FSD - Final Project - Group II

Spring Boot Integration With Angular 14 CRUD

# ****Introduction****

 In this project, you will develop a CRUD (create read update delete) web application using Angular 14 and Spring Boot with Spring REST. The application contains the Employee form which has CRUD operations like add, view, delete, and update it. For the front end, we will use Angular and consume the service. Whereas the backend will be in Spring Boot which will provide us some data.

## ****BackEnd Technology****

* SpringBoot 2
* Hibernate 5
* Spring Data JPA

## **FrontEnd Technology**

* Angular11
* Bootstrap
* Jquery

****So We will be creating it in 5 parts****

*1. Creating SpringBoot project*

*2. Creating Rest API using Spring Boot*

*3. Create an Angular App*

*4. Creating Component, Service and Modal files in angular*

*5 Configuring App to run*

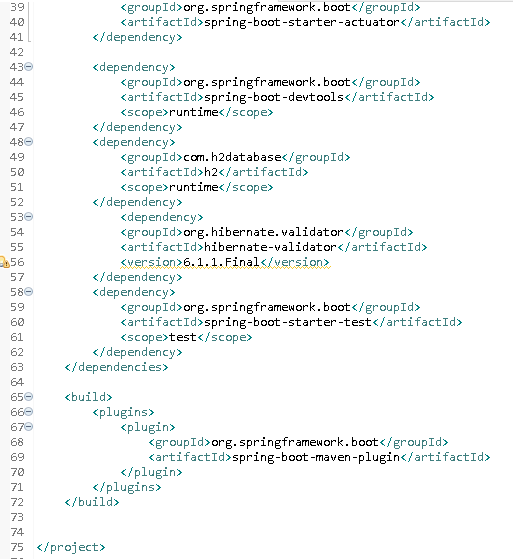
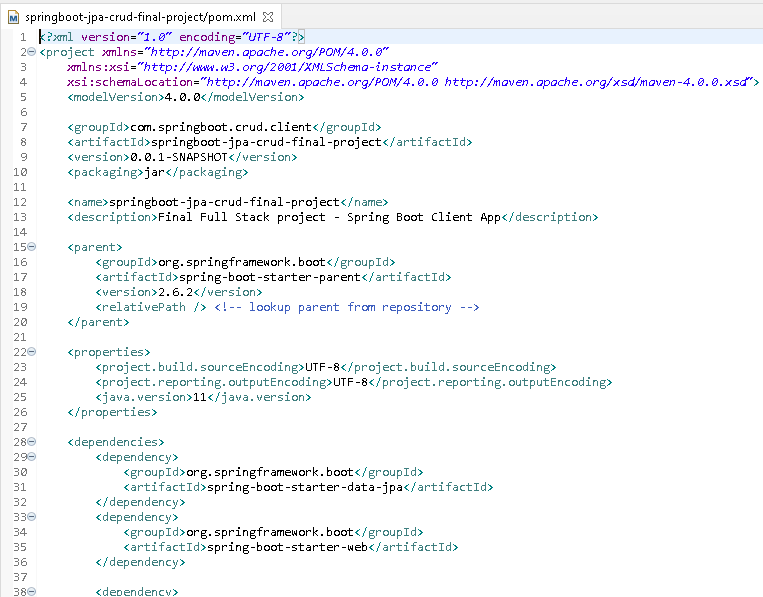
# **1. Creating SpringBoot project**

* ****1.1 Create a Spring project****

To create a spring boot project you can go use **[spring Initializer](https://start.spring.io/" \t "https://medium.com/analytics-vidhya/_blank)** or the Spring Starter project in Spring Tool Suite****.****You can begin with the default settings and generate a project and then unzip it and open in any IDE(preferrable Spring tool suite or Eclipse). Below is the image you can add the following dependency:

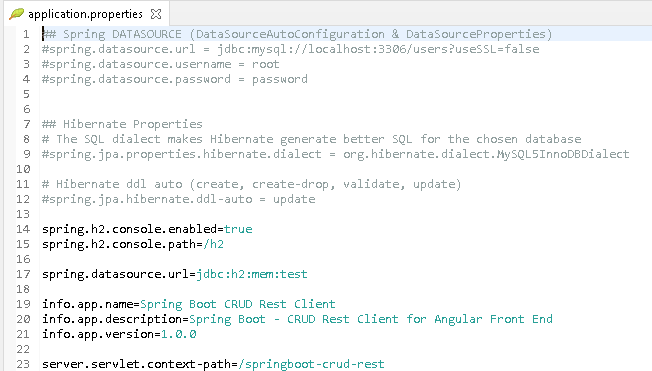
*   
  ****1.2 Update POM.xml****

Next you need to go to pom.xml in package structure and update the following dependencies as needed. You can use any version of Spring Boot 2.0 as long as your app compiles and executes properly.



