1. **Develop Angular Frontend Application**

Let's develop a step-by-step CRUD (Create, Read, Update, Delete) web application using Angular 12 which consumes CRUD rest APIs.

## **. Create Angular App using Angular CLI**

***ng new frontendApp --no-standalone***

## **. Create Angular Components and Service Classes using Angular CLI**

1. – ng g c create-employee
2. – ng g c update-employee
3. – ng g c employee-details
4. – ng g c employee-list
5. - ng g s employee

## **1.3. Integrate JQuery and Bootstrap with Angular**

npm install bootstrap jquery --save

Configure installed **Bootstrap** & **JQuery** in an angular.json file:

..

"styles": [

 "src/styles.css",

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

],

"scripts": [

 "node\_modules/jquery/dist/jquery.min.js",

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

]

...

## **1.4. Create an Employee Model (TypeScript)**

**Path - src/app/employee.ts**

Before defining the **EmployeeListComponent**, let’s define an **Employee** class for working with employees. create a new file employee.ts inside src/app folder and add the following code to it -

export class Employee {

 id: number;

 firstName: string;

 lastName: string;

 emailId: string; }

## **1.5. Create Employee Service - REST Client**

**Path - src/app/employee.service.ts**

The **EmployeeService** will be used to get the data from the backend by calling spring boot APIs. Update the employee.service.ts file inside src/app directory with the following code to it -

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

import { HttpClient } from '@angular/common/http'

import { Observable } from 'rxjs';

import { Employee } from './employee';

@Injectable({

 providedIn: 'root'

})

export class EmployeeService {

 private baseURL = "http://localhost:8080/api/v1/employees";

 constructor(private httpClient: HttpClient) { }

 getEmployeesList(): Observable<Employee[]>{

 return this.httpClient.get<Employee[]>(`${this.baseURL}`);

 }

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

 return this.httpClient.post(`${this.baseURL}`, employee);

 }

 getEmployeeById(id: number): Observable<Employee>{

 return this.httpClient.get<Employee>(`${this.baseURL}/${id}`);

 }

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

 return this.httpClient.put(`${this.baseURL}/${id}`, employee);

 }

 deleteEmployee(id: number): Observable<Object>{

 return this.httpClient.delete(`${this.baseURL}/${id}`);

 }

}

## **1.6. Creating Employee List Component and Template**

### Path - src/app/employee-list/**employee-list.component.ts**

Let's create the **EmployeeListComponent** component which will be used to display a list of employees, create a new employee, and delete an employee.

**Update/**remove the content of employee-list.component.ts inside src/app directory and add the following code to it -

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

import { Employee } from '../employee'

import { EmployeeService } from '../employee.service'

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

@Component({

 selector: 'app-employee-list',

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

 styleUrls: ['./employee-list.component.css']

})

export class EmployeeListComponent implements OnInit {

 employees: Employee[];

 constructor(private employeeService: EmployeeService,

 private router: Router) { }

 ngOnInit(): void {

 this.getEmployees();

 }

 private getEmployees(){

 this.employeeService.getEmployeesList().subscribe(data => {

 this.employees = data;

 });

 }

 employeeDetails(id: number){

 this.router.navigate(['employee-details', id]);

 }

 updateEmployee(id: number){

 this.router.navigate(['update-employee', id]);

 }

 deleteEmployee(id: number){

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

 console.log(data);

 this.getEmployees();

 })

 }

}

### **Path - src/app/employee-list/employee-list.component.html**

Add employee-list.component.html file with the following code to it -

<div class = "row">

 <h2> Employee List</h2>

</div>

<table class = "table table-striped table-bordered">

 <thead>

 <tr>

 <th> First Name</th>

 <th> Last Name </th>

 <th> Email Id</th>

 <th> Actions </th>

 </tr>

 </thead>

 <tbody>

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

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

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

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

 <td>

 <button (click) = "updateEmployee(employee.id)" class = "btn btn-primary"> Update</button>

 <button (click) = "deleteEmployee(employee.id)" class = "btn btn-danger" style="margin-left: 10px"> Delete</button>

