- by Mr.RAGHU [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

Spring Boot 2.x Angular 9.x - RAGHU SIR

Code and SETUP Full EXAMPLE Setup for Angular

FB: https://www.facebook.com/groups/thejavatemple/

Spring Boot code:

1. pom.xml

```
<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<optional>true</optional>
		</dependency>
		<dependency>
```

```
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.46</version><!--$NO-MVN-MAN-VER$-->
        </dependency>
```

2. application.properties

```
server.port=9898
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL55Dialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update

server.servlet.context-path=/springboot-crud-rest
```

3. Model class

```
package in.nit.raghu.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@NoArgsConstructor
@RequiredArgsConstructor
@AllArgsConstructor
@Entity
public class Student {
	@Id
	@GeneratedValue
	private Integer stdId;
	@NonNull
	private String stdName;
	@NonNull
	private Double stdFee;
	@NonNull
	private String stdCourse;
}
-------------------
package in.nit.raghu.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Message {
	private String type;
	private String message;
}
```

4. Repository Interface

```
package in.nit.raghu.repo;

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

import in.nit.raghu.model.Student;

public interface StudentRepository
	extends JpaRepository<Student,Integer>
{

}
```

5. Service Interface

```
package in.nit.raghu.service;

import java.util.List;
import java.util.Optional;

import in.nit.raghu.model.Student;

public interface IStudentService {

	public Integer saveStudent(Student s);
	public List<Student> getAllStudents();
	public Optional<Student> getOneStudent(Integer id);
	public boolean isExist(Integer id);
	public void deleteStudent(Integer id);
}
```

