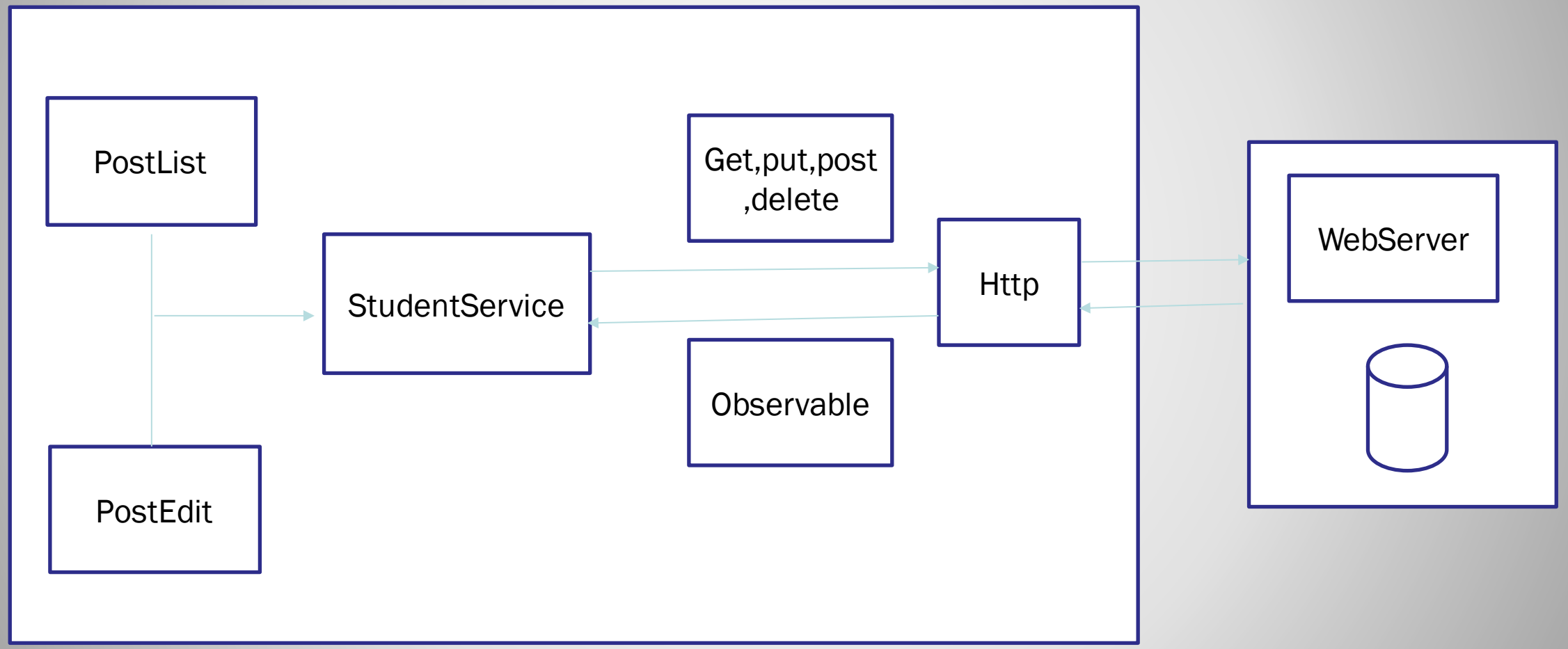
Building Application using Angular

Routing

HTTP And Observables In Angular

HTTP And Observables In Angular



Topics

- Introduction to HTTP requests.
- Using API to make request.
- Using HttpClient API to make HTTP request

Introduction to HTTP requests

- An HTTP request is a packet of information which is transferred from source to destination and termed as Client-Server respectively. A client makes the HTTP request, and the server sends the information to the client. HTTP has multiple request methods to make the request and error to indicate the error.

HTTP Request Methods

- OPTIONS :his is used to check all metadata the server requires to ensure a secure request like CORS. It is called automatically before any HTTP request. You can read more about OPTIONS .
- GET: This is used to get data from the server.
- POST: This is used to send data to the server.
- PUT: This is used to update data.
- DELETE: This is used to delete data from the server

Response Status Codes

- 200: This is used to indicate the "OK" message when the HTTP request has completed successfully.
- 404: This is used to indicate a "Resource NOT Found" message when HTTP doesn't find the specified URL at the server.
- You can read more about HTTP Response codes and HTTP request methods .

Using APIs to Make an HTTP Request

- There are two main types of APIs used to make an HTTP request.
  - XMLHttpRequest(XHR)
  - Fetch

Other libraries like Axios and HttpClient use one of the above internally and abstract the complexities around these APIs.

- You can read more about XMLHttpRequest and Fetch.

