



# HTTP



Mohamed Mukthar Ahmed

141



## HTTP



### Outline

Introduction

Using @angular/comm/http

Setup

Injecting HttpClient Service

A Basic Request

Building Simple HTTP Component

Building Template

APIs of @angular/comm/http

Making a POST Request

PUT / PATCH / DELETE / HEAD

Making a DELETE Request

Custom HTTP Headers

Summary

Q&A

Mohamed Mukthar Ahmed

142

## ■ Introduction



- **Angular** comes with its own **HTTP** library which we can use to call out to external **APIs**.
- **HTTP** requests are **asynchronous**.
- Dealing with **asynchronous** code is, historically, more **tricky**.
- There are generally three approaches to dealing with **async** code:
  - **Callbacks**
  - **Observables**
  - **Promises**
- In **Angular**, the preferred method of dealing with **async** code is using **Observables**.

Mohamed Mukthar Ahmed

143

## ■ Using @angular/common/http



- **HTTP** has been split into a separate module in Angular.
- To use **HTTP** you need to **import** constants from **@angular/common/http**.

```
import {  
  //      The      NgModule      for      using  
  @angular/common/http  
  HttpClientModule,  
  // the class constants  
  HttpClient  
} from '@angular/common/http';
```

Mohamed Mukthar Ahmed

144

## Using @angular/common/http



```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule, HttpClient } from '@angular/common/http';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

Mohamed Mukthar Ahmed

145

## Setup - @angular/common/http



- In our **app.module.ts** we're going to **import HttpClientModule** which is a convenience collection of modules.
- In our **@NgModule** we will add **HttpClientModule** to the list of **imports**.
  - Doing so, we will be able to **inject HttpClient** into our components.
- Now we can inject the **HttpClient service** into our components

```
export class SimpleHttpComponent implements OnInit {
  data: Object | undefined;
  loading: boolean = false;

  constructor(private http: HttpClient) { }
```

Mohamed Mukthar Ahmed

146

## A Basic Request



- We're going to do make a simple **GET** request to the **jsonplaceholder API**.
- What we're going to do is:
  - Have a button that calls **makeRequest**
  - **makeRequest** will call the **http** library to perform a **GET** request on our API
  - When the request returns, we'll update **this.data** with the results of the **data**, which will be rendered in the view.

```
1 <h1>Basic Request</h1>
2 <button type="button" (click)="makeRequest()">Make Request</button>
3 <div *ngIf="loading">loading...</div>
4 <pre>{{data | json}}</pre>
```

Mohamed Mukthar Ahmed

147

## Building - SimpleHttpComponent



```
export class SimpleHttpComponent implements OnInit {
  data: Object | undefined;
  loading: boolean = false;