 <button (click) = "employeeDetails(employee.id)" class = "btn btn-primary" style="margin-left: 10px"> View</button>

 </td>

 </tr>

 </tbody>

</table>

## **1.7. Creating Employee List Component and Template**

### Path - src/app/create-employee/create-employee.component.ts

**CreateEmployeeComponent** is used to create and handle a new employee form data. Add the following code to it -

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

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

import { EmployeeService } from '../employee.service';

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

@Component({

 selector: 'app-create-employee',

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

 styleUrls: ['./create-employee.component.css']

})

export class CreateEmployeeComponent implements OnInit {

 employee: Employee = new Employee();

 constructor(private employeeService: EmployeeService,

 private router: Router) { }

 ngOnInit(): void {

 }

 saveEmployee(){

 this.employeeService.createEmployee(this.employee).subscribe( data =>{

 console.log(data);

 this.goToEmployeeList();

 },

 error => console.log(error));

 }

 goToEmployeeList(){

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

 }

 onSubmit(){

 console.log(this.employee);

 this.saveEmployee();

 }

}

### **Path - src/app/create-employee/create-employee.component.html**

The create-employee.component.html shows the add employee HTML form. Add the following code to it -

<div class="row">

 <div class="card col-md-6 offset-md-3 offset-md-3">

 <div class="row">

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

 <hr />

 <div class="card-body">

 <form (ngSubmit)="onSubmit()">

 <div class="form-group">

 <label> First Name</label>

 <input type="text" class="form-control" id="firstName" [(ngModel)]="employee.firstName"

 name="firstName">

 </div>

 <div class="form-group">

 <label> Last Name</label>

 <input type="text" class="form-control" id="lastName" [(ngModel)]="employee.lastName"

 name="lastName">

 </div>

 <div class="form-group">

 <label> Email Id</label>

 <input type="text" class="form-control" id="emailId" [(ngModel)]="employee.emailId"

 name="emailId">

 </div>

 <br />

 <button class="btn btn-success" type="submit">Submit</button>

 </form>

 </div>

 </div>

 </div>

</div>

## **1.8. Create Update Employee Component and Template**

In this UpdateEmployeeComponent, we first get the employee object using REST API and populate it in HTML form via data binding. Users can edit the employee form data and submit the form.

Let's add the following code to UpdateEmployeeComponent -

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

import { EmployeeService } from '../employee.service';

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

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

@Component({

 selector: 'app-update-employee',

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

 styleUrls: ['./update-employee.component.css']

})

export class UpdateEmployeeComponent implements OnInit {

 id: number;

 employee: Employee = new Employee();

 constructor(private employeeService: EmployeeService,

 private route: ActivatedRoute,

 private router: Router) { }

 ngOnInit(): void {

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

 this.employeeService.getEmployeeById(this.id).subscribe(data => {

 this.employee = data;

 }, error => console.log(error));

 }

 onSubmit(){

 this.employeeService.updateEmployee(this.id, this.employee).subscribe( data =>{

 this.goToEmployeeList();

 }

 , error => console.log(error));

 }

 goToEmployeeList(){

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

 }

}

### **Path - src/app/update-employee/update-employee.component.html**

The *update-employee.component.html* shows the updated employee HTML form. Add the following code to this file -

<div class="row">

 <div class="card col-md-6 offset-md-3 offset-md-3">

 <div class="row">

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

 <hr />

 <div class="card-body">

 <form (ngSubmit)="onSubmit()">

 <div class="form-group">

 <label> First Name</label>

 <input type="text" class="form-control" id="firstName" [(ngModel)]="employee.firstName"

 name="firstName">

 </div>

 <div class="form-group">

 <label> Last Name</label>

 <input type="text" class="form-control" id="lastName" [(ngModel)]="employee.lastName"

 name="lastName">

 </div>

 <div class="form-group">

 <label> Email Id</label>

 <input type="text" class="form-control" id="emailId" [(ngModel)]="employee.emailId"

 name="emailId">

 </div>

 <br />

 <button class="btn btn-success" type="submit">Submit</button>

 </form>

 </div>

 </div>

 </div>

</div>

## **1.9 Create View Employee Details Component and Template**

### Path - src/app/employee-details/employee-details.component.ts

The *EmployeeDetailsComponent* component is used to display a particular employee detail. Add the following code to it -

