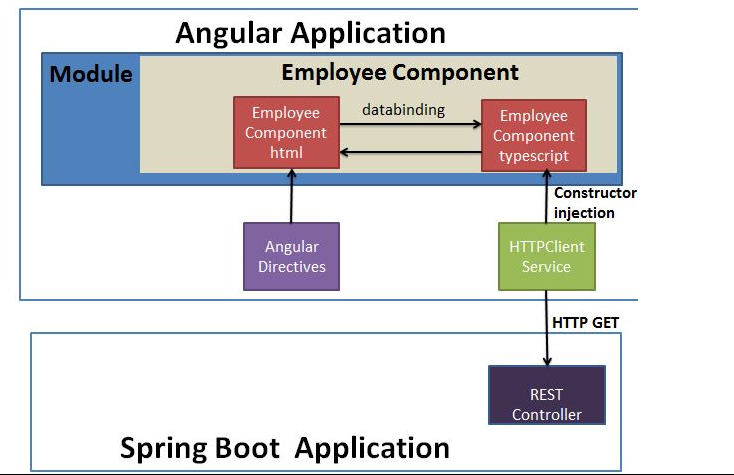
Explanation of Angular and Spring Boot Application

1. We are developing full stack application with frontend and backend and they are loosely coupled.
2. The backend what it does is that it expose the Restend points which will be consumed by the frontend for displaying the data.
3. In future even if another front end device is to be used, there will not be much change and the new device will need to consume these end points.
4. We will be developing springboot application and Angular application. In the spring boot application we will have the RestController. The endpoints will be consumed by the angular application.
5. In angular application there will be something known as components and services.
6. In spring boot application we will have to expose a Rest endpoint to return a list of employees.
7. We will develop a maven project.



Spring Boot Application

SpringBootHelloWorldApplication.java

* *It is a BootStrap class with @SpringBootApplication annotation.*

package com.sandip;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}}

TestController.java

It has 3 annotations

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

It is use to state that calls to the controller will be from different Domain. The Angular application will run on port no :- http://localhost:4200

2) @RestController

3) @RequestMapping(value = "/employees", method = RequestMethod.GET, produces = "application/json")

This will return the list of employee .We have a static method which will return the list of employee in json format.

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.RequestMapping;

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

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

import com.sandip.model.Employee;

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

@RestController

public class TestController {

private List<Employee> employees = createList();

@RequestMapping(value = "/employees", method = RequestMethod.GET, produces = "application/json")

public List<Employee> firstPage() {

return employees;

}

private static List<Employee> createList() {

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

Employee emp1 = new Employee();

emp1.setName("Kiran");

emp1.setDesignation("manager");

emp1.setEmpId("1");

emp1.setSalary(3000);

Employee emp2 = new Employee();

emp2.setName("sandip");

emp2.setDesignation("developer");

emp2.setEmpId("2");

emp2.setSalary(3000);

tempEmployees.add(emp1);

tempEmployees.add(emp2);

return tempEmployees;

}}

package com.sandip.model;

public class Employee {

private String empId;

private String name;

private String designation;

private double salary;

public Employee() {

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public String getEmpId() {

return empId;

}

public void setEmpId(String empId) {

this.empId = empId;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((designation == null) ? 0 : designation.hashCode());

result = prime \* result + ((empId == null) ? 0 : empId.hashCode());

result = prime \* result + ((name == null) ? 0 : name.hashCode());

long temp;

temp = Double.doubleToLongBits(salary);

result = prime \* result + (int) (temp ^ (temp >>> 32));

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

Employee other = (Employee) obj;

if (designation == null) {

if (other.designation != null)

return false;

} else if (!designation.equals(other.designation))

return false;

if (empId == null) {

if (other.empId != null)

return false;

} else if (!empId.equals(other.empId))

return false;

if (name == null) {

if (other.name != null)

return false;

} else if (!name.equals(other.name))

return false;

if (Double.doubleToLongBits(salary) != Double.doubleToLongBits(other.salary))

return false;

return true; }}

POM.XML

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.sandip</groupId>

<artifactId>SpringBootHelloWorld</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringBootHelloWorld</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.1.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Angular Application

1) Installing Angular CLI

2) Install node js by downloading the installable from Install NodeJS

3) install angular cli using the following command. It wll get us the latest version of angular cli.

npm install -g @angular/cli

4)We can check the angular cli version -

ng version

5) Next we will create a new angular project using the angular cli

ng new employee-management

6) To get the angular cli project started use the following command. We must go inside the employee-management folder and then use it.

ng serve

7) Go to localhost:4200

8) install visualStudioCode ID

9)Package.json is for all dependencies required for angular application. It is similar to Maven used in our spring application. All dependencies will be downloaded into the node module.

10) tsconfig.json is used to convert typescript file into the javascript file.

11)tslint.json specify the coding standard of angular application.

12) TypeScript

TypeScript is a superset of JavaScript. It is a strongly typed language. So unlike JavaScript we know if some syntax is wrong while typing itself and not at runtime. In Angular it is compiled to JavaScript while rendering application in browser.

13)Component

In angular we break complex code into reusable parts called components. Major part of the development with Angular 7 is done in the components. Components are basically classes that interact with the .html file of the component, which gets displayed on the browser.

Example :- our EmployeeManagementSystem Application is broken down to components like

1) getEmployee Component

2) addNewEmployee Component

3) Login/Logout Component

4) Menu Component

Components consists of 3 files

1. Component.ts
2. Component.html
3. Component.css

14) Service

In angular where some code needs to be reused in multiple components. For example a data connection that fetches data from database might be needed in multiple components. This is achieved using services.

HttpClient Servive will be responsible to get data from springboot application.

15) Create employee component

We will be creating Employee Component which will fetch data from spring boot and display it.

ng generate component employee

16) Create HttpClient Service

We will be creating a HTTPClient Service. This service will be having the httpClient and will be responsible for calling http GET request to the backend spring boot application.

ng generate service service/httpClient

17) The following service files are created-

http-client.service.ts

In the constructor define the HTTPClient instance we will be using to make a call to the Spring Boot application. Here we will be using the Angular HTTPClient for calling the Spring Boot API to fetch the employee data. Also we will creating a method which makes call to the spring boot application using the defined httpClient.

@ Injectable :-Identify that it is a service class.

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');

}}

\*Also we need to add the HTTPClientModule to the app.module.ts

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';

@NgModule({

declarations: [

AppComponent,

EmployeeComponent

],

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

Insert HttpClient Service in Employee Component

Next using constructor dependency injection we will be providing the EmployeeComponent an instance of HttpClientService. Using this service we make a call to spring boot application to get a list of employees.

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

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

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

employees:string[];

constructor(

private httpClientService:HttpClientService

) { }

ngOnInit() {

this.httpClientService.getEmployees().subscribe(

response =>this.handleSuccessfulResponse(response),

); }

handleSuccessfulResponse(response)

{ this.employees=response;}}

In the employee.component.html we iterate over the list of employees we got in the employee.component.ts file.

<table border="1">

<thead></thead>

<tr>

<th>name</th>

<th>designation</th>

</tr>

<tbody>

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

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

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

</tr>

</tbody>

</table>