Spring Boot CRUD REST full API Using Project Structure

Spring Boot REST Full API layers Controller, Service and DAO

-:Controller Layer:-

:Example Controller Layer:

```
package com.Gopal.Controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
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.PatchMapping;
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.RestController;
import com.Gopal.Pojo.Student;
import com.Gopal.Service.MyService;
```

```
@RestController
public class MyController {

    @Autowired
    private MyService myService;

    @GetMapping("showSingleStudentRecord/{id}")
    public String showSingleStudentRecord(@PathVariable int id) {
        return myService.showSingleStudentRecord(id);
    }

    @GetMapping("showMultipleStudentRecord")
    public String showMultipleStudentRecord() {
        return myService.showMultipleStudentRecord();
    }

    @PostMapping("insertSingleStudentRecord")
    public String insertSingleStudentRecord(@RequestBody Student
student) {
        return myService.insertSingleStudentRecord(student);
    }

    @PostMapping("insertSingleRecordPath/{id}/{name}")
    public String insertAPI(@PathVariable int id, String name) {
        return myService.insertAPI(id, name);
    }
```

```
    @PostMapping("insertMultipleStudentRecord")
    public String insertMultipleStudentRecord(@RequestBody
List<Student> list) {
        return myService.insertMultipleStudentRecord(list);
    }

    @PutMapping("updateSingleStudentRecord")
    public String updateSingleStudentRecord(@RequestBody Student
student) {
        return myService.updateSingleStudentRecord(student);
    }

    @PutMapping("updateMultipleRecord")
    public String multipleupdate(@RequestBody List<Student> list) {
        return myService.multipleupdate(list);
    }

    @PatchMapping("partialSingleFieldUpdate/{id}/{name}")
    public String partialSingleFieldUpdate(@PathVariable int id, String
name) {
        return myService.partialSingleFieldUpdate(id, name);
    }

    @PatchMapping("partialMultipleFieldUpdate/{id}/{name}")
    public String partialMultipleFieldUpdate(@PathVariable int id, String
name, String city) {
        return myService.partialMultipleFieldUpdate(id, name, city);
    }
```

```
    @DeleteMapping("deleteSingleStudentRecord/{id}")
    public String deleteSingleStudentRecord(@PathVariable int id) {
        return myService.deleteSingleStudentRecord(id);
    }

    @DeleteMapping("deleteMultipleRecord")
    public String deleteAllStudentRecord() {
        return myService.deleteAllStudentRecord();
    }

}
```

-:Service Layer:-

:Example Service Layer:

```
package com.Gopal.Service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.Gopal.DAO.MyDAO;
import com.Gopal.Pojo.Student;

@Service
public class MyService {
```

```
@Autowired
private MyDAO myDAO;

public String showSingleStudentRecord(int id) {
    return myDAO.showSingleStudentRecord(id);
}

public String showMultipleStudentRecord() {
    return myDAO.showMultipleStudentRecord();
}

public String insertSingleStudentRecord(Student student) {
    return myDAO.insertSingleStudentRecord(student);
}

public String insertAPI(int id, String name) {
    return myDAO.insertAPI(id, name);
}

public String insertMultipleStudentRecord(List<Student> list) {
    return myDAO.insertMultipleStudentRecord(list);
}

public String updateSingleStudentRecord(Student student) {
    return myDAO.updateSingleStudentRecord(student);
}
```

```
    public String multipleupdate(List<Student> list) {
        return myDAO.multipleupdate(list);
    }

    public String partialSingleFieldUpdate(int id, String name) {
        return myDAO.partialSingleFieldUpdate(id, name);
    }

    public String partialMultipleFieldUpdate(int id, String name, String
city) {
        return myDAO.partialMultipleFieldUpdate(id, name, city);
    }

    public String deleteSingleStudentRecord(int id) {
        return myDAO.deleteSingleStudentRecord(id);
    }

    public String deleteAllStudentRecord() {
        return myDAO.deleteAllStudentRecord();
    }

}
```

