**ANGULAR HTTPCLIENT EXERCISE**

1. HttpClient is an injectable service that comes with methods to communicate with the remote server. HttpClient API can send Http POST, GET, PUT and DELETE requests easily. HttpClient Methods include:
   1. request()
   2. delete()
   3. get()
   4. head()
   5. jsonp()
   6. options()
   7. patch()
   8. post()
   9. put()
2. GET, PUT, POST and DELETE communicate with a REST API server. We will set up the HttpClientModule in Angular app, make a request using a local server with JSON server NPM package, and how to make GET, POST, PUT & DELETE request with Angular using HttpClient API.
3. Let’s create an employee record management system with Angular and consume a RESTful API via a HttpClient service.
4. Set up a new project. Say yes to adding Angular routing, and use a CSS stylesheet.

ng new angular-httpclient-app

1. Go into the project’s folder: cd angular-httpclient-app
2. We will add the Bootstrap 4 CSS framework npm install bootstrap Go to the angular.json file and add the following to the styles array, adding a comma if necessary:

"node\_modules/bootstrap/dist/css/bootstrap.min.css"

1. Generate components:

ng g c employee-create

ng g c employee-edit

ng g c employee-list

1. Configure a JSON Server in Angular: Let’s create a fake server for testing our Angular app: json-server NPM package. To install JSON server in our project, run the following command in Angular CLI. If we have issues we will use npx json-server to run it instead of installing.

npm i json-server --save

1. Create a folder called server at the same level as src, and keep a database file in it to manage the APIs locally.

mkdir server && cd server

1. Create a file inside server called db.json
2. When the db.json file is created then add some data to it.

{

"employees": [{

"id": 1,

"name": "Tony Stark",

"email": "tony@mcu.com",

"phone": "001-123-4567"

}, {

"id": 2,

"name": "Black Widow",

"email": "black-widow@mcu.com",

"phone": "001-123-4568"

}]

}

1. Run the following command to run:

json-server --watch db.json

1. Or, npx json-server --watch db.json
2. So if you make a request with Http post, put, get or delete, the db.json file will get updated locally. You can see the new data written to the file.
3. Check the local db.json file on this URL http://localhost:3000/employees.
4. For navigating between components in Angular, activate routing service in our application. Add code in the app-routing.module.ts file:

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

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

import { EmployeeCreateComponent } from './employee-create/employee-create.component';

import { EmployeeEditComponent } from './employee-edit/employee-edit.component';

import { EmployeeListComponent } from './employee-list/employee-list.component';

const routes: Routes = [

{ path: '', pathMatch: 'full', redirectTo: 'create-employee' },

{ path: 'create-employee', component: EmployeeCreateComponent },

{ path: 'employees-list', component: EmployeeListComponent },

{ path: 'employee-edit/:id', component: EmployeeEditComponent },

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule],

})

export class AppRoutingModule {}

1. Make sure this is in app.component.html file:

<router-outlet></router-outlet>

1. Import the import AppRoutingModule from ‘./app-routing.module’ in app.module.ts file.;
2. Access the external server to fetch the data using the RESTful API in Angular with HttpClient service that uses HttpClient API.
3. In app.module.ts:

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

1. Include the HttpClientModule in @NgModule's imports array.

@NgModule({

imports: [

HttpClientModule

]

})

1. We have injected the HttpClientModule in your application, so we can use it.
2. The app.module.ts file should look like this:

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

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

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

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

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

import { EmployeeCreateComponent } from './employee-create/employee-create.component';

import { EmployeeEditComponent } from './employee-edit/employee-edit.component';

import { EmployeeListComponent } from './employee-list/employee-list.component';

