SpringAPI Integration With Angular 8

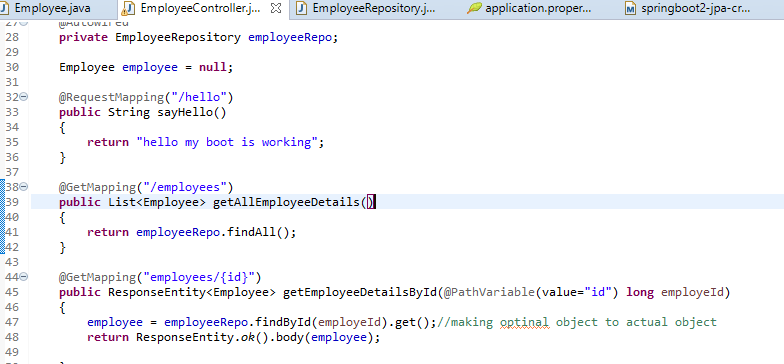
Create springbootapplication by

Chalanges:

During object creation for employee by employee id getting

**type mismatch: cannot convert from Optional<Employee> to Employee site:stackoverflow.com**

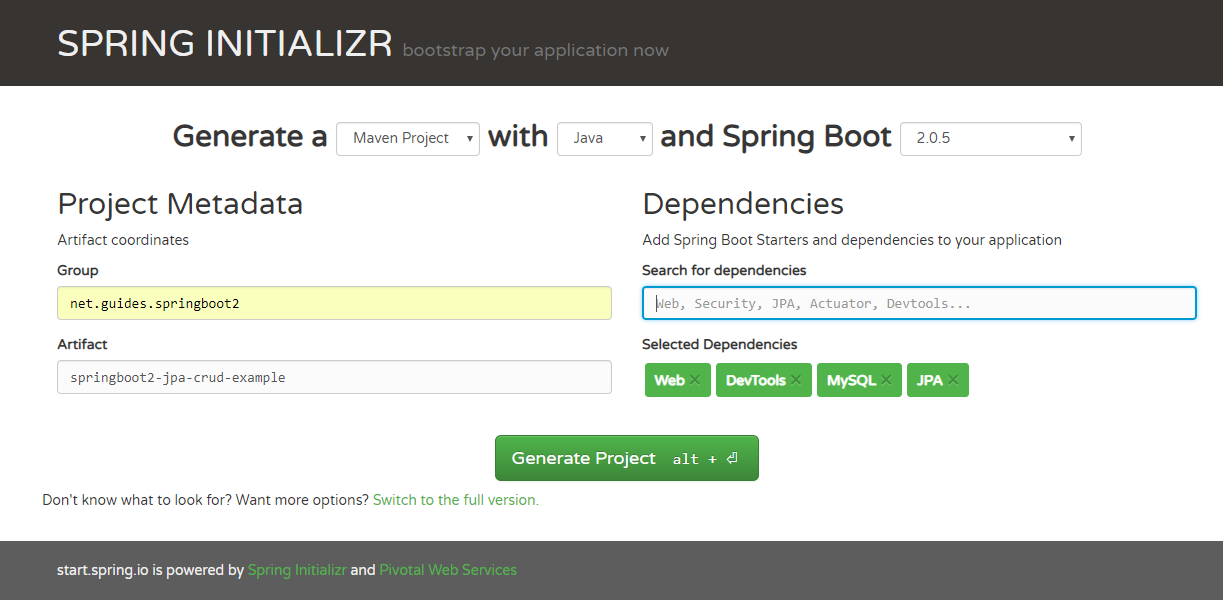
this error so we can make optional object to actuall object conatian by calling .get method on init



<https://start.spring.io/>

**1. Creating and Importing a Project**

There are many ways to create a Spring Boot application. The simplest way is to use Spring Initializrat [**http://start.spring.io/**](http://start.spring.io/), which is an online Spring Boot application generator.

**[](https://2.bp.blogspot.com/-1PSt3-IwbAE/W5tr3IUzS8I/AAAAAAAADyU/pgT0-em1wDo6ai2fTJ6NCGofOSU7VJ5yQCLcBGAs/s1600/create-project.PNG)**

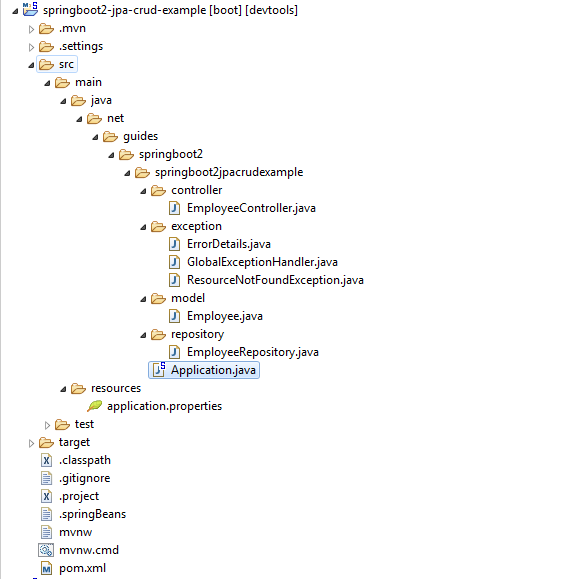
Look at the above diagram, we have specified the following details:

