TError` >**
- **IObservableHttpError**
- **IObservableHttpCustomError< `TError` >**

Add package to your project and in your app module

```
import { HttpClientExtModule } from 'angular-extended-http-client';
```

and in the @NgModule imports

```
  imports: [
    .
    .
    .
    HttpClientExtModule
  ],
```

Your Models

```
//Normal response returned by the API.
```

```
    className: string;
}
```

Your Service

In your Service, you just create params with these callback types.

Then, pass them on to the **HttpClientExt**'s get method.

```
import { Injectable, Inject } from '@angular/core'
import { RacingResponse, APIException } from '../models/models'
import { HttpClientExt, IObservable, IObservableError, ResponseType, ErrorType } from
'angular-extended-http-client';
.
.

@Injectable()
export class RacingService {

    //Inject HttpClientExt component.
    constructor(private client: HttpClientExt, @Inject(APP_CONFIG) private config:
AppConfig) {

    }

    //Declare params of type IObservable<T> and IObservableError<TError>.
    //These are the success and failure callbacks.
    //The success callback will return the response objects returned by the underlying
HttpClient call.
    //The failure callback will return the error objects returned by the underlying
HttpClient call.
    getRaceInfo(success: IObservable<RacingResponse>, failure?:
IObservableError<APIException>) {
        let url = this.config.apiEndpoint;

        this.client.get(url, ResponseType.IObservable, success,
ErrorType.IObservableError, failure);
    }
}
```

```
ngOnInit() {
  this.service.getRaceInfo(response => this.result = response.result,
                           error => this.errorMsg = error.className);

}
```

Both, **response** and **error** returned in the callbacks are strongly typed. Eg. **response** is type **RacingResponse** and **error** is **APIException**.

You only deal with your Models in these strongly-typed callbacks.

Hence, The rest of your code only knows about your Models.

Also, you can still use the traditional route and return Observable< `HttpResponse< T >` > from Service API.

edited Feb 14 at 4:20

answered Feb 14 at 4:10

Shane

49 3

0

HttpClient is a new API that came with 4.3, it has updated API's with support for progress events, json deserialization by default, Interceptors and many other great features. See more here ***https://angular.io/guide/http***

Http is the older API and will eventually be deprecated.

Since their usage is very similar for basic tasks I would advise using HttpClient since it is the more modern and easy to use alternative.

answered Apr 15 at 9:08

Chirag

254 2 11