  constructor(private http: HttpClient) { }

  ngOnInit(): void {
  }

  makeRequest(): void {
    this.loading = true;
    this.http
      .get('https://jsonplaceholder.typicode.com/posts/1')
      .subscribe(data => {
        this.data = data;
        this.loading = false;
      });
  }
}
```

Mohamed Mukthar Ahmed

148

## A Basic Request

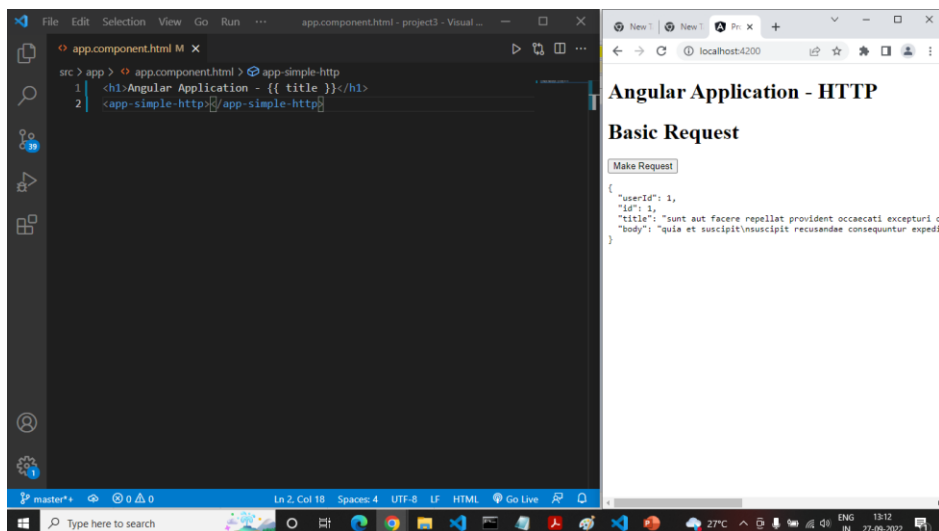


- When we call **makeRequest**, the first thing we do is set **this.loading = true**.
- This will turn on the loading indicator in our view.
- To make an **HTTP** request is straightforward:
  - We call **this.http.get** and pass the URL to which we want to make a **GET** request.
- **http.get** returns an **Observable**.
  - We can **subscribe** to changes using **subscribe**.
- When the request returns, we'll update **this.data** with the results of the **data**.
- We're not loading anymore so we set **this.loading = false**.

Mohamed Mukthar Ahmed

149

## Building the Template



Mohamed Mukthar Ahmed

150

## ■ APIs of @angular/common/http



- The **HTTP** requests we've made is the simple **GET** request.
- It's important that we know how we can make other requests too.
  - **POST**
  - **PUT**
  - **PATCH**
  - **DELETE**
  - **HEAD**

Mohamed Mukthar Ahmed

151

## ■ Making a POST request



- Making **POST** request with **@angular/common/http** is very much like making a **GET** request except that we have one **additional parameter**: a body.
- **jsonplaceholder** API also provides a convent URL for testing our **POST** requests, so let's use it for a POST:

Mohamed Mukthar Ahmed

152

## Making a POST request

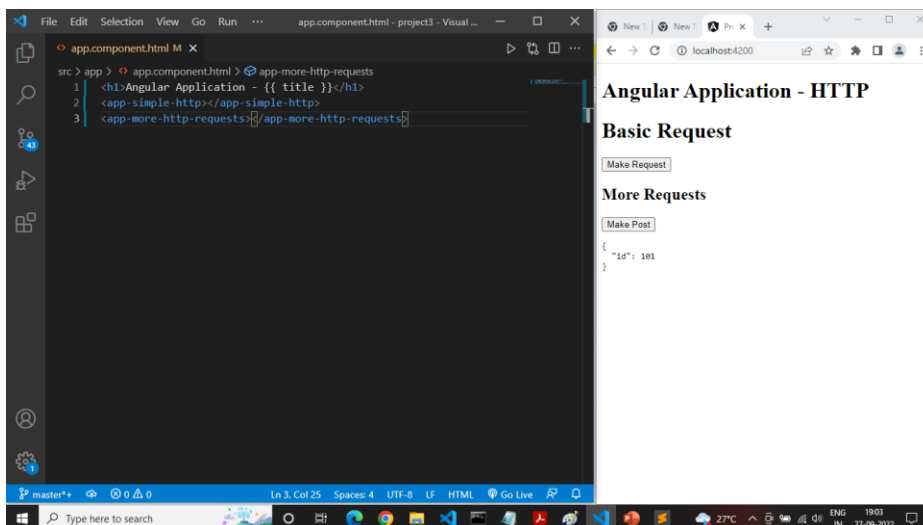