* **Generate**: Maven Project
* **Java Version**: 1.8 (Default)
* **Spring Boot**:2.0.4
* **Group**: net.guides.springboot2
* **Artifact**: springboot2-jpa-crud-example
* **Name**: springboot2-jpa-crud-example
* **Description**: Rest API for a Simple Employee Management Application
* **Package Name** : net.guides.springboot2.springboot2jpacrudexample
* **Packaging**: jar (This is the default value)
* **Dependencies**: Web, JPA, MySQL, DevTools

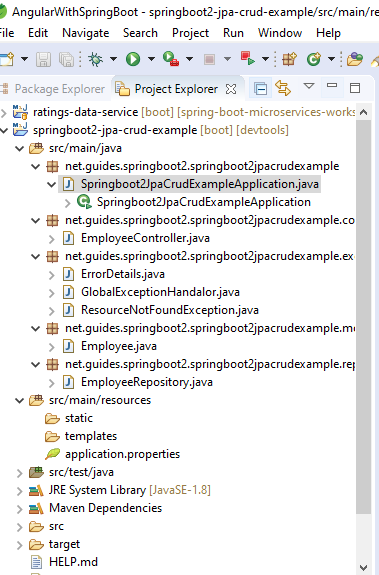
Once, all the details are entered, click on Generate Project button will generate a spring boot project and downloads it. Next, Unzip the downloaded zip file and import it into your favorite IDE.

**2. Packaging Structure**

Following is the packing structure of our Employee Management System -

**[](https://1.bp.blogspot.com/--r6LOKhuo1c/W5traXxjc5I/AAAAAAAADyQ/3Nttx0BtXF4toSUyAL5MbDgw14xGVXo1gCEwYBhgL/s1600/packaging-structure.PNG)**

My existing directory structure



1.Employee.java class

### 4. Create JPA Entity - Employee.java

package net.guides.springboot2.springboot2jpacrudexample.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "employees")

public class Employee {

private long id;

private String firstName;

private String lastName;

private String emailId;

public Employee() {

}

public Employee(String firstName, String lastName, String emailId) {

this.firstName = firstName;

this.lastName = lastName;

this.emailId = emailId;

}

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

@Column(name = "first\_name", nullable = false)

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

@Column(name = "last\_name", nullable = false)

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

@Column(name = "email\_address", nullable = false)

public String getEmailId() {

return emailId;

}

public void setEmailId(String emailId) {

this.emailId = emailId;

}

@Override

public String toString() {

return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", emailId=" + emailId

+ "]";

}

}

### Create a Spring Data Repository - EmployeeRepository.java

package net.guides.springboot2.springboot2jpacrudexample.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

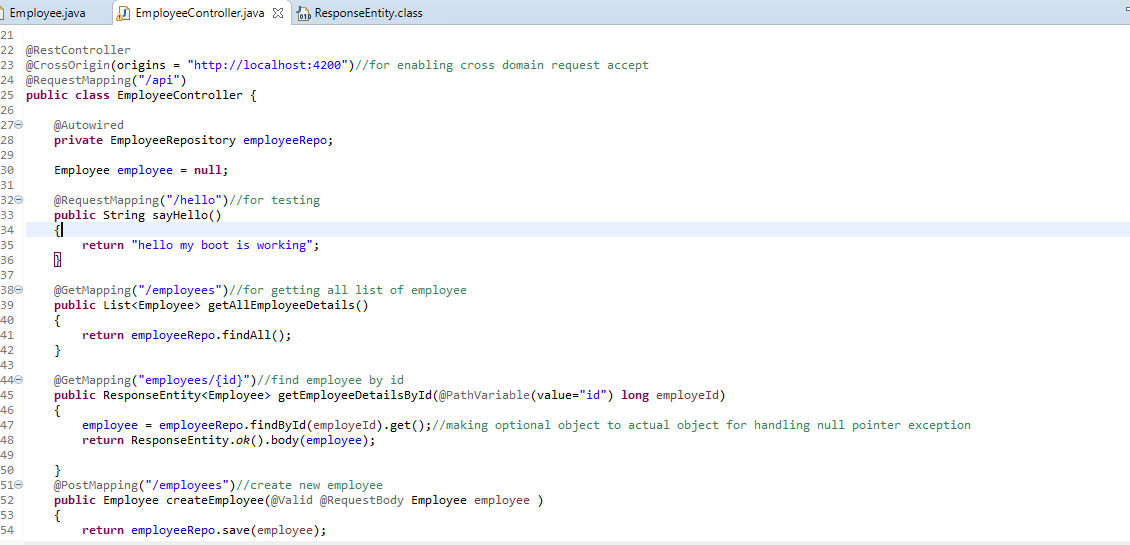
import net.guides.springboot2.springboot2jpacrudexample.model.Employee;

@Repository

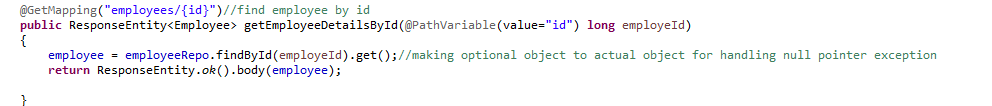
public interface EmployeeRepository extends JpaRepository<Employee, Long>{

}

Create employeeController for making crud request o api



Challenges get at



Full code from tutorial

### Create Spring Rest Controller - EmployeeController.java

package net.guides.springboot2.springboot2jpacrudexample.controller;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

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.PutMapping;

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

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

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

import net.guides.springboot2.springboot2jpacrudexample.exception.ResourceNotFoundException;

import net.guides.springboot2.springboot2jpacrudexample.model.Employee;

import net.guides.springboot2.springboot2jpacrudexample.repository.EmployeeRepository;

