**HTTPClient Setup**

To use HTTPClient service, you need to do two steps:

1. **Import HttpClientModule in root module**

Import HttpClientModule module from @angular/common/http package and add it’s entry in imports attribute of @NgModule.

|  |
| --- |
| app.module.ts |
| 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 { } |

1. **Inject HttpClient in service constructor**

Now inject actual HttpClient service in your service code as start using it.

|  |
| --- |
| employee.service.ts |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';    @Injectable({    providedIn: 'root'  })  export class EmployeeService {      constructor(private http: HttpClient) { }    } |

**Create service which return Observable**

We will consume the REST API created with [REST mock server](https://howtodoinjava.com/angular/mock-rest-server/). Let’s edit the code of employee service class and return Observable from it.

|  |
| --- |
| employee.service.ts |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { Employee } from '../model/employee';  import { Observable } from 'rxjs';    @Injectable({    providedIn: 'root'  })  export class EmployeeService {      constructor(private http: HttpClient) { }      public getEmployees(): Observable<Employee[]>    {      const url = 'http://localhost:3000/employees';        return this.http.get<Employee[]>(url);    }  } |

Above code, hits the REST API "/employees" and fetch employee array. It then return the employee array as observable collection. Any method can subscribe to it to listen data events on this array.

FYI, Employee is model class to store data.

|  |
| --- |
| employee.ts |
| export class Employee {      public id: number;    public name: string;    public status: string;      constructor(id:number, name:string, status:string) {      this.id = id;      this.name = name;      this.status = status;    }    } |

### Install json-server

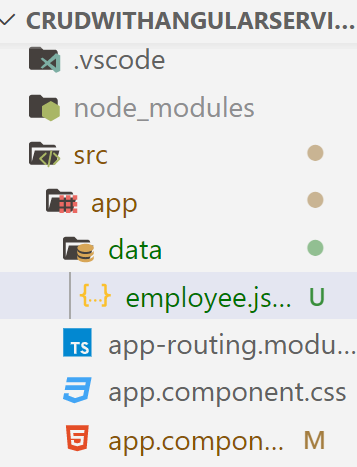
Run the npm command to install the json-server package globally.

$ npm install -g json-server

Copy

### Create a JSON file

Next, create a data.json file inside the **app/data.json**  this file will keep the data that will be used for CRUD operations.



Add the following property in **data.json** file

{

    "employees": [{

            "id": "100",

            "name": "Raj",

            "status": "Active"

        },

        {

            "id": "101",

            "name": "Raju",

            "status": "Active"

        },

        {

            "id": "102",

            "name": "Rajeev",

            "status": "Active"

        },

        {

            "id": "103",

            "name": "Aman",

            "status": "InActive"

        },

        {

            "id": "104",

            "name": "Rahul",

            "status": "Active"

        }

    ]

}

### Start mock server

Finally, run the mock server in new CLI instance with json file specified using below command

json-server --watch ./src/app/data/employee.json

\{^\_^}/ hi!

Loading ./src/app/data.json

Done

Resources

http://localhost:3000/books

Home

http://localhost:3000

Type s + enter at any time to create a snapshot of the database

Watching...

**Create observer which subscribe to Observable**

We will create subscriber in component file. It will read the data from observable array and assign to model attribute. Model attribute can be used to map data from UI.

|  |
| --- |
| app.component.ts |
| import { Component } from '@angular/core';  import { EmployeeService } from './service/employee.service';  import { Employee } from './model/employee';   import{map,catchError}from ‘rxjs/operator’;  @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    title = 'app';    employees = new Array<Employee>();      constructor( empService:EmployeeService ) {        empService.getEmployees().subscribe(response =>      {        this.employees = response.map(item =>        {          return new Employee(              item.id,              item.name,              item.status          );        });      });      }  } |

**View HTML Template**

Time to update view HTML which will render employee array data as soon as it’s available.

|  |
| --- |
| app.component.html |
| <h1>    Angular HTTP Service Example  </h1>  <table border="1" style="width: 33%">    <tr>      <th>Id</th>      <th>Name</th>      <th>Status</th>    </tr>    <tr \*ngFor="let emp of employees">      <td>{{ emp.id }}</td>      <td>{{ emp.name }}</td>      <td>{{ emp.status }}</td>    </tr>  </table> |

**Demo**

To test the above written code, you will to start mock REST server as well as angular application.

1. Start Mock server with this command.

|  |
| --- |
| $ json-server --watch 'E:\ngexamples\db.json' |

1. Start angular application with command.

|  |
| --- |
| $ ng serve |

Check the application in browser.

Open another command prompt window by clicking on the plus symbol

Ng serve --open