6. ServiceImpl class

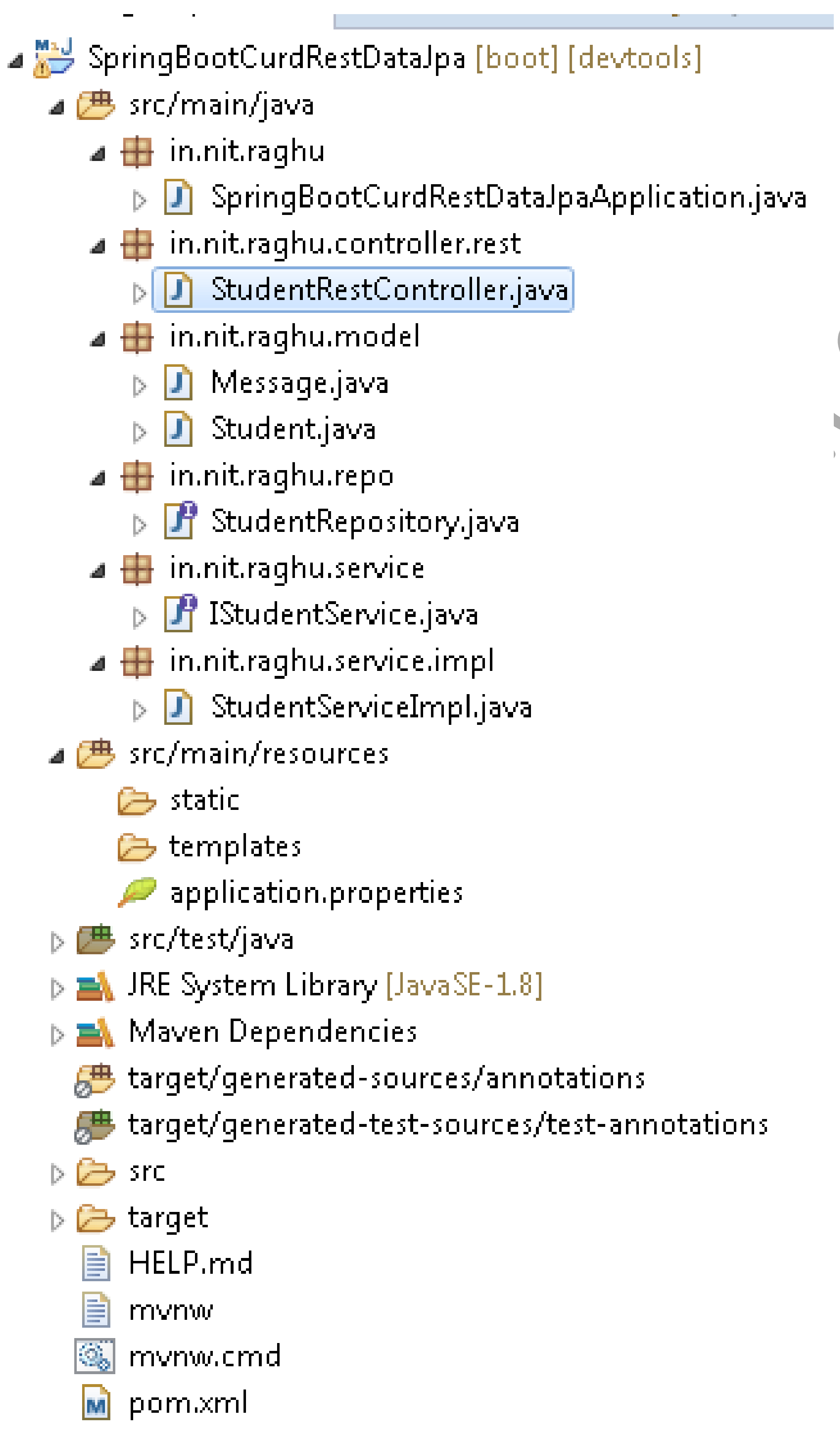
```
package in.nit.raghu.service.impl;

import java.util.List;
import java.util.Optional;

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

import in.nit.raghu.model.Student;
import in.nit.raghu.repo.StudentRepository;
import in.nit.raghu.service.IStudentService;

@Service
public class StudentServiceImpl
	implements IStudentService
{
	@Autowired
	private StudentRepository repo; //HAS-A

	@Override
	public Integer saveStudent(Student s) {
		return repo.save(s).getStdId();
	}

	@Override
	public List<Student> getAllStudents() {
		return repo.findAll();
	}

	@Override
	public Optional<Student> getOneStudent(Integer id) {
		return repo.findById(id);
	}

	@Override
	public void deleteStudent(Integer id) {
		repo.deleteById(id);
	}

	@Override
	public boolean isExist(Integer id) {
		return repo.existsById(id);
	}

}
```

