

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, H2, 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 calculation? / Throw errors.
- Calls repository methods.

Service is the logic hub.

③ Repository — Database layer

→ Talks directly to database.

- It
- saves
 - updates
 - Deletes
 - fetches.

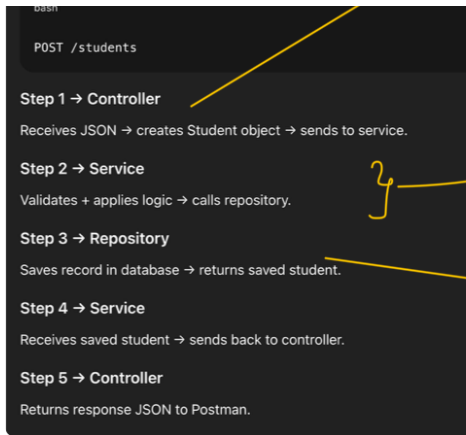
📌 PUTTING IT ALL TOGETHER (FULL FLOW)
If you call:

In Controller (API Endpoints)

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

These are the API URLs users call.

In Service (Logic)



Code Implementatⁿ?

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> getStudents(){
25         return service.getAllStudents();
26     }
27 }
```

②

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

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     This gives you READY-MADE CRUD methods, like:
9     save()
10    findAll()
11    findById()
12    deleteById()
13    */
14 }
```