REST API In Angular

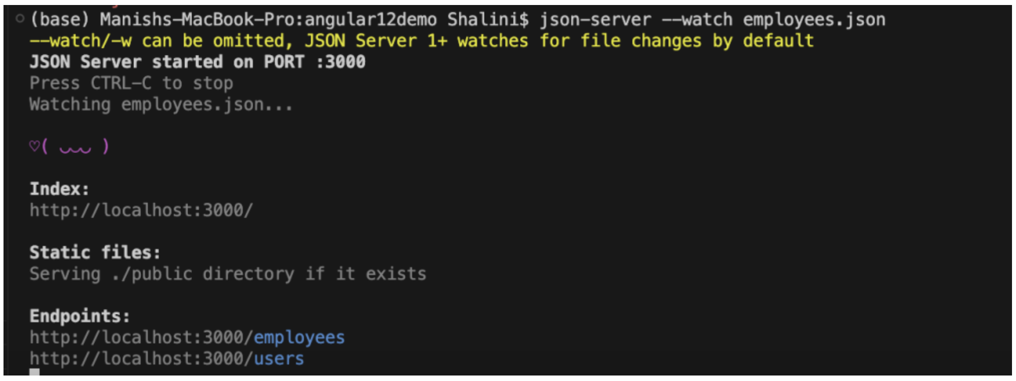
CREATING REST API:

1. Open command prompt or terminal and install json-server that is a light weight server using below command:
   npm install -g json-server

2. Create a file employees.json within the angulardemo folder and add the below code:

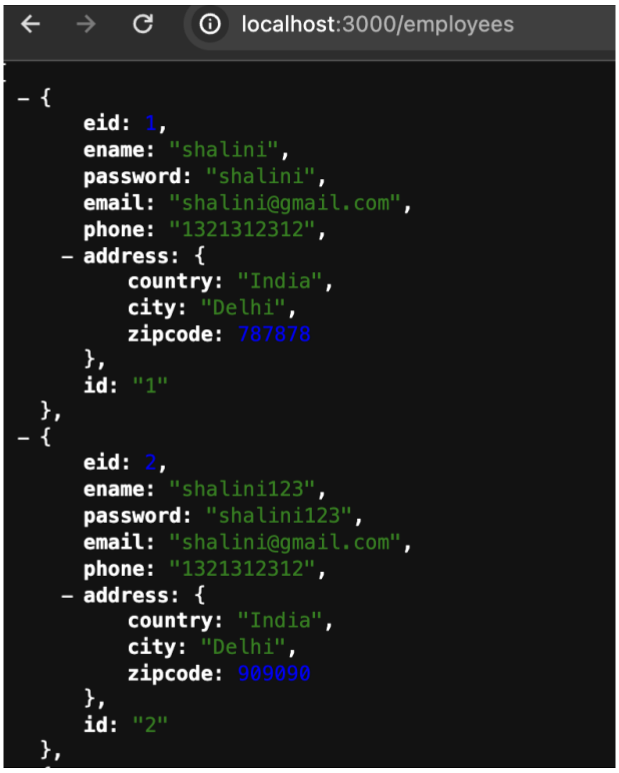
```
    {
 "employees": [
  {
   "eid": 1,
   "ename": "shalini",
   "password": "shalini",
   "email": "shalini@gmail.com",
   "phone": "1321312312",
   "address": {
    "country": "India",
    "city": "Delhi",
    "zipcode": 787878
   },
   "id": 1
  },
  {
   "eid": 2,
   "ename": "shalini123",
   "password": "shalini123",
   "email": "shalini@gmail.com",
   "phone": "1321312312",
   "address": {
    "country": "India",
    "city": "Delhi",
    "zipcode": 909090
   },
   "id": 2
  },
  {
   "eid": 3,
   "ename": "shalini123",
   "password": "shalini123",
   "email": "shalini@gmail.com",
   "phone": "1321312312",
   "address": {
    "country": "India",
    "city": "Delhi",
    "zipcode": 909090
   },
   "id": 3
  },
  {
   "eid": "4",
   "ename": "dummy",
   "email": "dumy@f.c",
   "password": "dummy",
   "phone": "4567890",
   "address": {
    "city": "Mum",
    "country": "India",
    "zipcode": "4567"
   },
   "id": 4
  },
  {
   "eid": "5",
   "ename": "pooja",
      "email": "p@s.d",
      "password": "pooja",
      "phone": "567896789",
      "address": {
        "city": "Pune",
        "country": "India",
        "zipcode": "79799"
      },
      "id": 5
    },
    {
      "eid": "6",
      "ename": "Rushal",
      "email": "rush@d.com",
      "password": "rushil",
      "phone": "67890678",
      "address": {
        "city": "Delhi",
        "country": "India",
        "zipcode": "56789"
      },
      "id": 6
    }
  ],
  "users":[
    {"username":"abc", "password":"abc123"},
    {"username":"pqr", "password":"pqr123"},
    {"username":"user", "password":"user123"}
  ]
}
```

3. Open VSCode terminal from the root of angulardemo folder and execute below command:
   json-server --watch employees.json



```
(base) Manishs-MacBook-Pro:angular12demo Shalini$ json-server --watch employees.json
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching employees.json...

♡( ◡◡ )

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/employees
http://localhost:3000/users
```

