

# Spring Boot 2 - Angular 9 CURD

## -by Mr. RAGHU

### SCREENS

**RAGHU SIR STUDENS APP** Register View All Students ▾

**Student Register**

Student Name :

Student Fee :

Student Course :

Register

**RAGHU SIR STUDENS APP** Register View All Students ▾

**Student Register**

Student Name :

Student Fee :

Student Course :

Register

**RAGHU SIR STUDENS APP** Register View All Students ▾

**Student Register**

Student Name :

Student Fee :

Student Course :

Register

1 saved

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

RAGHU SIR STUDENS APP Register View All Students ▾

ALL STUDENTS

ID	NAME	Fee	Course	Operations
1	PARAMESH	5200.5	SPRING PACKAGE	<button>DELETE</button> <button>EDIT</button>

RAGHU SIR STUDENS APP Register View All Students ▾

Welcome to Student Edit Page

Student Id :

Student Name :

Student Fee :

Student Course :

Update

RAGHU SIR STUDENS APP Register View All Students ▾

ALL STUDENTS

ID	NAME	Fee	Course	Operations
1	PARAMESH REBEL	5200.5	SPRING PACKAGE	<button>DELETE</button> <button>EDIT</button>

RAGHU SIR STUDENS APP Register View All Students ▾

ALL STUDENTS

ID	NAME	Fee	Course	Operations
1-removed				

## **RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]**

**Step#1 Create one Spring Starter Project in STS with Dependencies : Spring Web, Lombok, Data JPA, MySQL Driver, DevTools**

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>
```

**Step#2 Specify application.properties for DB Connection, JPA and server.**

```
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.MySQL55D
ialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

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

### Step#3 Create one Model class with JPA Mapping and Lombok Annotations

```
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;
}
```

and also for Message Return

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

import lombok.AllArgsConstructor;
import lombok.Data;
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
import lombok.NoArgsConstructor;
```

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Message {

    private String type;
    private String message;
}
```

### Step#4 Define 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>
{

}
```

### Step#5 Define Service Interface for all basic Operations

```
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);
}
```

### Step#6 Create ServiceImpl class for service interface

```
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);
    }
}
```

```
}  
  
}
```

### Step#7 Define RestController with Operations

```
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)  
     */  
}
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
@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;
}

/**
```



## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
* 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("SUCESSS",id+"-removed"),HttpStatus.OK);
        }else {
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
        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 ResponseEntity
 */
@PutMapping("/update")
public ResponseEntity<Message> updateStudent(
    @RequestBody Student student) {
    ResponseEntity<Message> resp=null;
    try {
        boolean exist=service.isExist(student.getStdId());
        if(exist) {
            service.saveStudent(student);
            resp=new ResponseEntity<Message>(new
Message("OK",student.getStdId()+"-Updated"),HttpStatus.OK);
        }else {
            resp=new ResponseEntity<Message>(new
Message("OK",student.getStdId()+"-Not Exist"),HttpStatus.BAD_REQUEST);
        }
    } catch (Exception e) {
        resp=new ResponseEntity<Message>(new
Message("OK","Unable to Update"),HttpStatus.INTERNAL_SERVER_ERROR);
        e.printStackTrace();
    }
    return resp;
}
}
```

# Angular Setup and code

---

## **## 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

## **## Check installtion of Node using cmd prompt**

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

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

## **## Install Angular (wait for : 10 mins to 1 hr after cmd)**

Open cmd prompt and type command like

- > npm install -g @angular/cli

## **## Check angular installtion using cmd**

- > ng version

## **## 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

## **## Open VS Code Editor**

- > File > open folder > create new folder (ex: myangapps) > Open
- > press ctrl+` (before to 1 Key)

## **RAGHU SIR [NARESH i TECHNOLOGIES, AMEERPET, HYDERABAD]**

**Step#8 Open Visual Studio Code Editor and Goto terminal 'ctrl+`'**

**Step#9 Create one new Angular Project**

cmd: ng new student-app-new  
with routing option : yes  
with CSS option : press Enter