7. RestController

```
package in.nit.raghu.controller.rest;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import in.nit.raghu.model.Message;
import in.nit.raghu.model.Student;
import in.nit.raghu.service.IStudentService;

@RestController
@CrossOrigin(origins = "*")
@RequestMapping("/rest/student")
public class StudentRestController {

	@Autowired
	private IStudentService service;

	/**
	 * 1. This method takes Student object
	 *    as input from JSON/XML using
	 *    @RequestBody and returns
	 *    ResponseEntity<T>.
	 *    call service.saveStudent(ob)
	 */
	@PostMapping("/save")
	public ResponseEntity<Message> saveStudent(
			@RequestBody Student student)
	{
		ResponseEntity<Message> resp=null;
		try {
			Integer id=service.saveStudent(student);
			resp=new ResponseEntity<Message>(new 
Message("SUCCESS",id+"-saved"),HttpStatus.OK);
		} catch (Exception e) {
			resp=new ResponseEntity<Message>(new 
Message("FAIL","Unable to Save"),HttpStatus.OK);
			e.printStackTrace();
		}
		return resp;
	}

	/***
	 * 2. This method reads data from DB
	 *  using findAll() and returns
	 *  List<Student> if data exist
	 *  or String (not exist)
	 *  as ResponseEntity using annotation
	 *  @ResponseBody
	 */
	@GetMapping("/all")
	public ResponseEntity<?> getAllStudents(){
		ResponseEntity<?> resp=null;
		try {
			List<Student> list=service.getAllStudents();
			if(list!=null && !list.isEmpty())
				resp=new 
ResponseEntity<List<Student>>(list,HttpStatus.OK);
			else
				resp=new ResponseEntity<String>("No Data 
Found",HttpStatus.OK);
		} catch (Exception e) {
```

```
                    resp=new ResponseEntity<String>("Unable to fetch
Data",HttpStatus.INTERNAL_SERVER_ERROR);
                    e.printStackTrace();
            }

            return resp;
        }

        /**
         * 3. Read PathVariable id (as input)
         * use service layer to find one object
         *  based on Id. Return Student if exist
         *  else String (error message) as
         *  ResponseEntity<?>
         */
        @GetMapping("/one/{id}")
        public ResponseEntity<?> getOneStudent(
                    @PathVariable Integer id)
        {
            ResponseEntity<?> resp=null;
            try {
                    Optional<Student> opt=service.getOneStudent(id);
                    if(opt.isPresent())
                            resp=new
ResponseEntity<Student>(opt.get(),HttpStatus.OK);
                    else
                            resp=new ResponseEntity<String>("No Data
Found",HttpStatus.BAD_REQUEST);
            } catch (Exception e) {
                    resp=new ResponseEntity<String>("Unable to Fetch
Data",HttpStatus.INTERNAL_SERVER_ERROR);
                    e.printStackTrace();
            }
```

```
		return resp;
	}

	/**
	 * 4. Read pathVariable id
	 * check row exist or not
	 * if exist call service delete
	 * else return String error msg
	 */
	@DeleteMapping("/remove/{id}")
	public ResponseEntity<Message> deleteStudent(
			@PathVariable Integer id)
	{
		System.out.println("welcome");
		ResponseEntity<Message> resp=null;
		try {
			boolean exist=service.isExist(id);
			if(exist) {
				service.deleteStudent(id);
				resp=new ResponseEntity<Message>(new
Message("SUCCESSS",id+"-removed"),HttpStatus.OK);
			}else {
				resp=new ResponseEntity<Message>(new
Message("FAIL",id+"-Not Exist"),HttpStatus.BAD_REQUEST);
			}
		} catch (Exception e) {
			resp=new ResponseEntity<Message>(new
Message("FAIL","Unable to
Delete"),HttpStatus.INTERNAL_SERVER_ERROR);
			e.printStackTrace();
		}

		return resp;
	}

	/**
	 * 5. Read Input as JSON/XML using
	 * @RequestBody , check id exist or not
	 * if exist call service save method
	 * Return ResponseeEntity
	 */
	@PutMapping("/update")
	public ResponseEntity<String> updateStudent(
			@RequestBody Student student)
	{
		ResponseEntity<String> resp=null;
		try {
			boolean exist=service.isExist(student.getStdId());
			if(exist) {
				service.saveStudent(student);
				resp=new
ResponseEntity<String>(student.getStdId()+"-Updated",HttpStatus.OK);
			}else {
				resp=new
ResponseEntity<String>(student.getStdId()+"-Not
Exist",HttpStatus.BAD_REQUEST);
			}
		} catch (Exception e) {
			resp=new ResponseEntity<String>("Unable to
Update",HttpStatus.INTERNAL_SERVER_ERROR);
			e.printStackTrace();
		}
		return resp;
	}
}
```



```
SpringBootCurdRestDataJpa [boot] [devtools]
  src/main/java
    in.nit.raghu
      SpringBootCurdRestDataJpaApplication.java
    in.nit.raghu.controller.rest
      StudentRestController.java
    in.nit.raghu.model
      Message.java
      Student.java
    in.nit.raghu.repo
      StudentRepository.java
    in.nit.raghu.service
      IStudentService.java
    in.nit.raghu.service.impl
      StudentServiceImpl.java
  src/main/resources
    static
    templates
    application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

Angular Setup

#1 Download and Install Node JS:
  https://nodejs.org/en/download/

> Click on OS Option(Ex: Windows)
> It will be downloaded as setup

> Double click on setup file > next > Next > Finsih

#2 Check installtion of Node using cmd prompt