-:DAO Layer (Data Access Object):-

:Example DAO Layer:

```
package com.Gopal.DAO;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import com.Gopal.Pojo.Student;

@Repository
public class MyDAO {

    @Autowired
    SessionFactory sf;

    // Show a Single Student Record
    public String showSingleStudentRecord(int id) {
        Session s = sf.openSession();
        Student ss = s.get(Student.class, id);
        return "Show Single Student Record \n" + ss;
    }
```

```
// Get a Multiple Student Record
public String showMultipleStudentRecord() {
	Session s = sf.openSession();
	Query query = s.createQuery("from Student");
	List<Student> list = query.list();
	return "Show Multiple Student Record \n" + list;
}

// Insert a Single Student Record
public String insertSingleStudentRecord(Student student) {
	Session s = sf.openSession();
	Transaction t = s.beginTransaction();
	s.save(student);
	t.commit();
	return "Insert Single Student Record \n" + student;
}

// Insert a Single Student Record Using Request API
public String insertAPI(int id, String name) {
	Session s = sf.openSession();
	Transaction t = s.beginTransaction();
	Student ss = new Student(id, name);
	s.save(ss);
	t.commit();
return "Insert a Single Student Record Using Request API \n" + ss;
}
```

```
// Insert a Single Student Record
public String insertMultipleStudentRecord(List<Student> list) {
    for (Student student : list) {
        Session ss = sf.openSession();
        Transaction tt = ss.beginTransaction();
        ss.save(student);
        tt.commit();
    }
    return "Insert Multiple Student Record \n" + list;
}

// Update a Single Student Record
public String updateSingleStudentRecord(Student student) {
    Session s = sf.openSession();
    Transaction t = s.beginTransaction();
    s.save(student);
    t.commit();
    return "Update Single Student Record \n" + student;
}

    // Update a Multiple Student Record
    public String multipleupdate(List<Student> list) {
        for (Student student : list) {
            Session ss = sf.openSession();
            Transaction tt = ss.beginTransaction();
            ss.update(student);
            tt.commit();
        }
            return "Update Multiple Student Record \n" + list;
        }
```

```
    // Partial Update a Single Student field Record
    public String partialSingleFieldUpdate(int id, String name) {
        Session s = sf.openSession();
        Student student = s.get(Student.class, id);
        student.setName(name);
        Transaction tx = s.beginTransaction();
        s.update(student);
        tx.commit();
        return "Partial Update a Single Student Record \n"+student;
    }

    // Partial Update a Multiple Student field Record
    public String partialMultipleFieldUpdate(int id, String name, String
city) {
        Session s = sf.openSession();
        Student student = s.get(Student.class, id);
        student.setName(name);
        // student.setCity(city);
        Transaction tx = s.beginTransaction();
        s.update(student);
        tx.commit();
        return "Partial Update a Multiple Student field Record
\n"+student;
    }
```

```
    // Delete a Single Student Record
    public String deleteSingleStudentRecord(int id) {
        // Database interaction code
        Session s = sf.openSession();
        Transaction t = s.beginTransaction();
        Student ss = s.get(Student.class, id);
        s.delete(s);
        t.commit();
        return "Delete Single Student Record \n" + ss;
    }

    // Delete a Multiple Student Record's
    public String deleteAllStudentRecord() {
        Session s = sf.openSession();
        Transaction t = s.beginTransaction();
        Query query = s.createQuery("DELETE FROM Student");
        query.executeUpdate();
        t.commit();
        return "Deletes All Table Record's";
    }
}
```

-:POJO Class:-

```
package com.Gopal.Pojo;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Student {

	@Id
	private int id;
	private String name;

	public Student() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Student(int id, String name) {
		super();
		this.id = id;
		this.name = name;
	}

	public int getId() {
		return id;
	}
```

```
    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }

}
```