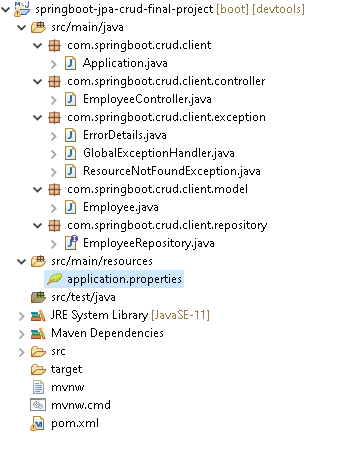
* ****1.3 Configure database properties****

Configure your database to store values



You can use MySQL by bringing in the MySQl connector or the H2 database as configured already.

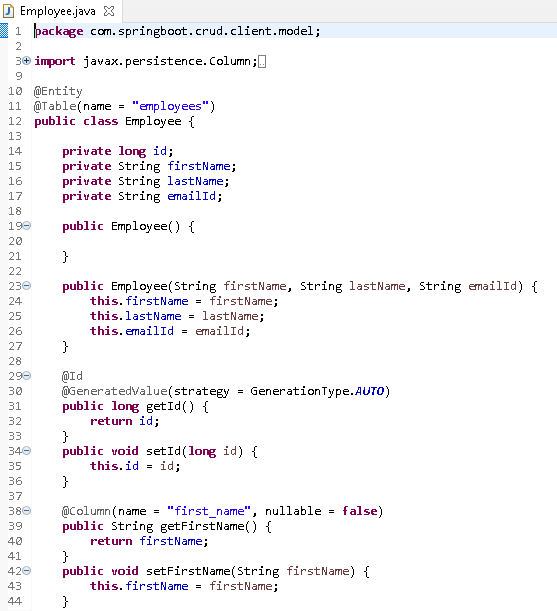
* ****1.3 Final Spring Boot Project structure****

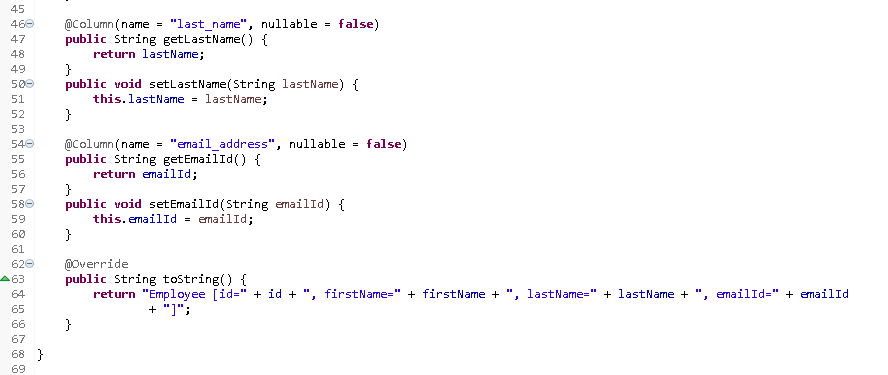


# **2. Creating Rest API using Spring Boot**

* Create employee
* Update Employee
* Delete Employee
* List Employee
* Get a single employee by its id
* ****2.1 Create the entity class (Employee.java)****

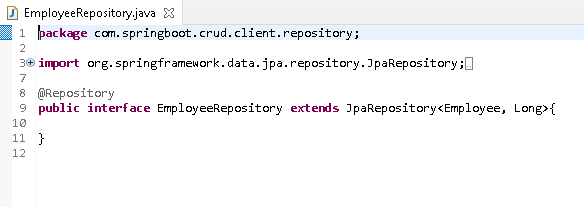
Here, we are creating an Entity/POJO (Plain Old Java Object) class.





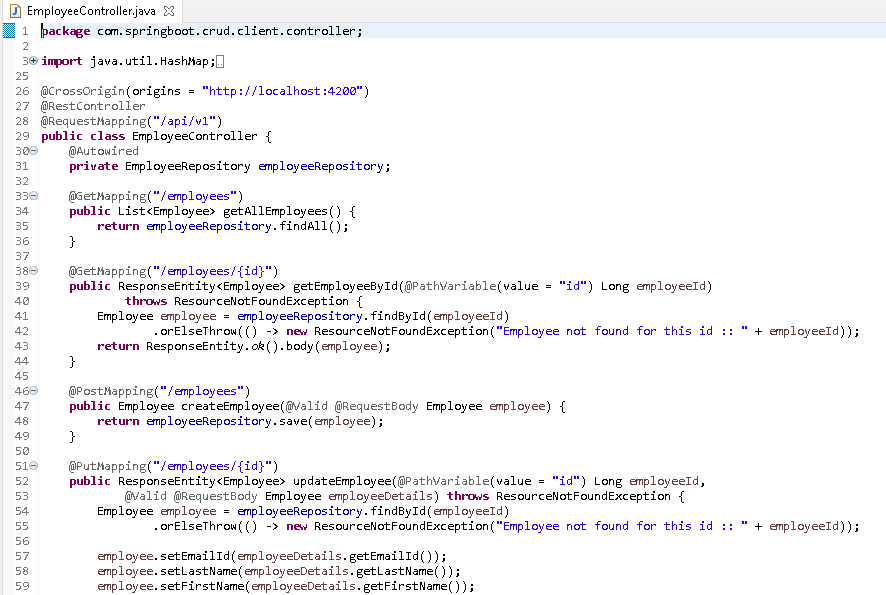
* ****2.2 Create the DAO interface(EmployeeRepository.java)****