```
C:\Users\nareshit> node -v
v12.16.3

C:\Users\nareshit> npm -v
6.14.4
```

#2 Install Angular (wait for : 10 mins to 1 hr after cmd)
 Open cmd prompt and type command like

```
> npm install -g @angular/cli
```

#3 Check angular installtion using cmd

```
> ng version
```

#4 Download Visual Studio Code Software and install

Goto : https://code.visualstudio.com/download
Click on OS Option (Ex: Windows)

> Double click on setup file > next > next > Finish

#5 Open VS Code Editor
> File > open folder > create new folder (ex: myangapps) > Open

> press ctrl+` (before to 1 Key)

#6 create new project (using terminal)
> ng new boot-student-app
> press Y for Routing and click enter even for CSS

WAIT for 15Min to 1Hr

#7 Switch to App Folder
> cd boot-student-app (press enter)

#8 Start app
> ng serve --open

WAIT for 15Min to 1Hr

#9 Open browser and Enter
  http://localhost:4200/

Project Code

Step#1 Project creation with files

```
npm install -g @angular/cli
ng new angular-springboot-student

cd angular-springboot-student

ng g class student  --spec false
ng g s student  --spec false
ng g c create-student
ng g c student-list
```

Step#2 Model class code

stdId, stdName, stdFee, stdCourse in student.ts

```
export class Student {
   stdId     : number;
   stdName   : string;
   stdFee    : number;
   stdCourse : string;
}
```

Step#3 Enable HTTP Client Module and register our service class

> open app.module.ts

> add one imports : HttpClientModule

```
  as:
   imports: [
    ..
    HttpClientModule
  ],
```

> From Module Import

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

> Register our service class

```
  providers : [StudentService]
```

Step#4 StudentService class

> add constructor HttpClient dependency from package '@angular/common/http'

```
  constructor(private http:HttpClient) { }
```

> define baseUrl="";

```
  private baseUrl = 'http://localhost:9898/springboot-crud-rest/rest/student';
```

> Define method that gets all students

```
  getAllStudents():Observable<Student[]>{
    return this.http.get<Student[]>(`${this.baseUrl}/all`);
  }

--code--
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';


@Injectable({
  providedIn: 'root'
})
export class StudentService {

  private baseUrl = 'http://localhost:9898/springboot-crud-
rest/rest/student';

  constructor(private http:HttpClient) { }

  getAllStudents():Observable<Student[]>{
    return this.http.get<Student[]>(`${this.baseUrl}/all`);
  }

}
---
```

Step#5 student-list component class coding

> Open StudentListComponent class
> define Observable variable for students array

```
 students : Student[];
```

> Add Constructor Dependencies for service and route

```
   constructor(private service:StudentService, private router:Router) { }
```

> Add method to read service getAllStudents and assign to students object

```
 getAllStudents(){
   this.service.getAllStudents().subscribe(data=>{this.students=data});
 }
```

> Also call in ngOnInit() method

```
--code--
import { Component, OnInit } from '@angular/core';
import { Student } from '../student';
import { StudentService } from '../student.service';
import { Router } from '@angular/router';



@Component({
 selector: 'app-student-list',
 templateUrl: './student-list.component.html',
 styleUrls: ['./student-list.component.css']
})
export class StudentListComponent implements OnInit {

 public students : Student[];



 constructor(private service:StudentService, private router:Router) { }

 ngOnInit(): void {
   this.getAllStudents();
 }

 getAllStudents(){
   this.service.getAllStudents().subscribe(data=>{this.students=data});
 }

}
```

Step#6 Student List component HTML code

```
<table>
    <tr>
        <th>ID</th>
        <th>NAME</th>
        <th>FEE</th>
        <th>COURSE</th>
    </tr>
    <tr *ngFor="let s of students">
```