**Step#10 Switch to student app folder**

cmd: cd student-app-new

**Step#11 Generate Models, Service and Components**

cmds:

ng g class student  
ng g class message  
ng g s student  
ng g c student-all  
ng g c student-create  
ng g c student-edit

**Step#12 Enable HTTP and Forms in Angular Project [app.module.ts]**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { StudentAllComponent } from './student-all/student-
all.component';
import { StudentCreateComponent } from './student-create/student-
create.component';
import { StudentEditComponent } from './student-edit/student-
edit.component';

@NgModule({
  declarations: [
    AppComponent,
    StudentAllComponent,
    StudentCreateComponent,
    StudentEditComponent
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
],  
imports: [  
  BrowserModule,  
  AppRoutingModule,  
  HttpClientModule,  
  FormsModule,  
],  
providers: [],  
bootstrap: [AppComponent]  
)  
export class AppModule { }
```

### Step#13 Add Properties in Student, Message Model classes

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

### Step#14 Define Service Layer Code with all HTTP Operations [student.service.ts]

```
import { Injectable } from '@angular/core';  
import { HttpClient } from '@angular/common/http';  
import { Student } from './student';  
import { Observable } from 'rxjs';  
import { Message } from './message';  
  
@Injectable({  
  providedIn: 'root'
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
  })
  export class StudentService {

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

    constructor(private http:HttpClient) { }

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

    deleteOneStudent(id:number):Observable<Message>{
      return this.http.delete<Message>(`${this.baseUrl}/remove/${id}`);
    }

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

    getOneStudent(id:number):Observable<Student>{
      return this.http.get<Student>(`${this.baseUrl}/one/${id}`);
    }

    updateStudent(student:Student):Observable<Message>{
      return this.http.put<Message>(`${this.baseUrl}/update`,student);
    }
  }
```

**Step#14 Provide Router link for each module**  
[app-routing.module.ts]

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { StudentAllComponent } from './student-all/student-all.component';
import { StudentCreateComponent } from './student-create/student-create.component';
import { StudentEditComponent } from './student-edit/student-edit.component';
```

```
const routes: Routes = [  
  {path: 'all', component: StudentAllComponent},  
  {path: 'add', component: StudentCreateComponent},  
  {path: 'edit/:id', component: StudentEditComponent},  
  {path: '', redirectTo: 'all', pathMatch: 'full'},  
];  
  
@NgModule({  
  imports: [RouterModule.forRoot(routes)],  
  exports: [RouterModule]  
})  
export class AppRoutingModule { }
```

**Step#15 Define Menu bar and routerLink [app.component.html]**

```
<nav class="navbar navbar-expand-lg navbar-light bg-success">  
  <a class="navbar-brand text-white" href="#">RAGHU SIR STUDENS APP</a>  
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">  
    <span class="navbar-toggler-icon"></span>  
  </button>  
  <div class="collapse navbar-collapse" id="navbarNavDropdown">  
    <ul class="navbar-nav">  
      <li class="nav-item active">  
        <a class="nav-link text-white" href="#" routerLink="add">Register </a>  
      </li>  
      <li class="nav-item">  
        <a class="nav-link text-white" href="#" routerLink="all">View All</a>  
      </li>  
  
      <li class="nav-item dropdown">  
        <a class="nav-link dropdown-toggle text-white" href="#" id="navbarDropdownMenuLink" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
        Students
      </a>
      <div class="dropdown-menu" aria-
labelledby="navbarDropdownMenuLink">
        <a class="dropdown-
item" href="#" routerLink="add">Register</a>
        <a class="dropdown-
item" href="#" routerLink="all">View All</a>

      </div>
    </li>
  </ul>
</div>
</nav>

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

### Step#17 Edit index.html page

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>StudentAppNew</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <!-- CSS only -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootst
rap/4.5.0/css/bootstrap.min.css" integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYyxFfc+NcPb1dKGj7Sk" cros
sorigin="anonymous">

  <!-- JS, Popper.js, and jQuery -->
  <script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js" integrity="sha384-
DfxDz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" cros
sorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/po
pper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" cros
sorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/boo
tstrap.min.js" integrity="sha384-
```