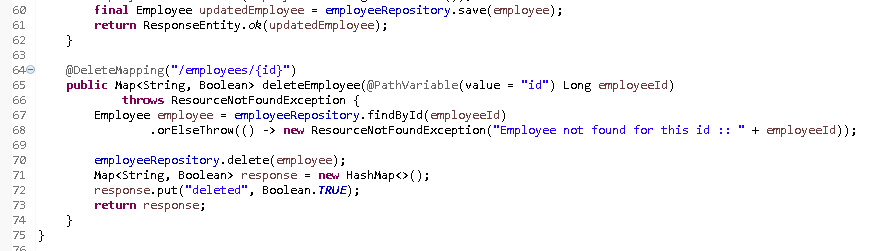
Here, we are creating the DAO interface to perform database related operations.



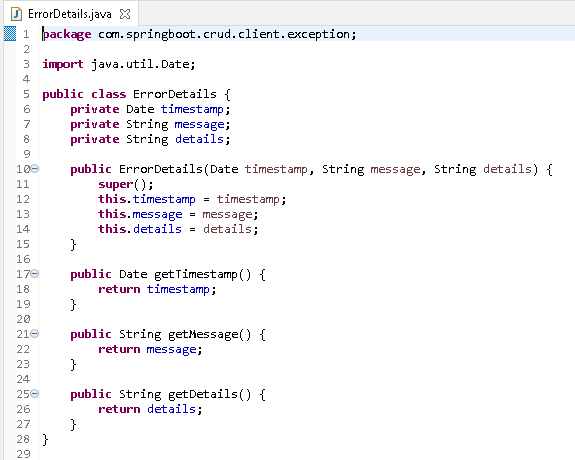
* ****2.3 Create the controller class****

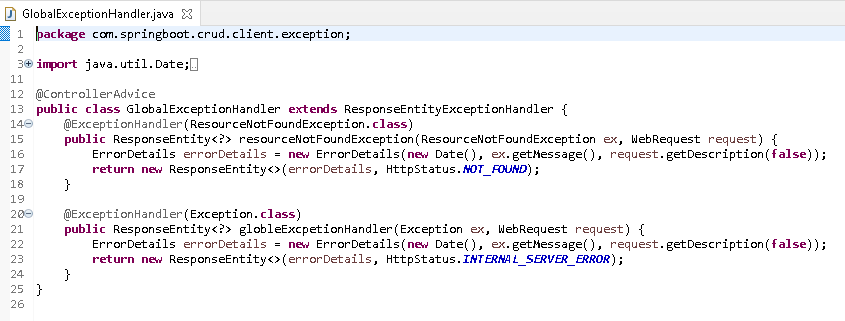
Here, we are creating the Controller class to make communication on a different path.

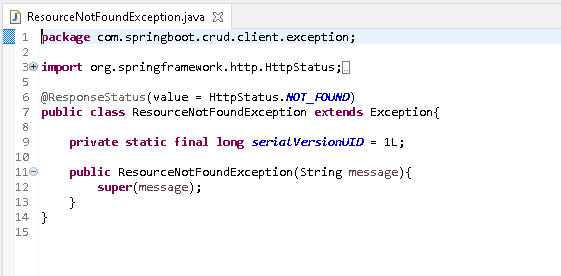




* ****2.4 Handling Errors****







# **3. Create an Angular App**

ng new angular14-springboot-client

So what we will add in angular app

## **Components:**

* create-employee
* employee-list
* employee-details

## **Service**

* employee.service.ts

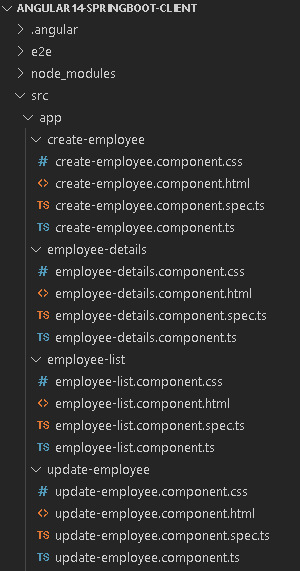
## **Model**

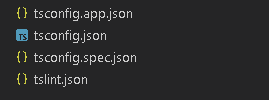
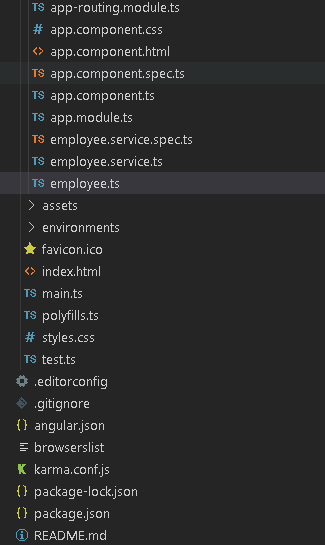
* employee.ts

**Note:** Refer to package.json and angular.json in project download folder if you have any issues with compiling or running the app.

jquery and bootstrap also need to be installed.