@RestController

@RequestMapping("/api/v1")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@GetMapping("/employees/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable(value = "id") Long employeeId)

throws ResourceNotFoundException {

Employee employee = employeeRepository.findById(employeeId)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found for this id :: " + employeeId));

return ResponseEntity.ok().body(employee);

}

@PostMapping("/employees")

public Employee createEmployee(@Valid @RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@PutMapping("/employees/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable(value = "id") Long employeeId,

@Valid @RequestBody Employee employeeDetails) throws ResourceNotFoundException {

Employee employee = employeeRepository.findById(employeeId)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found for this id :: " + employeeId));

employee.setEmailId(employeeDetails.getEmailId());

employee.setLastName(employeeDetails.getLastName());

employee.setFirstName(employeeDetails.getFirstName());

final Employee updatedEmployee = employeeRepository.save(employee);

return ResponseEntity.ok(updatedEmployee);

}

@DeleteMapping("/employees/{id}")

public Map<String, Boolean> deleteEmployee(@PathVariable(value = "id") Long employeeId)

throws ResourceNotFoundException {

Employee employee = employeeRepository.findById(employeeId)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found for this id :: " + employeeId));

employeeRepository.delete(employee);

Map<String, Boolean> response = new HashMap<>();

response.put("deleted", Boolean.TRUE);

return response;

}

}

## Enable CORS on the Server

## To enable CORS on the server, add a *@CrossOrigin* annotation to the *EmployeeController.*

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

@RestController

@RequestMapping("/api/v1")

public class EmployeeController {

// ....

}

Note : Checking on Postman this all developed api we are getting

**invalidCors 403** Issue so for that we have to comment this

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

**Now generate spingBoot Client with angular 8**

**open a VS code in a folder (Angular\_EmployeeDetails)**

**(a)ng new angular8-springboot-client**

1. **Components**

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

1. **Services**

* employee.service.ts - Service for Http Client methods

1. **Modules**

* FormsModule
* HttpClientModule
* AppRoutingModule.

1. **Employee Class (Typescript class)**

* employee.ts: class Employee (id, firstName, lastName, emailId)