## RAGHU SIR [NARESH i TECHNOLOGIES, AMEERPET, HYDERABAD]

```
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" cros
sorigin="anonymous"></script>
</head>
<body>
<div class="container">
  <app-root></app-root>
</div>
</body>
</html>
```

### Step#18 Display all Students [students-all.component.ts]

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

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

  students : Student[];
  message : Message = new Message();

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

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

  getAllStudents(){
    this.service.getAllStudents().subscribe(data=>{this.students=data}
    ,
    error=>{
      this.students=[]
    });
  }

  deleteStudent(id:number){
    this.service.deleteOneStudent(id).subscribe(
```

**RAGHU SIR [NARESH i TECHNOLOGIES, AMEERPET, HYDERABAD]**

```
data=>{
  this.message=data,
  this.getAllStudents();
},
error=>{console.log(error)}
);
}
editStudent(id:number){
  this.router.navigate(['edit',id]);
}
}
```

[student-all.component.html]

[illegible]

### Step#19 Create Student Register [student-create.component.ts]

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

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

  student : Student = new Student();
  message : Message = new Message();

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

  ngOnInit(): void {
  }

  createStudent(){
    this.service.createStudent(this.student).subscribe(data=>{
      this.message=data;
    });
    this.student=new Student();
  }
}
```

### [student-create.component.html]

```
<h3>Student Register</h3>
<form (ngSubmit)="createStudent()">
  Student Name : <input type="text" class="form-
control" [(ngModel)]="student.stdName" name="stdName"/>
  Student Fee : <input type="text" class="form-
control" [(ngModel)]="student.stdFee" name="stdFee"/>
  Student Course : <input type="text" class="form-
control" [(ngModel)]="student.stdCourse" name="stdCourse"/>
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

```
<input type="submit" value="Register" class="btn btn-success"/>
</form>
{{message.message}}
```

Step#20 Create Student Edit Page and update code  
[student-edit.component.ts]

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

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

  student : Student ;
  id : number;

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

  ngOnInit(): void {
    this.student =new Student();
    this.id=this.activeRouter.snapshot.params['id'];
    this.service.getOneStudent(this.id).subscribe(
      data=>{
        this.student=data;
      }
    );
  }
  updateStudent(){
    this.service.updateStudent(this.student).subscribe(data=>{
      console.log(data),
      this.router.navigate(['/all']);
    });
  }
}
```

## RAGHU SIR [NARESH I TECHNOLOGIES, AMEERPET, HYDERABAD]

[student-edit.component.html]

```
<h3>Welcome to Student Edit Page</h3>
<form (ngSubmit)="updateStudent()">
  Student Id : <input type="text" class="form-
control" [(ngModel)]="student.stdId" name="stdId" readonly/>

  Student Name : <input type="text" class="form-
control" [(ngModel)]="student.stdName" name="stdName"/>

  Student Fee : <input type="text" class="form-
control" [(ngModel)]="student.stdFee" name="stdFee"/>

  Student Course : <input type="text" class="form-
control" [(ngModel)]="student.stdCourse" name="stdCourse"/>

  <input type="submit" value="Update" class="btn btn-success"/>
</form>
```

**Step#21 Start Angular Application**

> ng serve --open

**Step#22 Go to browser and Enter URL : <http://localhost:4200>**

**Email: [javabyraghu@gmail.com](mailto:javabyraghu@gmail.com)**

**FB: <https://www.facebook.com/groups/thejavatemple/>**