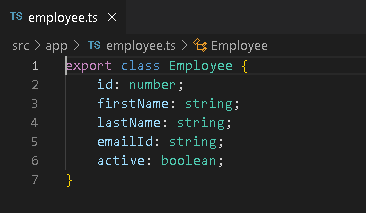
**Project Structure**



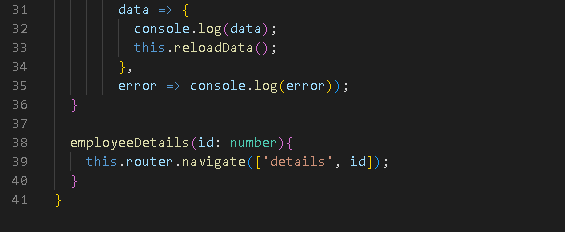


# **4. Creating Component, Service and Model files in angular**

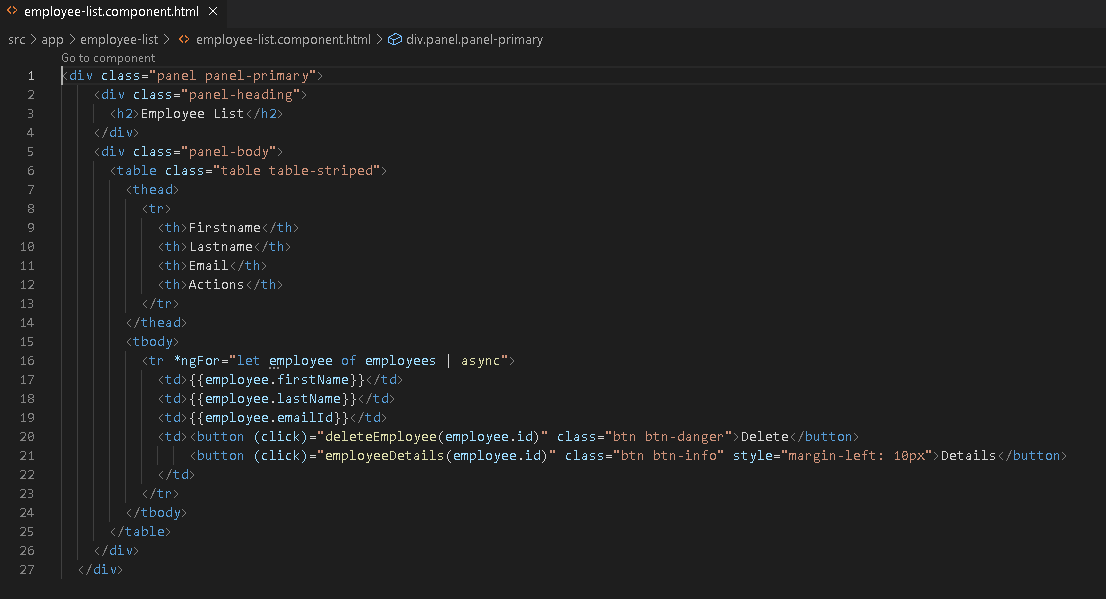
## ****4.1 Model Class(Employee.ts)****



## ****4.2 Employee List Component(employee-list.component.ts)****



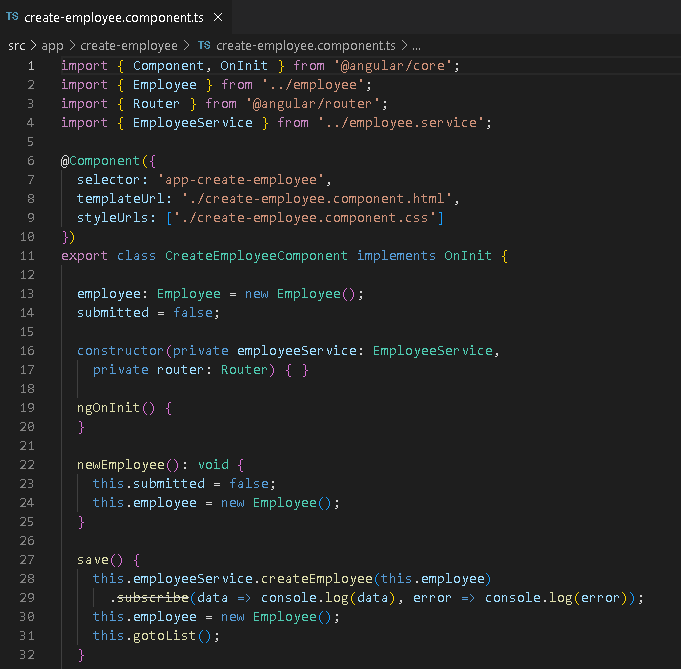
****employee-list-component.html****

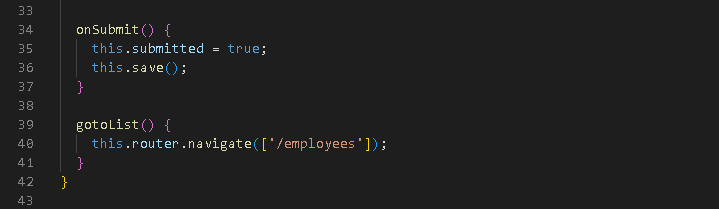


## ****4.3 Create Employe Component****

In this, we will write code for creating an employee.