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

@NgModule({

declarations: [

AppComponent,

EmployeeCreateComponent,

EmployeeEditComponent,

EmployeeListComponent,

],

imports: [

BrowserModule,

HttpClientModule,

FormsModule,

ReactiveFormsModule,

AppRoutingModule,

],

providers: [],

bootstrap: [AppComponent],

})

export class AppModule {}

1. Create a service in order to consume a RESTful API using Angular HttpClient, and do CRUD operations. Generate employee.ts class and rest-api.service.ts files.
2. Generate the employee interface class (this uses the shortcut):

ng g i shared/Employee

1. In shared/employee.ts, define data types within the Employee class.

export class Employee {

id: string;

name: string;

email: string;

phone: number;

}

1. Generate the RestApiService

ng g s shared/rest-api

1. We will also use RxJS to handle asynchronous operations and errors in this demo app. RxJS is an independent Reactive JavaScript library for Observables. https://rxjs.dev/
2. In the shared/rest-api.service.ts file, add the following code:

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Employee } from '../shared/employee';

import { Observable, throwError } from 'rxjs';

import { retry, catchError } from 'rxjs/operators';

@Injectable({

providedIn: 'root',

})

export class RestApiService {

// Define API

apiURL = 'http://localhost:3000';

constructor(private http: HttpClient) {}

/\*========================================

CRUD Methods for consuming RESTful API

=========================================\*/

// Http Options

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

}),

};

// HttpClient API get() method => Fetch employees list

getEmployees(): Observable<Employee> {

return this.http

.get<Employee>(this.apiURL + '/employees')

.pipe(retry(1), catchError(this.handleError));

}

// HttpClient API get() method => Fetch employee

getEmployee(id: any): Observable<Employee> {

return this.http

.get<Employee>(this.apiURL + '/employees/' + id)

.pipe(retry(1), catchError(this.handleError));

}

// HttpClient API post() method => Create employee

createEmployee(employee: any): Observable<Employee> {

return this.http

.post<Employee>(

this.apiURL + '/employees',

JSON.stringify(employee),

this.httpOptions

)

.pipe(retry(1), catchError(this.handleError));

}

// HttpClient API put() method => Update employee

updateEmployee(id: any, employee: any): Observable<Employee> {

return this.http

.put<Employee>(

this.apiURL + '/employees/' + id,

JSON.stringify(employee),

this.httpOptions

)

.pipe(retry(1), catchError(this.handleError));

}

// HttpClient API delete() method => Delete employee

deleteEmployee(id: any) {

return this.http

.delete<Employee>(this.apiURL + '/employees/' + id, this.httpOptions)

.pipe(retry(1), catchError(this.handleError));

}

// Error handling

handleError(error: any) {

let errorMessage = '';

if (error.error instanceof ErrorEvent) {

// Get client-side error

errorMessage = error.error.message;

} else {

// Get server-side error

errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;

}

window.alert(errorMessage);

return throwError(() => {

return errorMessage;

});

}

}

1. Create data using HTTP POST request. In employee-create.component.html add the following code.

<div class="container custom-container">

<div class="col-md-12">

<h3 class="mb-3 text-center">Create Employee</h3>

<div class="form-group">

<input type="text" [(ngModel)]="employeeDetails.name" class="form-control" placeholder="Name">

</div>

<div class="form-group">

<input type="text" [(ngModel)]="employeeDetails.email" class="form-control" placeholder="Email">

</div>

<div class="form-group">

<input type="text" [(ngModel)]="employeeDetails.phone" class="form-control" placeholder="Phone">

</div>

<div class="form-group">

<button class="btn btn-success btn-lg btn-block" (click)="addEmployee(employeeDetails)">Create Employee</button>

</div>

</div>

</div>

1. In employee-create.component.ts file add the following code.

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

import { Router } from '@angular/router';

import { RestApiService } from '../shared/rest-api.service';

@Component({

selector: 'app-employee-create',

templateUrl: './employee-create.component.html',

styleUrls: ['./employee-create.component.scss'],

})

export class EmployeeCreateComponent implements OnInit {

@Input() employeeDetails = { name: '', email: '', phone: 0 };

constructor(public restApi: RestApiService, public router: Router) {}

ngOnInit() {}

addEmployee(dataEmployee: any) {

this.restApi.createEmployee(this.employeeDetails).subscribe((data: {}) => {

this.router.navigate(['/employees-list']);

