* Spring Boot Application

Previous application we had created a simple spring boot application which exposed a REST endpoint for fetching a list of employees. In this tutorial **we will be adding 2 more REST endpoints - One for creating an employee and other for deleting it.**

package com.javainuse.controllers;

import java.util.ArrayList;

import java.util.List;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.javainuse.model.Employee;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping({ "/employees" })

public class TestController {

private List<Employee> employees = createList();

@GetMapping(produces = "application/json")

public List<Employee> firstPage() {

return employees;

}

**@DeleteMapping(path = { "/{id}" })**

**public Employee delete(@PathVariable("id") int id) {**

**Employee deletedEmp = null;**

**for (Employee emp : employees) {**

**if (emp.getEmpId().equals(id)) {**

**employees.remove(emp);**

**deletedEmp = emp;**

**break;**

**}**

**}**

**return deletedEmp;**

**}**

**@PostMapping**

**public Employee create(@RequestBody Employee user) {**

**employees.add(user);**

**System.out.println(employees);**

**return user;**

**}**

private static List<Employee> createList() {

List<Employee> tempEmployees = new ArrayList<>();

Employee emp1 = new Employee();

emp1.setName("emp1");

emp1.setDesignation("manager");

emp1.setEmpId("1");

emp1.setSalary(3000);

Employee emp2 = new Employee();

emp2.setName("emp2");

emp2.setDesignation("developer");

emp2.setEmpId("2");

emp2.setSalary(3000);

tempEmployees.add(emp1);

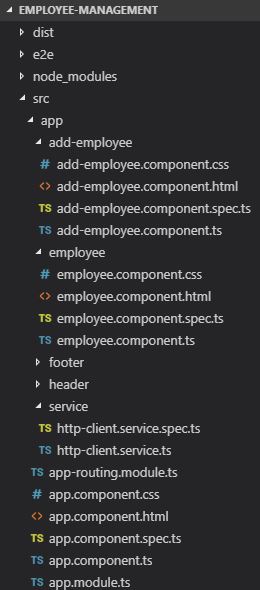
tempEmployees.add(emp2);

return tempEmployees;

}

}

Angular 7 development

The angular project we will be developing is as follows-   


* Modify existing HtppClient Service

We modify the HttpClient service to add methods for **performing add new employee and deleting employee** in addition to getting the list of employees using the httpClient.

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

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

export class Employee{

constructor(

public empId:string,

public name:string,

public designation:string,

public salary:string,

) {}

}

@Injectable({

providedIn: 'root'

})

export class HttpClientService {

constructor(

private httpClient:HttpClient

) {

}

getEmployees()

{

console.log("test call");

return this.httpClient.get<Employee[]>('http://localhost:8080/employees');

}

**public deleteEmployee(employee) {**

**return this.httpClient.delete<Employee>("http://localhost:8080/employees" + "/"+ employee.empId);**

**}**

**public createEmployee(employee) {**

**return this.httpClient.post<Employee>("http://localhost:8080/employees", employee);**

**}**

}

* Adding BootStrap CSS

In the style.css add the bootstrap css url-

/\* You can add global styles to this file, and also import other style files \*/

**@import url(https://unpkg.com/bootstrap@4.1.0/dist/css/bootstrap.min.css)**

* Modify existing employee component to add delete functionality

In the employee.component.ts file add a call for deleting the employee in addition to getting the list of employees.

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

import { HttpClientService, Employee } from '../service/http-client.service';

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

employees: Employee[];

constructor(

private httpClientService: HttpClientService

) { }

ngOnInit() {

this.httpClientService.getEmployees().subscribe(

response =>{this.employees = response;}

);

}

**deleteEmployee(employee: Employee): void {**

**this.httpClientService.deleteEmployee(employee)**

**.subscribe( data => {**

**this.employees = this.employees.filter(u => u !== employee);**

**})**

};

}

We will be displaying the list of employees in employee.component.html and give users option to delete the employee.

<div class="col-md-6">

<table class="table table-striped">

<thead>

<tr>

<th>name</th>

<th>designation</th>

</tr>

</thead>

<tbody>

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

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

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

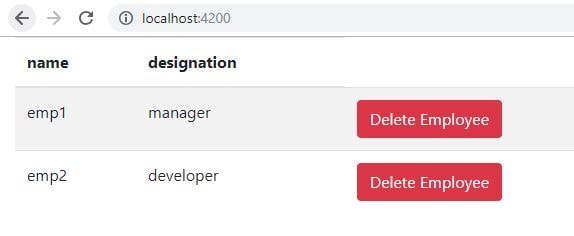
<td><button class="btn btn-danger" (click)="deleteEmployee(employee)"> Delete Employee</button></td>

</tr>

</tbody>

</table>

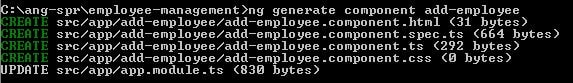
</div>

Now if we go to localhost:4200   


* Creating the add employee component

We will be creating a new component named add-employee

ng generate component add-employee

  
Modify the add-employee.component.ts to make call to spring boot for creating new employee using the httpClientService which is injected in the class using constructor injection.