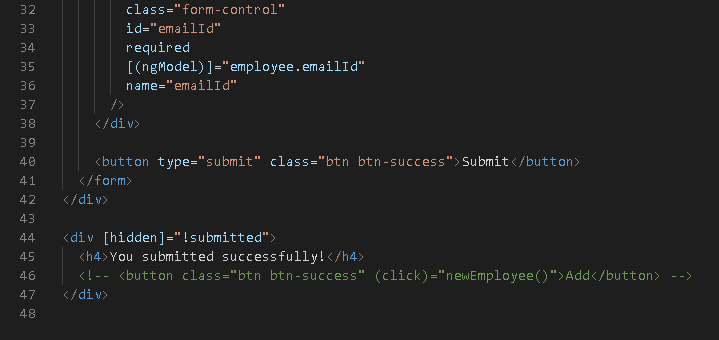
****create-employe.component.ts****





****create-employee.component.html****

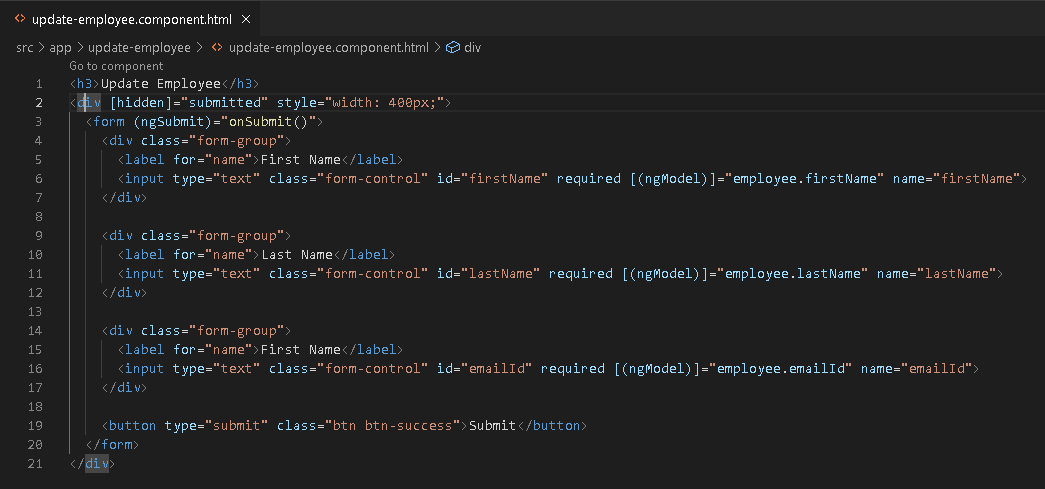




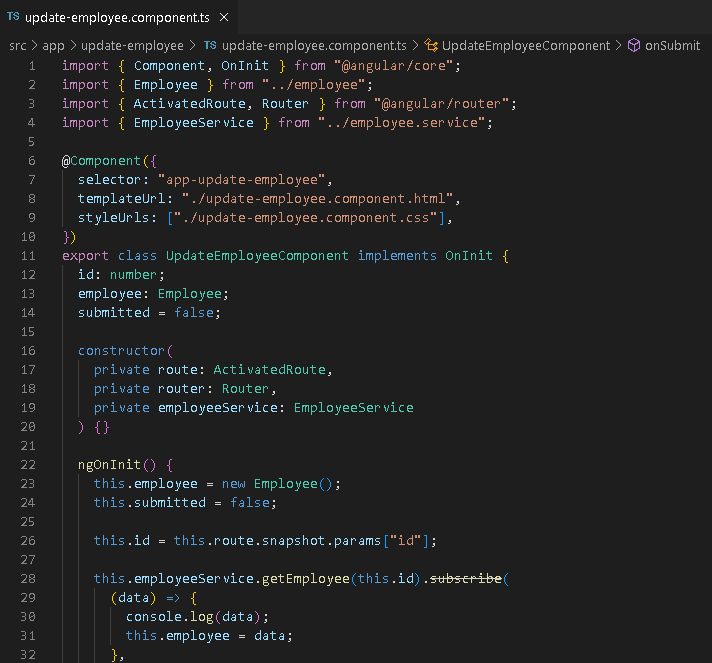
## ****4.4 Update Employee Component****

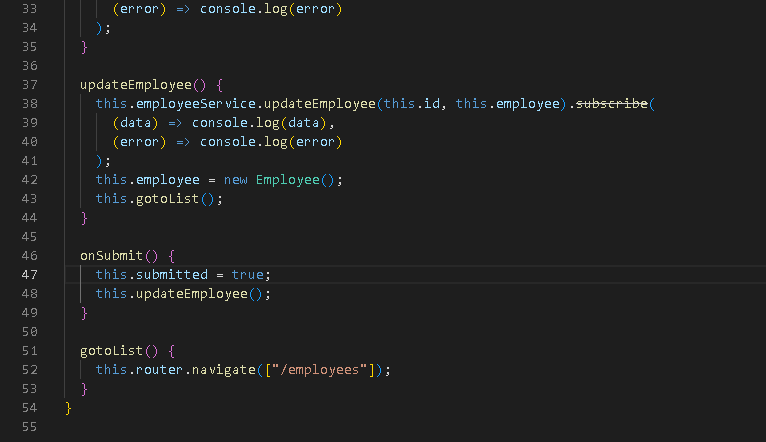
Here we will write angular code for updating component