```
makePost(): void {  
  this.loading = true;  
  this.http  
    .post(  
      'https://jsonplaceholder.typicode.com/posts',  
      JSON.stringify({  
        body: 'HTTP POST Method',  
        title: 'Http Methods',  
        userId: 1  
      })  
    )  
    .subscribe(data => {  
      this.data = data;  
      this.loading = false;  
    });  
}
```

Mohamed Mukthar Ahmed

153

## Making a POST request



Mohamed Mukthar Ahmed

154

## ■ PUT / PATCH / DELETE / HEAD



- There are a few other fairly common **HTTP** requests and we call them in much the same way.
- **http.put** and **http.patch** map to **PUT** and **PATCH** respectively and both take a URL and a body
- **http.delete** and **http.head** map to **DELETE** and **HEAD** respectively and both take a URL (no body)

Mohamed Mukthar Ahmed

155

## ■ Making a DELETE Request



- The delete request just take an URL (no body)
- Look into the following code:

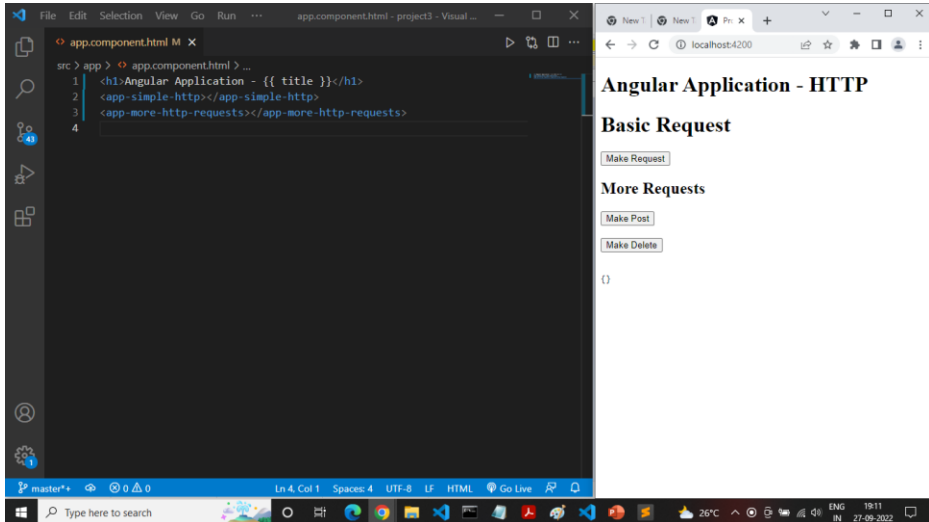
```
makeDelete(): void {  
  this.loading = true;  
  this.http  
    .delete(  
      'https://jsonplaceholder.typicode.com/posts/1'  
    ).subscribe(data => {  
      this.data = data;  
      this.loading = false;  
    });  
}
```

Mohamed Mukthar Ahmed

156



## Making a DELETE request



Mohamed Mukhtar Ahmed

157

## Custom HTTP Headers



- Let's say we want to craft a **GET** request that uses a special **X-API-TOKEN** header.
- We can create a request with this header as follows:

```
makeHeaders(): void {
  const headers: HttpHeaders = new HttpHeaders({
    'X-API-TOKEN': 'my-token'
  });
}
```

- Don't forget to import the **HttpHeaders** and **HttpRequest** modules
- Once the headers are crafted, we can send the request to the **GET** method with the **URL** and the **header**.

Mohamed Mukhtar Ahmed

158

## Custom HTTP Headers



```
makeHeaders(): void {  
  const headers: HttpHeaders = new HttpHeaders({  
    'X-API-TOKEN': 'my-token'  
  });  
  
  const req = new HttpRequest(  
    'GET',  
    'https://jsonplaceholder.typicode.com/posts/1',  
    {  
      headers: headers  
    }  
  );  
  
  this.http.request(req).subscribe(data => {  
    this.data = data;  
  });  
}
```

Mohamed Mukhtar Ahmed

159

## Custom HTTP Headers



The screenshot displays an Angular application with a code editor on the left and a browser on the right. The code editor shows the following HTML structure:

```
1 <h2>More Requests</h2>  
2 <button type="button" (click)="makePost()">Make Post</button>  
3 <br />  
4 <button type="button" (click)="makeDelete()">Make Delete</button>  
5 <br />  
6 <button type="button" (click)="makeHeaders()">Make Headers</button>  
7 <br />  
8 <div *ngIf="loading">loading...</div>  
9 <pre>{{data | json}}</pre>
```

The browser window shows the application running at localhost:4200. The page title is "Angular Application - HTTP". The "More Requests" section contains three buttons: "Make Request", "Make Post", "Make Delete", and "Make Headers". The "Make Headers" button is highlighted. The response data is displayed as a JSON object:

```
{  
  "headers": {  
    "normalizedNames": {},  
    "lazyUpdate": null  
  },  
  "status": 200,  
  "statusText": "OK",  
  "url": "https://jsonplaceholder.typicode.com/posts/1",  
  "ok": true,  
  "type": "4",  
  "body": {  
    "userId": 1,  
    "id": 1,  
    "title": "sunt aut facere repellat provident occaecati excepturi",  
    "body": "quia et suscipit\nsuscipit recusandae consequuntur exp"br/>  }  
}
```

Mohamed Mukhtar Ahmed

160

## Summary



- [@angular/common/http](#) is flexible and suitable for a wide variety of APIs.
- One of the great things about [@angular/common/http](#) is that it has support for mocking the backend which is very useful in testing.

Mohamed Mukhtar Ahmed

161



# Q&A

Mohamed Mukhtar Ahmed

162