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

import { HttpClientService, Employee } from '../service/http-client.service';

@Component({

selector: 'app-add-employee',

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

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

})

export class AddEmployeeComponent implements OnInit {

user: Employee = new Employee("","","","");

constructor(

private httpClientService: HttpClientService

) { }

ngOnInit() {

}

**createEmployee(): void {**

**this.httpClientService.createEmployee(this.user)**

**.subscribe( data => {**

**alert("Employee created successfully.");**

**});**

};

}

Modify the add-employee.component.html to create form for getting the new employee details for the employee object to be created.

<div class="col-md-6">

<h2 class="text-center">Add Employee</h2>

<form>

<div class="form-group">

<label for="name">Name:</label>

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

</div>

<div class="form-group">

<label for="designation">Designation:</label>

<input [(ngModel)]="user.designation" placeholder="Designation" name="designation" class="form-control" id="designation">

</div>

<div class="form-group">

<label for="empId">Employee Id</label>

<input [(ngModel)]="user.empId" placeholder="Employee Id" name="Employee Id" class="form-control" id="employeeid">

</div>

<button class="btn btn-success" (click)="createEmployee()">Create</button>

</form>

</div>

Since we are using ngModel directive, we will need to add the FormsModule in the app.module.ts- If this is not done you will get an exception as follows - **Can't bind to 'ngModel' since it isn't a known property of 'input'. ("**

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

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

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

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

import { EmployeeComponent } from './employee/employee.component';

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

import { AddEmployeeComponent } from './add-employee/add-employee.component';

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

import { HeaderComponent } from './header/header.component';

import { FooterComponent } from './footer/footer.component';

@NgModule({

declarations: [

AppComponent,

EmployeeComponent,

AddEmployeeComponent,

HeaderComponent,

FooterComponent

],

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule,

**FormsModule**

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

In the app.routing.js add the route for this add employee component

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

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

import { EmployeeComponent } from './employee/employee.component';

import { AddEmployeeComponent } from './add-employee/add-employee.component';

const routes: Routes = [

{ path:'', component: EmployeeComponent},

**{ path:'addemployee', component: AddEmployeeComponent},**

];

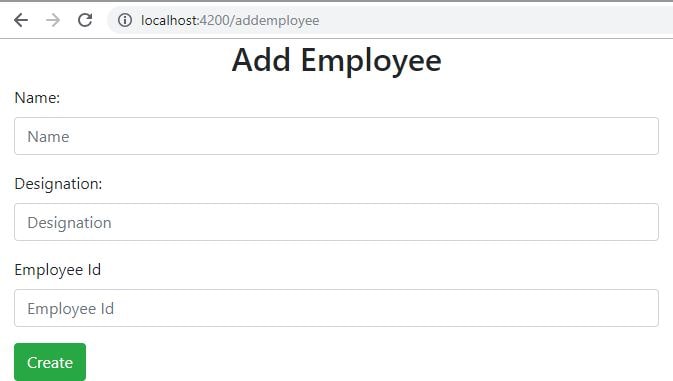
@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

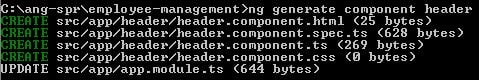
export class AppRoutingModule { }

Go to localhost:8080/addemployee   


* Creating the header menu

Create new header component-

ng generate component header

  
Modify the header.component.html to create the menu with links to the view employees and add new employee pages.

<header>

<nav class ="navbar navbar-expand-md navbar-dark bg-dark">

<div><a href="https://www.javainuse.com" class="navbar-brand">JavaInUse</a></div>

<ul class="navbar-nav">

<li><a routerLink="/" class="nav-link">View Employees</a></li>

<li><a routerLink="/addemployee" class="nav-link">Add New Employee</a></li>

</ul>

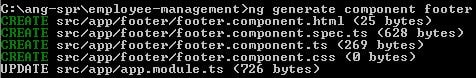
</nav>

</header>

* Create Footer Component

Create a new component named footer-

ng generate component footer

  
In the footer specify the data-

<footer class="footer">

<div class="container">

<span class="text-muted">All Rights Reserved 2019 @JavaInUse</span>

</div>

</footer>

Modify the footer.component.css-

.footer {

position: absolute;

bottom: 0;

width:100%;

height: 40px;

background-color: #222222;

}

* Modify the app.component.html to add the footer and menu to the application byadding the selector tags.
* **<app-header></app-header>**
* <router-outlet></router-outlet>
* **<app-footer></app-footer>**

Go to localhost:4200   