**(b)generate all component by following commands in app folder**

**– ng g c create-employee**

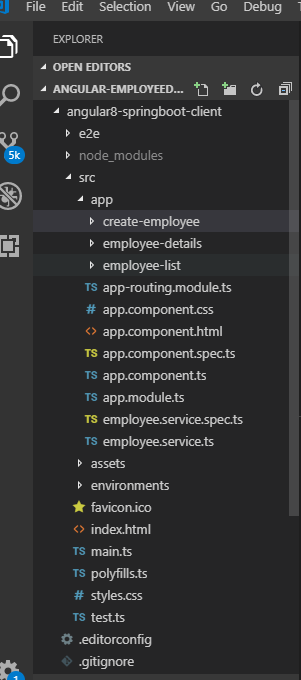
**– ng g c employee-details**

**– ng g c employee-list**

**(c)generate service by following commands in app folder**

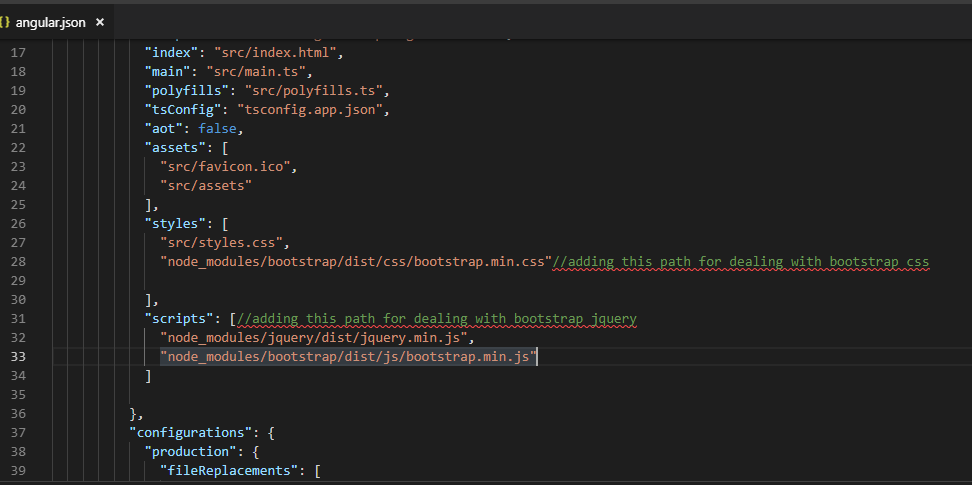
**- ng g s employee**

**Hence the directory structure as follows:**



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

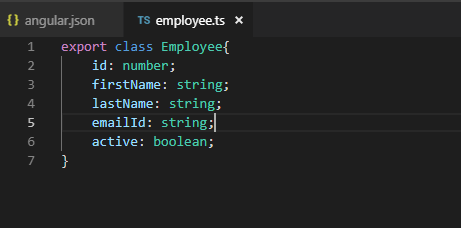
**npm install bootstrap jquery --save**



**(f)Create Employee class 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 –

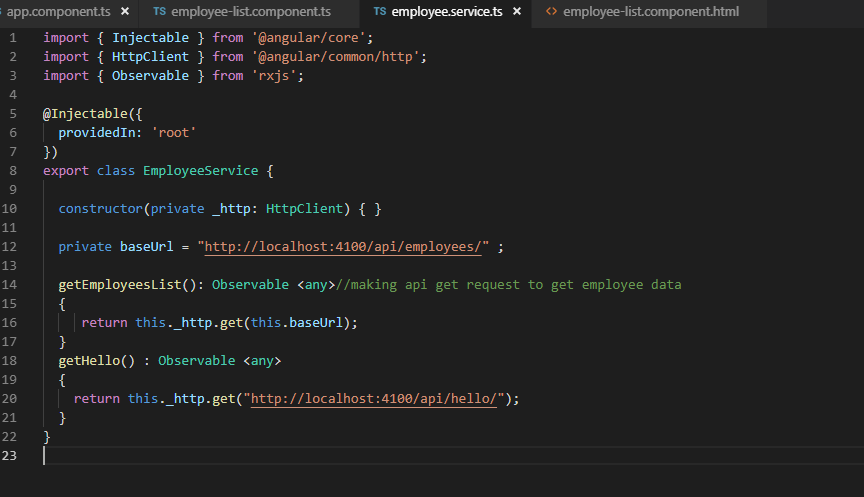
For make api response field compatible with this feild

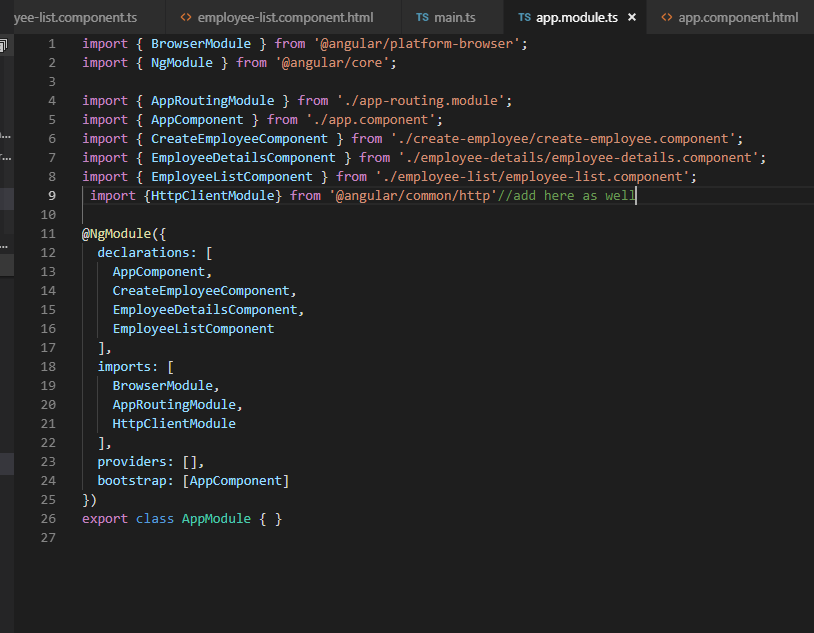


**(g)EmployeeService - employee-service.ts**

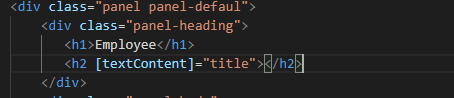
The *EmployeeService* will be used to get the data from backend by calling spring boot APIs. Update the *employee.service.ts* file inside *src/app* directory with the following code to it –

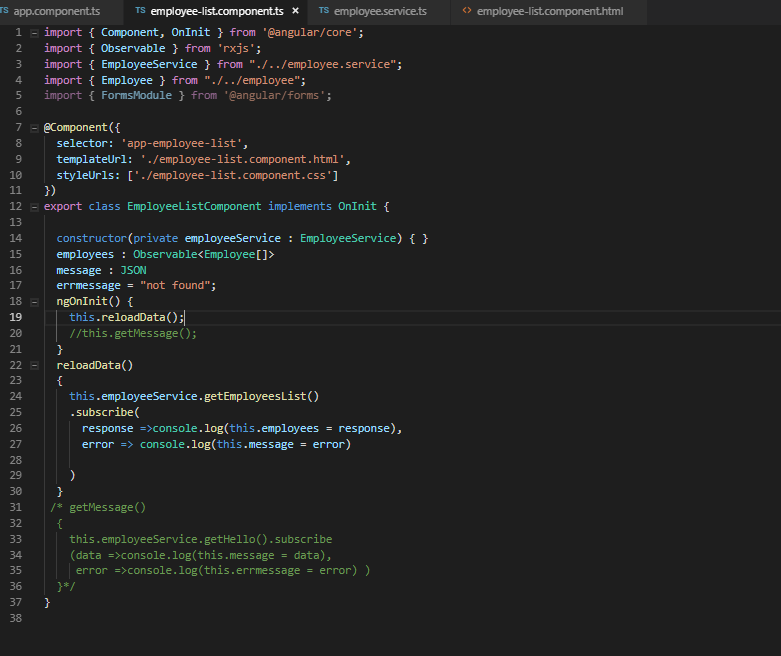
1.for making get request to get employee data from the spring boot api