Using HttpClient API to make an HTTP request

- The HttpClient in @angular/common/http offers a simplified client HTTP API for Angular applications that rests on the XMLHttpRequest interface exposed by browsers.
- Observable provides better performance over Promise. If you're not familiar with Observable, you can read more about Observable here.
- To use the HttpClient in your project, you need to import HttpClient in app.module.ts. After importing, your app.module.ts file will look like following,:

Using HttpClient API to make an HTTP request

```
import { BrowserModule } from "@angular/platform-browser";
import { HttpClientModule } from "@angular/common/http";

import { NgModule } from "@angular/core";
import { AppComponent } from "./app.component";

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

Create a service

- ng g s httpcruddemo
- Before writing the methods to make the HTTP requests, we need to inject HttpClient in the service class.
  - constructor(private http: HttpClient) { }
- It's best practice to divide the whole URL into two parts :
  - Main URL like https://www.yourserver.com
  - EndPoints like /users

Service look like this

```
export class AppModule {}


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

@Injectable({
  providedIn: 'root'
})
export class HttpCrudDemoService {
  url: string;

constructor(private httpClient: HttpClient) {
  this.url = "https://jsonplaceholder.typicode.com";
}
```

Get request

```
public getPosts(){
  let endPoints="“;
  this.httpClient.get(this.url+endPoints,headers).subscribe(data => {
    console.log(data);
  });
}
```

HTTPClient it converts the data from JSON to an observable form for us. To extract the data from the observable, we need to use the subscribe method of rxjs, which will give the data in a typescript/javascript object which we can use in any other function.

Get Post by ID

```
public getPostById() {
  let id: number = 1;
  let endPoints = "/posts/" + id;
  this.httpClient.get(this.url + endPoints).subscribe(data => {
    console.log(data);
  });
}
```

Add New Post

```
public addPost(postData: Object) {
  let endPoints = ”/posts”
   let headers = new Headers();
      headers.append('Content-Type','application/json');

  this.httpClient.post(this.url + endPoints, postData,{headers:
headers}).subscribe(data => {
    console.log(data);
  });
}
```

The POST method is used for sending the data to the server. It takes two parameters, the service URL and the request body. In many cases, the servers send the ID of the object in response to confirm that your data has been processed by the server and the object has been created successfully.

Update the Post

```
public updatePost(postData: Object) {
  let endPoints = "/posts/1"
  this.httpClient.put(this.url + endPoints, postData).subscribe(data => {
    console.log(data);
  });
}
```

The PUT method is used for updating an object which is already saved in the database. It also requires two parameters, first the URL and second request body. For updating the object, we need to pass the object ID in the URL as a route parameter.

Delete the Post

```
public deletePost() {
  let endPoints = "/posts/1"
  this.httpClient.delete(this.url + endPoints).subscribe(data => {
    console.log(data);
  });
}
```

The DELETE method only requires the URL which has the ID of the object. It checks the ID and deletes the data from the database. But in a real-world application it soft deletes the data, meaning data is not be removed from the server permanently, it just gets inactivated, mainly for reporting purposes.

To make the HTTP request, you should always look for the headers because in many cases the server will only accept the request from an authorized client. In such cases an authorization or access token must be passed in the authorization header.

How it works

- Make a request from StudentService.
- Get the observable and cast it.
- Subscribe the observable to the components.
- Store it to the local variables to make it useful in your component.

Implementing HTTP Functionality

- Open app.module.ts
- Import HttpClientModule

```
import { HttpClientModule } from '@angular/common/http';
imports: [
                BrowserModule,
                 HttpClientModule
        ],
```

- Create json file on location '/assets/AppData/students.json'.

  Right now we are using the JSON file to get the data. When the data will come from web API or services then we just need to change the requesting URL.

Implementing HTTP Functionality(contd..)

- Content of JSON file

```
[
  {"id" : 1001, "name" : "Ritu", "marks" : 100},
  {"id" : 1002, "name" : "Meghna", "marks" : 90},
  {"id" : 1003, "name" : "Rahul", "marks" : 70},
  {"id" : 1004, "name" : "Reena", "marks" : 85},
  {"id" : 1005, "name" : "Rita", "marks" : 60}
]
```

- Add a file (interface) student.ts under app directory and add the below contents,

```
export interface IStudent {
    id: number,
    name: string,
    marks: number
}
```

Implementing HTTP Functionality(contd..)

- Open student.service.ts and add the below contents,

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { IStudent } from './student';
import { Observable } from 'rxjs';

@Injectable()
export class StudentService {

   private url : string ='/assets/AppData/students.json';

  constructor(private http: HttpClient) { }
  getStudents() : Observable<IStudent[]>{
//return observable
    return this.http.get<IStudent[]>(this.url);
  }
```

Open student-details.component.ts and add the below contents,

```
import { Component, OnInit } from '@angular/core';
import { StudentService } from '../student.service';

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

  public students = [];
  constructor(private studentService : StudentService) {
  }

  ngOnInit() {
    this.studentService.getStudents().subscribe(data => this.students = data);

  }
}
```

Open student-marks.component.ts and add the below contents,

```
import { Component, OnInit } from '@angular/core';
import { StudentService } from '../student.service';

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

  public students = [];
  constructor(private studentService : StudentService)
  {
  }

  ngOnInit() {
    this.studentService.getStudents().subscribe(data => this.students = data);
  }

}
```

Conclusion

- HttpClient is one of the best APIs to make HTTP requests. It returns an Observable, but if you want to return a Promise that can also be done using HttpClient. It is best practice to return an Observable and subscribe it in other functions.
- That's it from this guide. I hope you are familiar with HttpClient now.