});

}

}

1. We can now create an employee by making an HTTP POST request via Angular component.
2. Let’s send HTTP GET and DELETE requests: get() and delete()
3. Add this code in employee-list.component.html file:

<div class="container custom-container-2">

<!-- Show it when there is no employee -->

<div class="no-data text-center" \*ngIf="Employee.length == 0">

<p>There is no employee added yet!</p>

<button class="btn btn-outline-primary" routerLink="/create-employee">

Add Empoyee

</button>

</div>

<!-- Employees list table, it hides when there is no employee -->

<div \*ngIf="Employee.length !== 0">

<h3 class="mb-3 text-center">Employees List</h3>

<div class="col-md-12">

<table class="table table-bordered">

<thead>

<tr>

<th scope="col">User Id</th>

<th scope="col">Name</th>

<th scope="col">Email</th>

<th scope="col">Phone</th>

<th scope="col">Action</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let employee of Employee">

<td>{{ employee.id }}</td>

<td>{{ employee.name }}</td>

<td>{{ employee.email }}</td>

<td>{{ employee.phone }}</td>

<td>

<span class="edit" routerLink="/employee-edit/{{ employee.id }}"

>Edit</span>

<span class="delete" (click)="deleteEmployee(employee.id)"

>Delete</span

>

</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

1. Add this code in employee-list.component.ts file:

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

import { RestApiService } from '../shared/rest-api.service';

@Component({

selector: 'app-employee-list',

templateUrl: './employee-list.component.html',

styleUrls: ['./employee-list.component.scss'],

})

export class EmployeeListComponent implements OnInit {

Employee: any = [];

constructor(public restApi: RestApiService) {}

ngOnInit() {

this.loadEmployees();

}

// Get employees list

loadEmployees() {

return this.restApi.getEmployees().subscribe((data: {}) => {

this.Employee = data;

});

}

// Delete employee

deleteEmployee(id: any) {

if (window.confirm('Are you sure, you want to delete?')) {

this.restApi.deleteEmployee(id).subscribe((data) => {

this.loadEmployees();

});

}

}

}

1. We can make a HTTP PUT request, to update current employee data.
2. In employee-edit.component.html:

<div class="container custom-container">

<div class="col-md-12">

<h3 class="mb-3 text-center">Update Employee</h3>

<div class="form-group">

<input type="text" [(ngModel)]="employeeData.name" class="form-control" placeholder="Name">

</div>

<div class="form-group">

<input type="text" [(ngModel)]="employeeData.email" class="form-control" placeholder="Email">

</div>

<div class="form-group">

<input type="text" [(ngModel)]="employeeData.phone" class="form-control" placeholder="Phone">

</div>

<div class="form-group">

<button class="btn btn-success btn-lg btn-block" (click)="updateEmployee()">Update Employee</button>

</div>

</div>

</div>

1. In employee-edit.component.ts

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

import { RestApiService } from "../shared/rest-api.service";

import { ActivatedRoute, Router } from '@angular/router';

@Component({

selector: 'app-employee-details',

templateUrl: './employee-edit.component.html',

styleUrls: ['./employee-edit.component.scss']

})

export class EmployeeEditComponent implements OnInit {

id = this.actRoute.snapshot.params['id'];

employeeData: any = {};

constructor(

public restApi: RestApiService,

public actRoute: ActivatedRoute,

public router: Router

) {

}

ngOnInit() {

this.restApi.getEmployee(this.id).subscribe((data: {}) => {

this.employeeData = data;

})

}

// Update employee data

updateEmployee() {

if(window.confirm('Are you sure, you want to update?')){

this.restApi.updateEmployee(this.id, this.employeeData).subscribe(data => {

this.router.navigate(['/employees-list'])

})

}

}

}

1. Test with ng serve