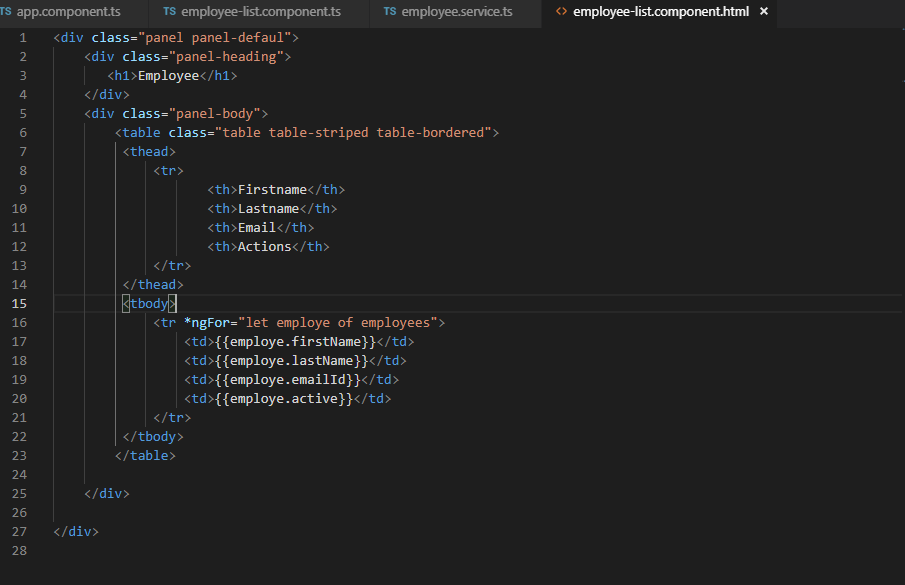


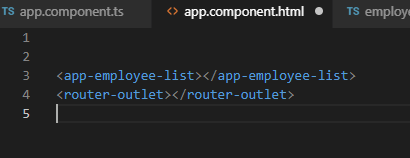


In <h1> tag binding

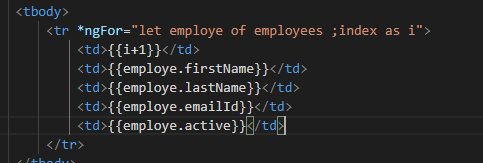




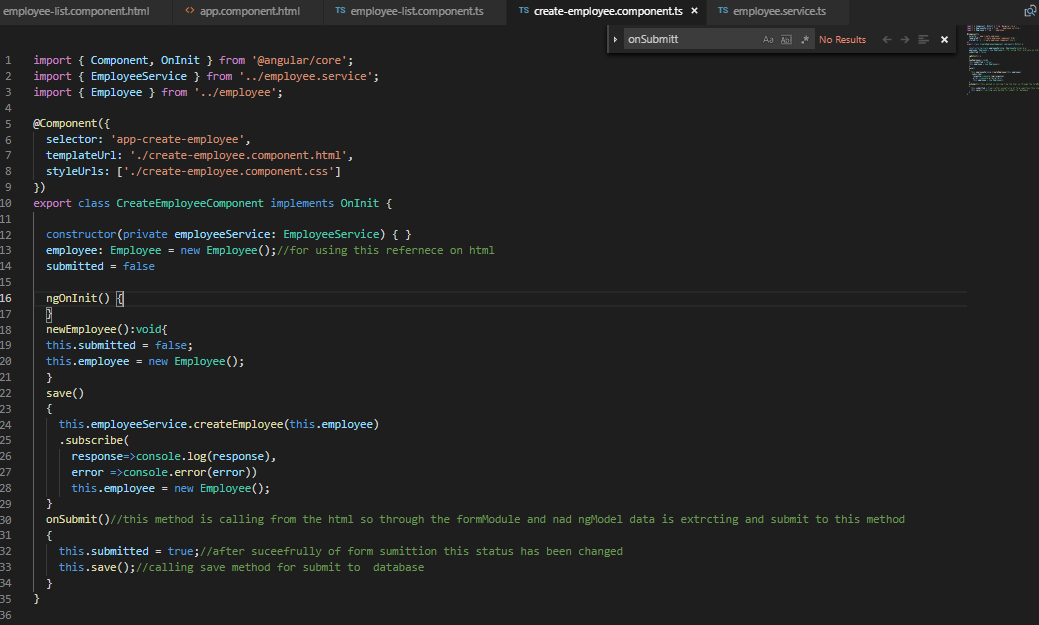


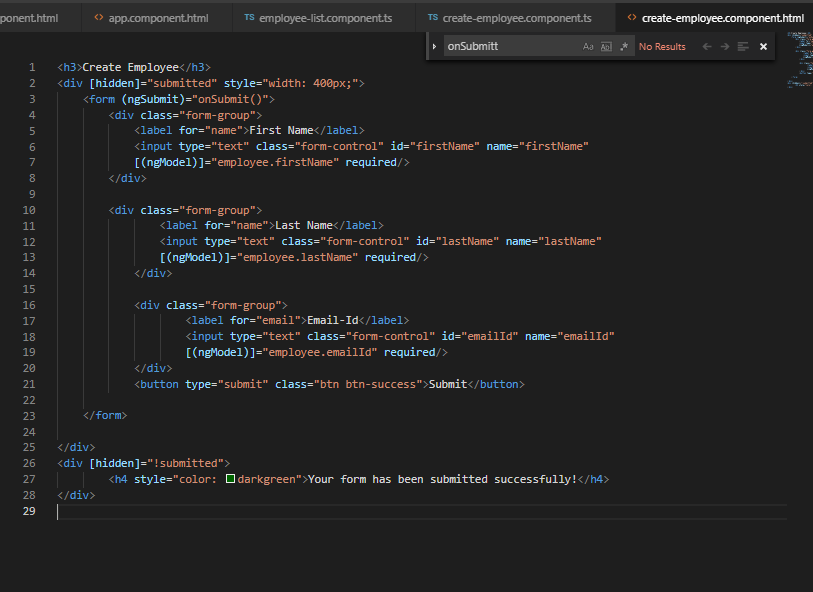


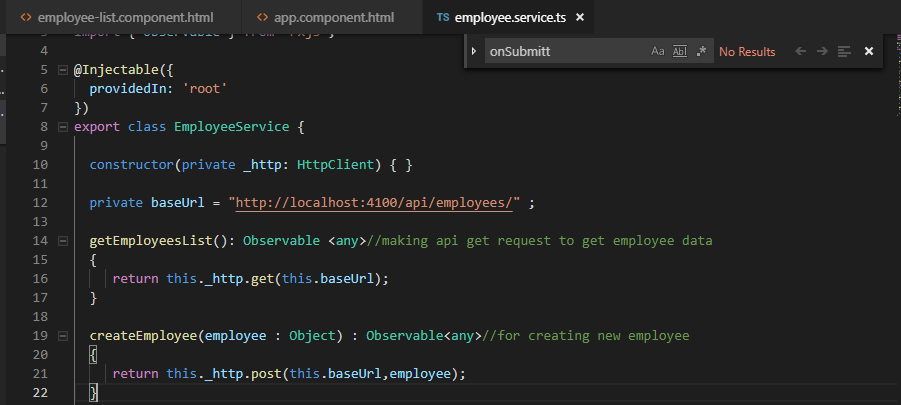
**We can provide indexing from 1 as follows**