update-employee.component.html



****update-component.ts****

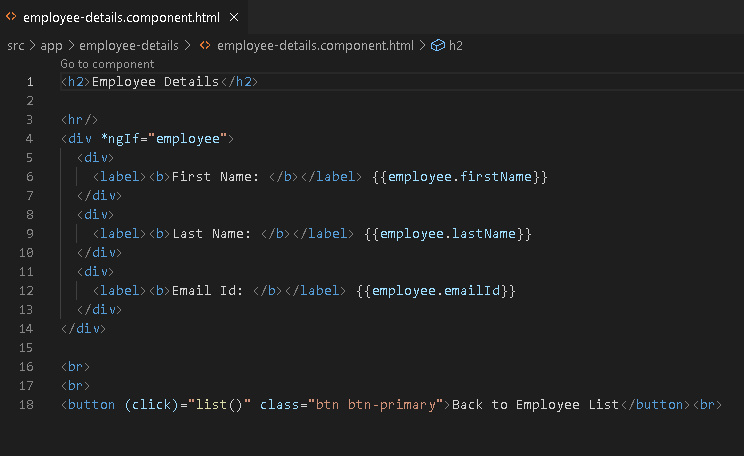




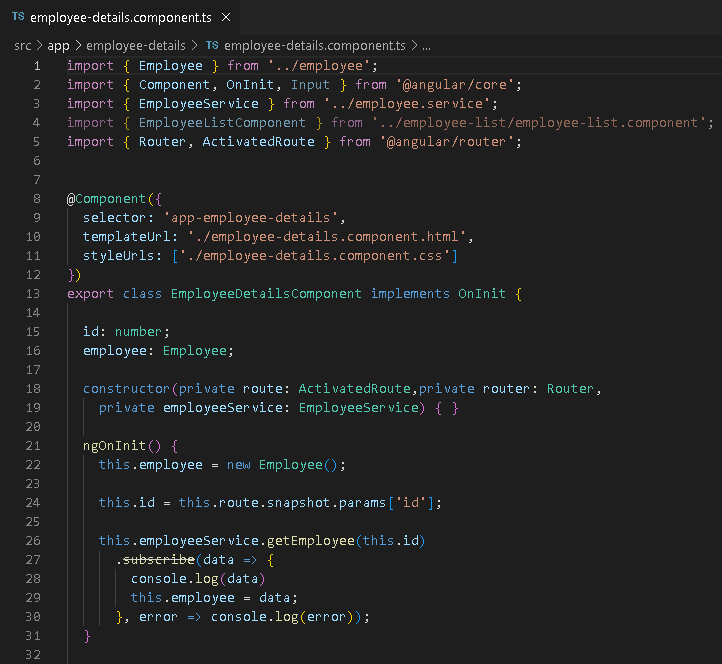
## **4.5 Employee Detail**

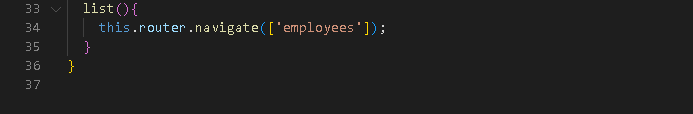
Here we will write angular code to show employee-detail .

****employee-detail.component.html****



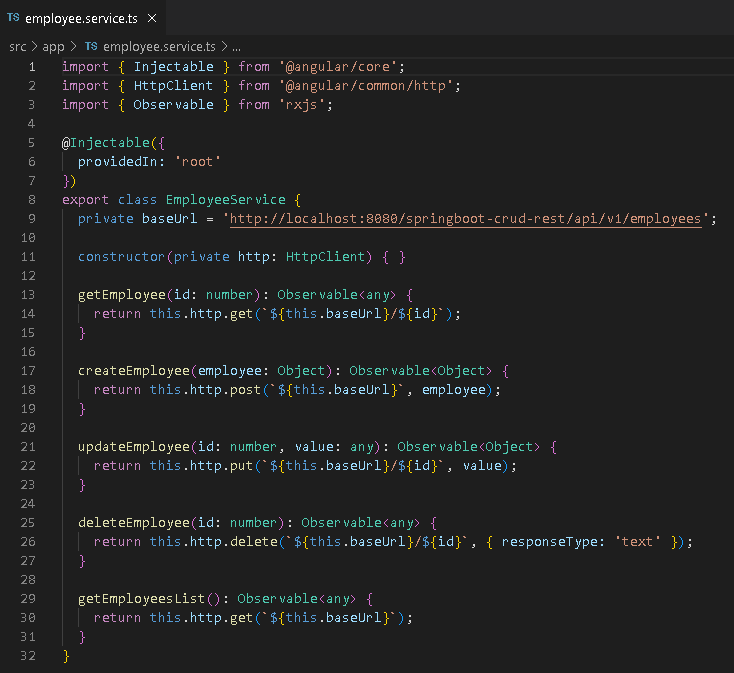
****employe-detail.component.ts****





# **Service**

Employee service will contain all the methods and provide data to all the components. The ****EmployeeService**** will be used to fetch the data from the Spring Boot local server (backend) .