```
        <td>{{s.stdId}}</td>
        <td>{{s.stdName}}</td>
        <td>{{s.stdFee}}</td>
        <td>{{s.stdCourse}}</td>
    </tr>
</table>
```

Step#7 add code in app routing modules (ts file)

```
  { path: '', redirectTo: 'students', pathMatch: 'full' },
  { path: 'all', component: StudentListComponent },
```

```
--code: app-routing.module.ts --(add below code)
const routes: Routes = [
  { path: '', redirectTo: 'all', pathMatch: 'full' },
  { path: 'all', component: StudentListComponent },

];
```

Step#8 app.component.html (menu bar and links)

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <!-- Brand/logo -->
  <a class="navbar-brand" href="#">NARESHIT(RAGHU SIR) STUDENTS
APP</a>

  <!-- Links -->
  <ul class="navbar-nav">
  <li class="nav-item">
    <a routerLink="add" class="nav-link text-
white"><b>Register</b></a>
  </li>
  <li class="nav-item">
    <a routerLink="all" class="nav-link text-white"><b>View</b></a>
  </li>

 </ul>
</nav>

<br/>

<router-outlet></router-outlet>
```

Spep#9 ng g class Message

```
export class Message {
  type:string;
  message:string;

}
```

Step#10 In Student Service class for save method

```
 createStudent(student : Student):Observable<Message>{
  return this.http.post<Message>(`${this.baseUrl}/save`,student);
 }
```

Step#11 Add code in CreateStudent component TS file

```
 student : Student = new Student();
 message : Message = new Message();
 error   : boolean ;

 constructor(private service:StudentService, private router:Router) { }

 ngOnInit(): void {
 }

 saveStudent(){

   console.log(this.student);
   this.service.createStudent(this.student)
   .subscribe(data =>{ this.message=data},
     error=>this.message=error);
   this.student=new Student();
   if(this.message.type=='FAIL') this.error=true;
   //this.gotoViewAll();
 }
/*
 gotoViewAll(){
   this.router.navigate(['/students']);
 }
 */
```

Step#12 CreateStudentComponent HTML

```
<h3>Create new Student</h3>
<form (ngSubmit)="saveStudent()">
   <div class="form-group">
      <label for="stdName">Student Name</label>
      <input type="text" class="form-control" id="stdName" required
[(ngModel)]="student.stdName" name="stdName"/>
   </div>
   <div>
      <label for="stdFee">Student Fee</label>
      <input type="text" class="form-control" id="stdFee" required
[(ngModel)]="student.stdFee" name="stdFee"/>
   </div>
   <div>
      <label for="stdCourse">Student Course</label>
      <input type="text" class="form-control" id="stdCourse" required
[(ngModel)]="student.stdCourse" name="stdCourse"/>
   </div>
   <div>
      <input type="submit" value="Register Student" class="btn btn-
success"/>
   </div>

</form>
<div *ngIf="error; else successBlock">
   <span class="text-danger">Unable to Save Student</span>
</div>
```

```
<ng-template #successBlock>
   <span class="text-success">{{message.message}}</span>
</ng-template>
```

Step#13 In StudentList HTML   For Delete Link

```
  <td>
     <button (click)="deleteStudent(s.stdId)" class='btn btn-
danger'>DELETE</button>
  </td>
```

Step#14 Student Service

```
 deleteStudent(stdId:number):Observable<Message>{
    console.log("Delete Service"+`${this.baseUrl}/remove/${stdId}`);
    return this.http.delete<Message>(`${this.baseUrl}/remove/${stdId}`);
  }
```

Step#15 StudentList Typescript file

```
deleteStudent(stdId:number){
    console.log("Delete data"+stdId);
    this.service.deleteStudent(stdId).subscribe(data=>{
      this.message=data,
      this.getAllStudents();
    });

}
```