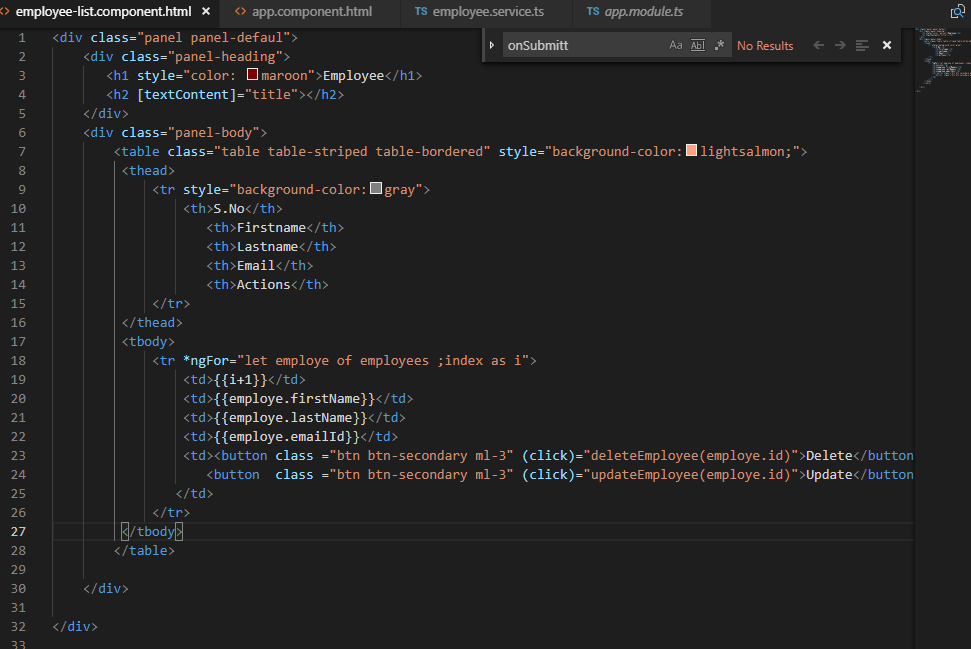
**Now for creating new employee**

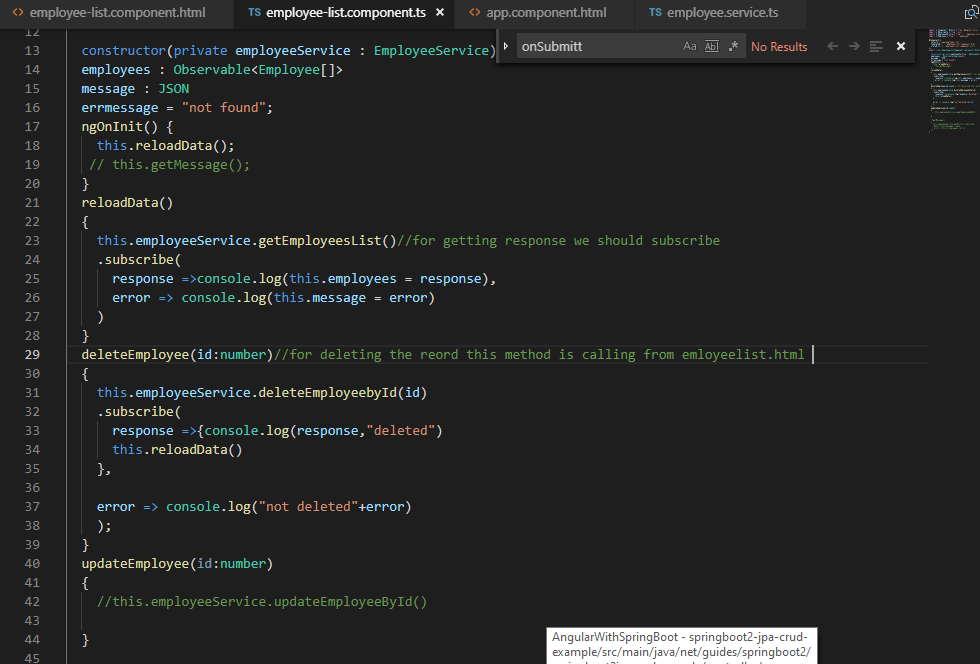


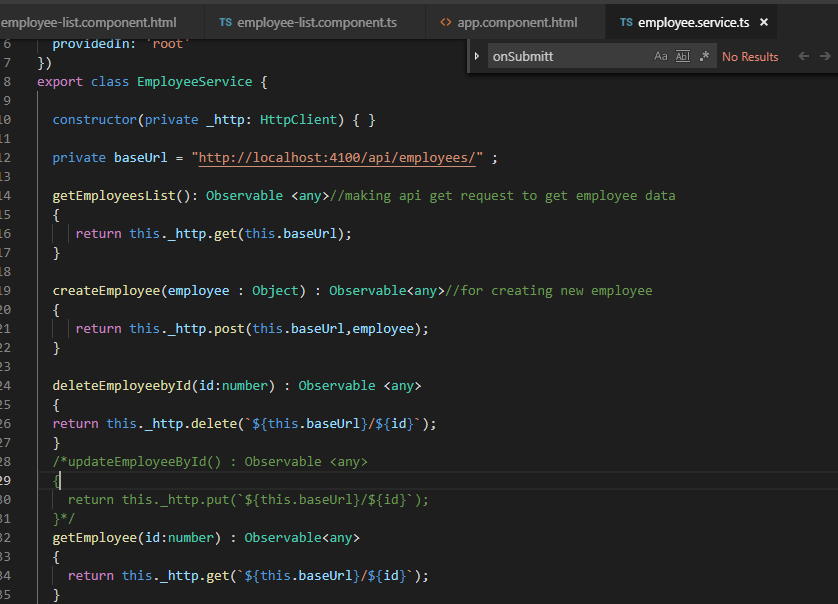