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

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

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

import { EmployeeService } from '../employee.service';

@Component({

 selector: 'app-employee-details',

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

 styleUrls: ['./employee-details.component.css']

})

export class EmployeeDetailsComponent implements OnInit {

 id: number

 employee: Employee

 constructor(private route: ActivatedRoute, private employeService: EmployeeService) { }

 ngOnInit(): void {

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

 this.employee = new Employee();

 this.employeService.getEmployeeById(this.id).subscribe( data => {

 this.employee = data;

 });

 }

}

### **Path - src/app/employee-details/employee-details.component.html**

The employee-details.component.html displays a particular employee detail. Add the following code to it -

<h3> View Employee Details</h3>

<div>

 <div>

 <label> <b> First Name: </b></label> {{employee.firstName}}

 </div>

 <div>

 <label> <b> Last Name: </b></label> {{employee.lastName}}

 </div>

 <div>

 <label> <b> Email Id: </b></label> {{employee.emailId}}

 </div>

</div>

## **1.10. package.json - Configure Dependencies**

### Path: /package.json

The package.json file contains project configuration information including package dependencies that get installed when you run npm install.

## **1.11. App Routing Module**

### **Path: /src/app/app.routing.module.ts**

Routing for the Angular app is configured as an array of **Routes**, each component is mapped to a path so the Angular Router knows which component to display based on the URL in the browser address bar.

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

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

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

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

import { UpdateEmployeeComponent } from './update-employee/update-employee.component';

import { EmployeeDetailsComponent } from './employee-details/employee-details.component';

const routes: Routes = [

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

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

 {path: '', redirectTo: 'employees', pathMatch: 'full'},

 {path: 'update-employee/:id', component: UpdateEmployeeComponent},

 {path: 'employee-details/:id', component: EmployeeDetailsComponent}

];

@NgModule({

 imports: [RouterModule.forRoot(routes)],

 exports: [RouterModule]

})

export class AppRoutingModule { }

## **1.12. App Component Template**

### Path: /src/app/app.component.html

Defines the HTML template associated with the root AppComponent

<nav class="navbar navbar-expand-sm bg-primary navbar-dark">

 <ul class = "navbar-nav">

 <li class = "nav-item">

 <a routerLink="employees" routerLinkActive="active" class="nav-link" >Employee List</a>

 </li>

 <li class = "nav-item">

 <a routerLink="create-employee" routerLinkActive="active" class="nav-link" >Add Employee</a>

 </li>

 </ul>

</nav>

<h1 class="text-center"> {{title}} </h1>

<div class = "container">

 <router-outlet></router-outlet>

</div>

<footer class = "footer">

 <div class = "container">

 <span>All Rights Reserved 2025 </span>

 </div>

</footer>

## **1.13. App Module**

Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here.

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

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

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

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

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

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

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

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

import { UpdateEmployeeComponent } from './update-employee/update-employee.component';

import { EmployeeDetailsComponent } from './employee-details/employee-details.component'

@NgModule({

 declarations: [

 AppComponent,

 EmployeeListComponent,

 CreateEmployeeComponent,

 UpdateEmployeeComponent,

 EmployeeDetailsComponent

 ],

 imports: [

 BrowserModule,

 AppRoutingModule,

 HttpClientModule,

 FormsModule

 ],

 providers: [],

 bootstrap: [AppComponent]

})

export class AppModule { }

## **1.14. Running Angular Client Application**

Let's run the above developed Angular App with a command:

ng serve

Just go to **tsconfig.json** and set

"compilerOptions": {

 "strict": false,

 ...

} https://www.javaguides.net/2021/08/angular-crud-example-with-spring-boot.html