Why do we need controller + Servia + 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 → Buisness logic, calculation, validat^n

③ Repo layer → Talks to database (Mysql, H_2, 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^n / Throw errors.
- Calls repository methods.

Service is the logic hub.

③ Repository — Database layer

→ Talks directly to database.

→ It
- saves
- updates
- Deletes
- fetches.

In Controller (API Endpoints)

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

These are the API URLs users call.

In Service (Logic)

📌 PUTTING IT ALL TOGETHER (FULL FLOW)

If you call:

```
POST /students
```

Step 1 → Controller

Receives JSON → creates Student object → sends to service.

Step 2 → Service

Validates + applies logic → calls repository.

Step 3 → Repository

Saves record in database → returns saved student.

Step 4 → Service

Receives saved student → sends back to controller.

Step 5 → Controller

Returns response JSON to Postman.

- `createStudent()`
- `getAllStudents()`
- `getStudentById()`
- `updateStudent()`
- `deleteStudent()`

These are business operations.

In Repository (Database)

- `save()`
- `findAll()`
- `findById()`
- `deleteById()`

These talk to the database.

Code Implementation

1

StudentController.java

```java
import java.util.List;

@RestController  no usages
@RequestMapping("/students")
public class StudentController {
    private final StudentService service;  3 usages

    public StudentController(StudentService service) {  no usages
        this.service = service;
    }

    @PostMapping  no usages
    public Student createStudent(@RequestBody Student student){
        return service.addStudent(student);
    }

    @GetMapping  no usages
    public List<Student> getStudents(){
        return service.getAllStudents();
    }
```

2

StudentService.java

```java
@Service  3 usages
public class StudentService {

    private final StudentRepository repo;  3 usages

    public StudentService(StudentRepository repo) {  no usages
        this.repo = repo;
    }

    public Student addStudent(Student student){  1 usage
        return repo.save(student);
    }

    public List<Student> getAllStudents(){  1 usage
        return repo.findAll();
    }
}
```

3

StudentRepository.java

```java
package com.example.Day5.repository;

import com.example.Day5.entity.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {
    /*
    This gives you READY-MADE CRUD methods, like:
    save()
    findAll()
    findById()
    deleteById()
     */
}
```