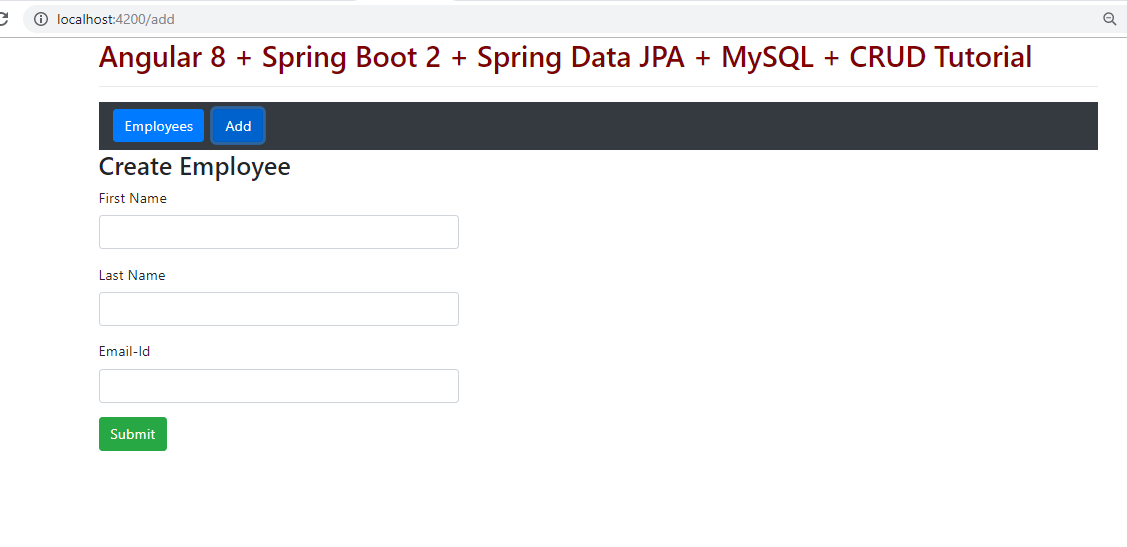
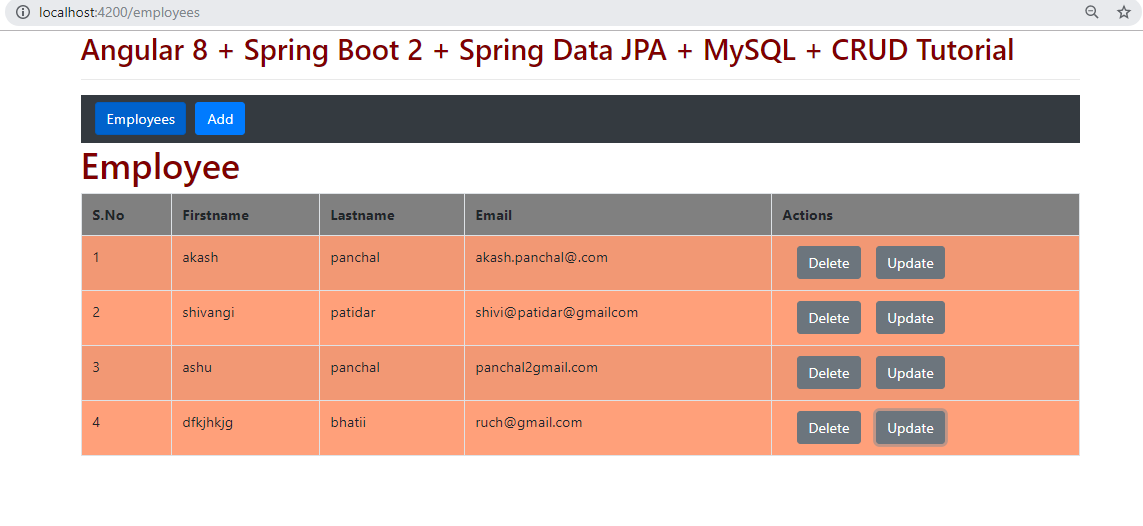
**For deleting the record**