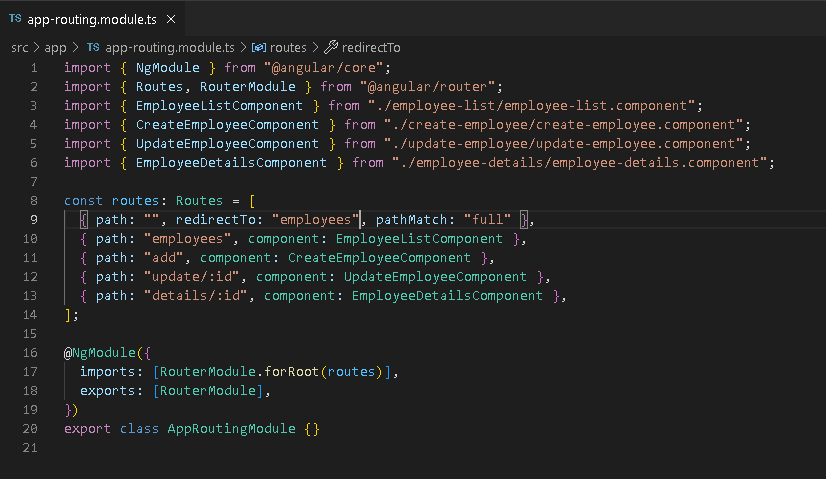
# **5 Configuring App to run**

Here in this section, we will be configuring app so that we can run it. First of all, we will add routing

## **5.1 Routing:**

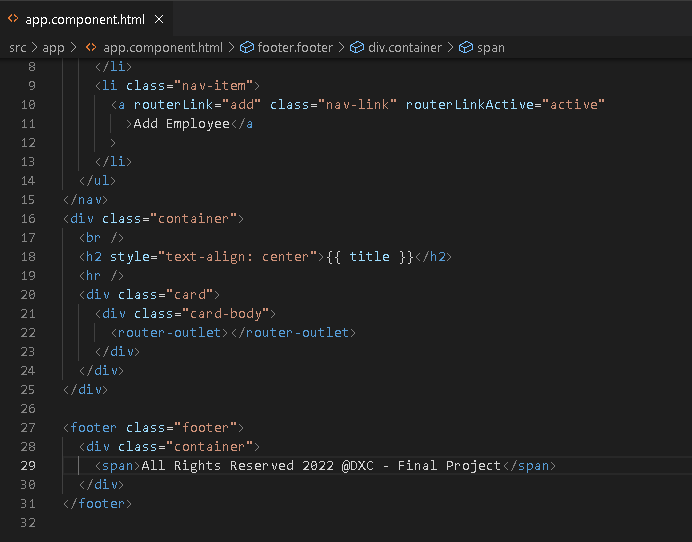
Routing is used to call or navigate different components in our Angular app.

****app.routing.module.ts****

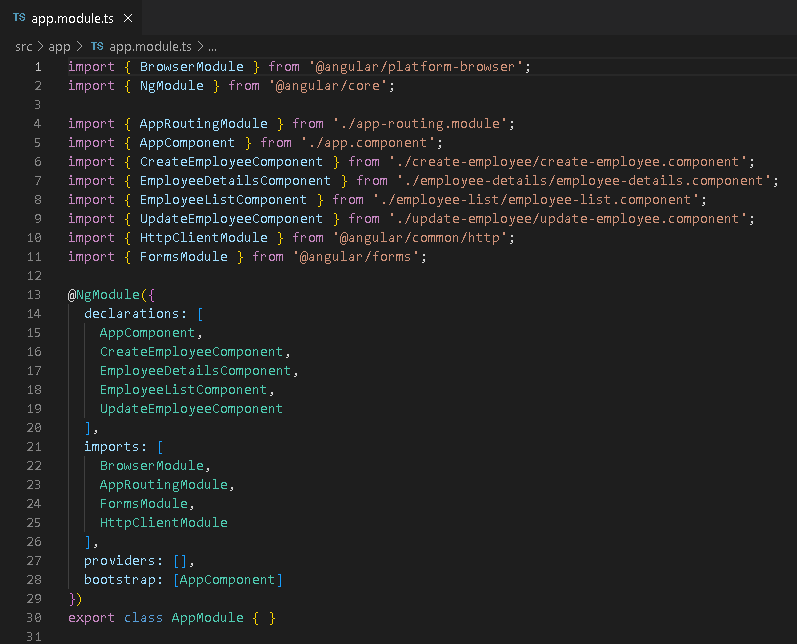


## ****5.2 AppComponent****

App component is the app-root file which is executed in index.html.To make our component accessible we need to add some code in app.component.html.



