

## Why do we need controller + Service + Repo?

Spring Boot follows very clean architecture called 3 tier architecture.

- ① Controller layer → Handles requests from user  
(Postman, browser, mobile app)
- ② Service layer → Business logic, calculation, validation?
- ③ Repo layer → Talks to database (MySQL, H<sub>2</sub>, PostgreSQL)

Client → Controller → Service → Repository → Database

- ① Controller -
- Accepts API requests
  - Converts JSON → Java objects.
  - Sends it to service

### ② Service - The brain!

- Thinks
- validates data
- checks emails / apply calculat? / Throw errors.
- calls repository methods.

Service is the logic hub.

### ③ Repository - Database layer

- Talks directly to database.
- It saves
  - updated
  - Deletes
  - fetches

#### In Controller (API Endpoints)

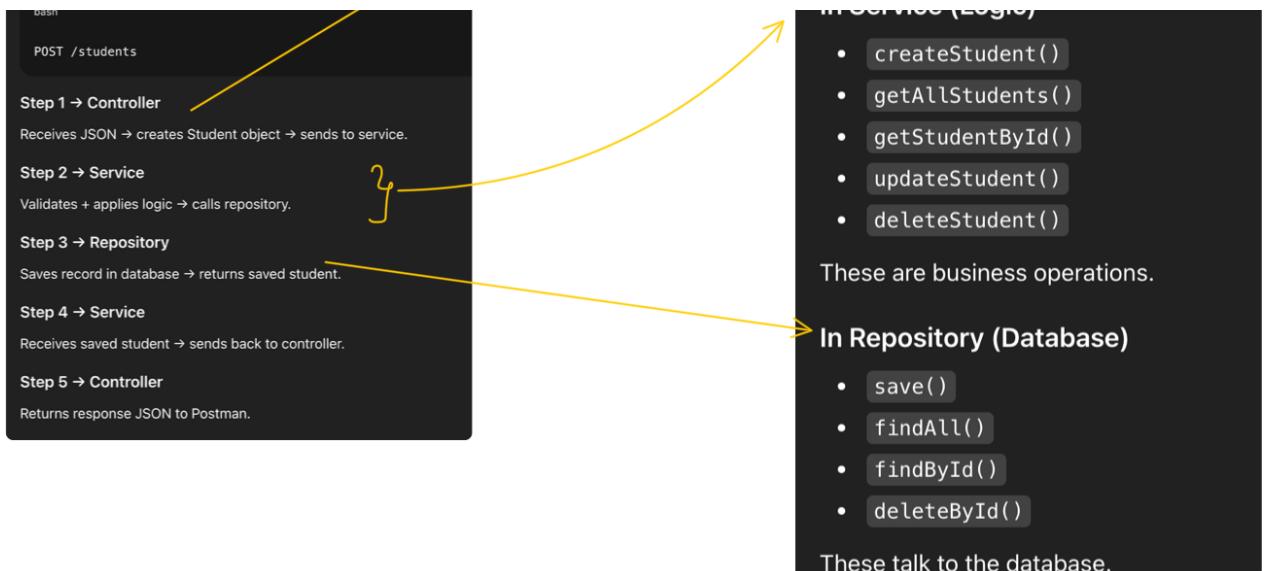
- @PostMapping → Create
- @GetMapping → Read
- @PutMapping → Update
- @DeleteMapping → Delete

These are the API URLs users call.

#### PUTTING IT ALL TOGETHER (FULL FLOW)

If you call:

#### In Service (Logic)



## Code Implementation?

①

```
① StudentController.java ×
6
7 import java.util.List;
8
9 @RestController no usages
10 @RequestMapping("students")
11 public class StudentController {
12     private final StudentService service; 3 usages
13
14     public StudentController(StudentService service) { no usages
15         this.service = service;
16     }
17
18     @PostMapping no usages
19     public Student createStudent(@RequestBody Student student){
20         return service.addStudent(student);
21     }
22
23     @GetMapping no usages
24     public List<Student> getAllStudents(){
25         return service.getAllStudents();
26     }

```

②

```
② StudentService.java ×
6
7     @Service 3 usages
8     public class StudentService {
9
10         private final StudentRepository repo; 3 usages
11
12         public StudentService(StudentRepository repo) { no usages
13             this.repo = repo;
14         }
15
16         public Student addStudent(Student student){ 1 usage
17             return repo.save(student);
18         }
19
20         public List<Student> getAllStudents(){ 1 usage
21             return repo.findAll();
22         }
23     }
24
25 }
26
```

③

```
③ StudentRepository.java ×
1 package com.example.Day5.repository;
2
3 import com.example.Day5.entity.Student;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface StudentRepository extends JpaRepository<Student, Long> {
7
8     /*
9      This gives you READY-MADE CRUD methods, like:
10     save()
11     findAll()
12     findById()
13     deleteById()
14 */

```