4. Open endpoints on the browser and you will see the employees data exposed as a REST API



localhost:3000/employees

```
- {
      eid: 1,
      ename: "shalini",
      password: "shalini",
      email: "shalini@gmail.com",
      phone: "1321312312",
    - address: {
          country: "India",
          city: "Delhi",
          zipcode: 787878
      },
      id: "1"
  },
- {
      eid: 2,
      ename: "shalini123",
      password: "shalini123",
      email: "shalini@gmail.com",
      phone: "1321312312",
    - address: {
          country: "India",
          city: "Delhi",
          zipcode: 909090
      },
      id: "2"
  },
```

OBERVABLES:

1. Create observables component:
   ng g c observables

2. Update observables html as follows:

```
<p >Observable Basics</p>
<hr/>
<b>Observable Data </b>
<div *ngFor="let f of fruits"> {{ f | uppercase }}</div>
<hr>
<div>
 <b>Error Status :</b>
{{anyErrors ? 'error occured ' : 'It All Good'}}
 <hr>
</div>
<div> <b> completion status : </b> {{ finished ? 'Observer completed ': '' }}</div>
<hr>
<button (click)="Start()">Start Emitting</button>
```

3. Lets update observables ts file :

```
data:Observable<string> | null;
  fruits: Array<string> = [];
  anyErrors: boolean =false;
  finished: boolean = false;
  sub:any;

  Start(){
    this.data = new Observable (observer =>
    {
      setTimeout(() => {  observer.next('Apple'); }, 1000);
      setTimeout(() => {  observer.next('mango'); }, 2000);
      setTimeout(() => {  observer.next('Orannge'); }, 3000);
      setTimeout(() => {  observer.next('banana'); }, 4000);
      setTimeout(() => {  observer.next('grapes'); }, 5000);
      setTimeout(() => {  observer.next('watermelon'); }, 6000);
     // setTimeout(() => {  observer.error('something went wrong'); }, 4000);
      setTimeout(() => {  observer.complete(); }, 7000);
    })

     this.sub = this.data.subscribe(fruit => {
        console.log(fruit);
        this.fruits.push(fruit)
      },
      error => this.anyErrors = true,
      () => this.finished = true)

}
  constructor() {
    this.data = null;
  }
  ngOnDestroy(): void {
    this.sub.unsubscribe()
  }

  ngOnInit(): void {
  }
```

https://www.telerik.com/blogs/angular-basics-comparing-data-producers-javascript-functions-promises-iterables-observables

https://www.telerik.com/blogs/angular-basics-introduction-observables-rxjs-part-1

https://www.telerik.com/blogs/angular-basics-introduction-observables-rxjs-part-2

HTTP CALLS IN ANGULAR:

1. To configure for listening to HTTP endpoint in angular, add HttpClientModule in app.module.ts file

2. Execute below command from within the service folder:
   ng g s emphttp

3. Add below property in emphttp service
   url:string = "http://localhost:8080/employees";

4. Update emphttp service to make a REST API call:

   constructor(private http:HttpClient) { }

```
getAllEmployees():Observable<any>
{
  return this.http.get<any>(this.url);
}
getEmployeeById(eid:number):Observable<Employee>
{
  return this.http.get<Employee>(this.url+'/'+eid);
}
addEmployee(employee:Employee):Observable<Employee>
{
  return this.http.post<Employee>(this.url, employee);
}
updateEmployee(employee:Employee):Observable<Employee>
{
  return this.http.put<Employee>(this.url+'/'+employee.eid, employee);
}
deleteEmployee(eid:number){
  return this.http.delete(this.url+'/'+eid)
}
```

5. Update employees list to use the service to fetch data from REST API

```
    constructor(private empservice:EmphttpService){}

    ngOnInit(): void {
  this.empservice.getAllEmployees()
  .subscribe(resp => {
   console.log('fetched employees')
   console.log(resp);
    this.employees = resp;
  })
 }
```

6. Likewise update emp form to make a POST request and respectively for update and delete as well.

7. Alternatively in employee list component instead of manual subscription to the observable, the code of ts and html can be updated as follows:

```
eList:Observable<Employee[]> = new Observable<Employee[]>();

  ngOnInit(): void {
    this.eList = this.empservice.getAllEmployees();
    // .subscribe(resp => {
    //  console.log('fetched employees')
    //  console.log(resp);
    //   this.eList = resp;
    // })
  }

@for(emp1 of eList | async; track emp1.eid){
        // other code
}
```

8. Also can transform the data as follows:

```
return this.http.get<any>(this.url)
    .pipe(
      tap(resp=> console.log(resp)),
      map(resp => resp.map((e: Employee) => {
        e.ename = e.ename.toUpperCase();
        return e;
   })));
```

Or

```
return this.http.get<Employee[]>(this.url)
    .pipe(mergeMap(v => v),
      map(r => {
        r.ename = r.ename.toUpperCase();
        return r;
      })
      ,toArray()
    );
```

Dummy sample
https://swapi.dev/api/people

https://www.thisdot.co/blog/mapping-returned-http-data-with-rxjs