**Output**



**For showing the toaster or we can say popup message for any operation we have to perform following steps**

<https://www.npmjs.com/package/toastr>

<https://www.npmjs.com/package/ngx-toastr>

npm install ngx-toastr –save

@angular/animations package is a required dependency for the default toast

npm install @angular/animations –save

//for updated angular version this dependency is already there

If you are using angular-cli you can add it to your angular.json

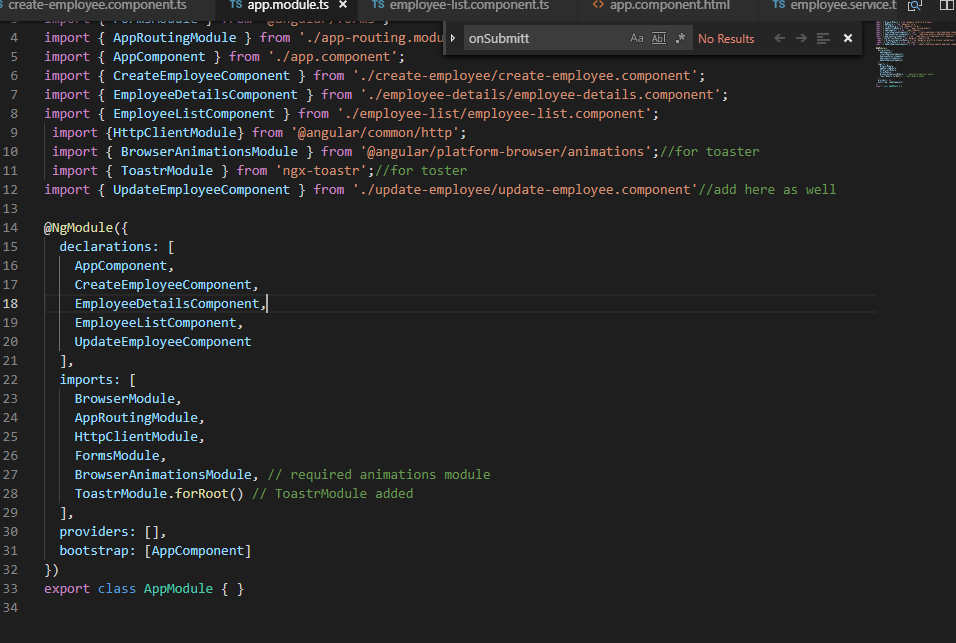
"styles": [

  "styles.scss",

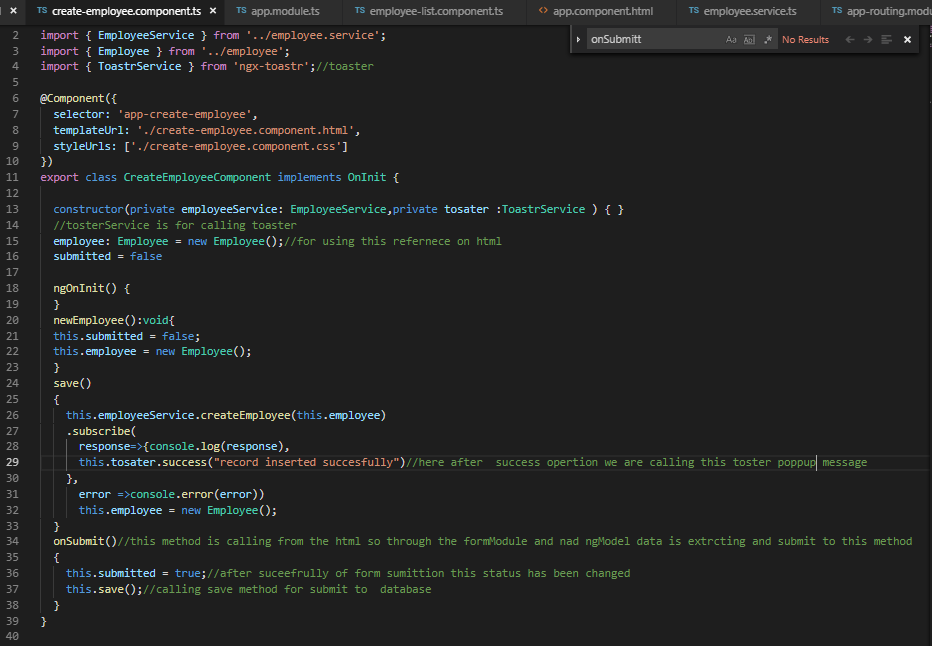
  "node\_modules/ngx-toastr/toastr.css" *// try adding '../' if you're using angular cli before 6*

]

App.,module.ts added module for toaster enableing



**In compontent.ts class where we wanted to call toaster**



**import** { ToastrService } **from** 'ngx-toastr';

@Component({**...**})

**export** class YourComponent {

  constructor(private toastr**:** ToastrService) {}

  showSuccess() {

    this.toastr.success('Hello world!', 'Toastr fun!');

  }

}