****app.module.ts****



# **6 Running the Anguler app and Spring Boot app**

****Step 1****: First of all run the spring boot app. Once you see the success message in the console of spring boot move to step 2

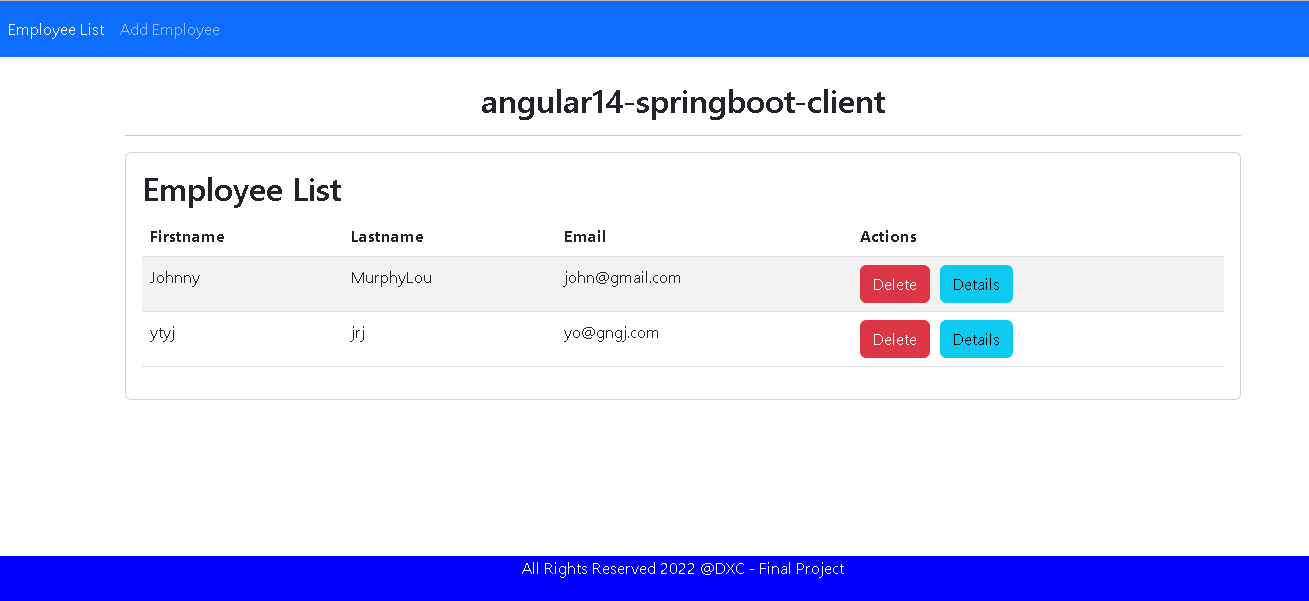
****Step 2****: Now run the angular app by running a command

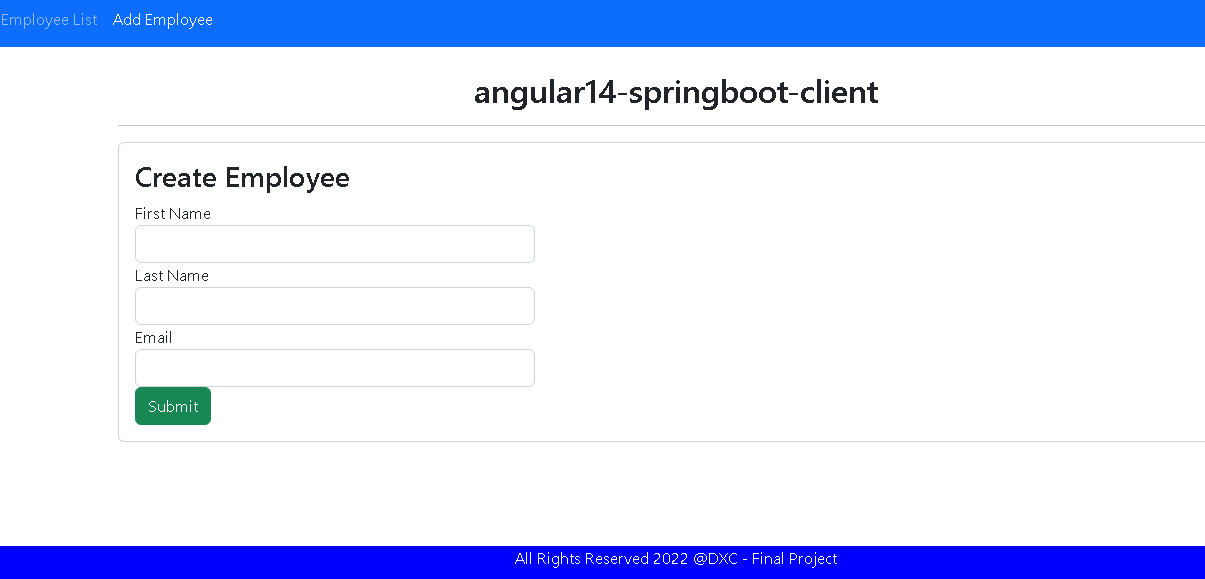
ng serve -o

If you find some port conflict issue you can manually specify port no

ng serve --port 5000

**Home Page**





Feel free to change the look and feel as you wish as well as the Title and/or page headings.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*