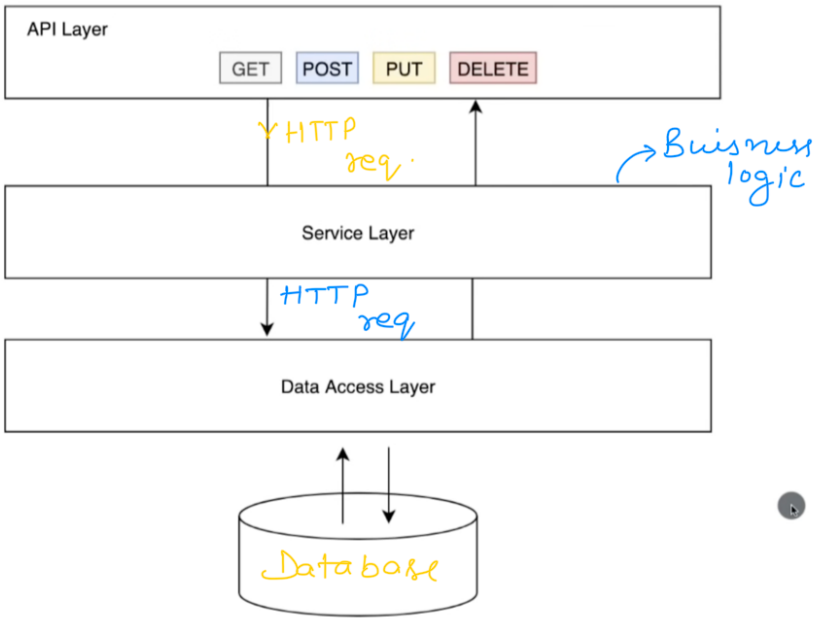
Spring Boot (framework)

→ It provides –

- Microservices
- Security
- dependency injection
- logging.
- easy to learn.



→ We use spring initializer to Bootstrap any spring boot Application.

Location : folderName → src → main → java → com.example.folderName → folderNameApplication

@RESTController → used to create Restful web servic

@Controller

- serializes return objects

@ResponseBody

- Directly returns the output from return of a method.

Automatically

serializes return objects into JSON / XML response

@__Mapping:

This is used to map HTTP requests to specific handler methods in spring Controllers

Example of a first web App -

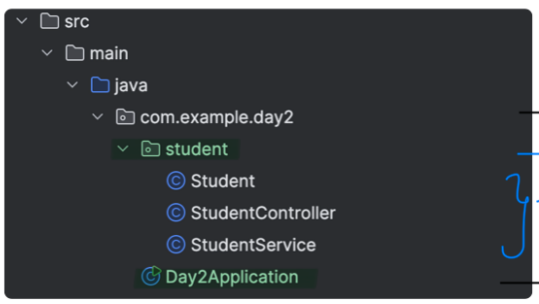
defines and runs as a Springboot App.

```java
package com.example.firstWebApp;

import ...

@SpringBootApplication
@RestController
public class FirstWebAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(FirstWebAppApplication.class, args);
    }

    @GetMapping("/hello")
    public String hello() {
        return "Hello Pankaj";
    }

    @GetMapping("/time")
    public String time() {
        return LocalTime.now().toString();
    }
}
```

- Gets mapping at localhost:8080/hello
- localhost:8080/time

Example of a Slightly advanced webApp

→ Structure:



→ head file
→ Sub file
→ classes inside sub file.
→ main Applicat^n

① Main Applicat^n

→ Simply runs the

```java
package com.example.day2;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

Spring Boot App.

```java
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication  Pankaj-Singh-Rawat
@RestController
public class Day2Application {

    public static void main(String[] args) {  Pankaj-Singh-Rawat
        SpringApplication.run(Day2Application.class, args);
    }

}
```

② Student class

Contains –

- private variables
- constructors
- getter & setter methods.

Student.java

```java
package com.example.day2.student;

import java.time.LocalDate;

public class Student {  3 usages  Pankaj-Singh-Rawat
    private Long id;  4 usages
    private String name;  5 usages
    private String email;  5 usages
    private int age;  5 usages
    private LocalDate dob;  5 usages

    public Student() {  no usages  Pankaj-Singh-Rawat
    }

    public Student(String name, String email, int age, LocalDate dob) {  no usages
        this.name = name;
        this.email = email;
        this.age = age;
        this.dob = dob;
    }

    public Student(Long id, String name, String email, int age, LocalDate dob) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.age = age;
        this.dob = dob;
    }

    public Long getId() {  no usages  Pankaj-Singh-Rawat
        return id;
    }

    public void setId(Long id) {  no usages  Pankaj-Singh-Rawat
```

③ StudentController

(API Level)

Creates RESTful API

Mapping at HTTP

Immutable variable for calling Student Service

dependency Injection

Constructor

StudentController.java

```java
package com.example.day2.student;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.time.LocalDate;
import java.time.Month;
import java.util.List;

@RestController  no usages  Pankaj-Singh-Rawat
@RequestMapping(path = "api/v1/student")
public class StudentController {

    private final StudentService studentService;  2 usages

    @Autowired  no usages  Pankaj-Singh-Rawat
    public StudentController(StudentService studentService) {
        this.studentService = studentService;
    }

    @GetMapping  no usages  Pankaj-Singh-Rawat
    public List<Student> getStudents() {
        return studentService.getStudents();
    }
}
```

getStudents() method to call data from service class.

④ StudentService

defined as @Service, @Controller can also be used.

get student method which returns list of students.

object.

StudentService.java

```java
package com.example.day2.student;

import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.GetMapping;

import java.time.LocalDate;
import java.time.Month;
import java.util.List;

@Service  2 usages  Pankaj-Singh-Rawat
public class StudentService {
    public List<Student> getStudents(){  1 usage  Pankaj-Singh-Rawat
        return List.of(
                new Student(
                        id: 1L,
                        name: "Pankaj",
                        email: "qoolphoenix@gmail.com",
                        age: 21,
                        LocalDate.of( year: 2003, Month.NOVEMBER,  dayOfMonth: 29)
                )
        );
    }
}
```

Dependency Injection (DI)

→ It is a core principle of Inversion of control (IOC) container. It recieves it's dependencies from Spring IOC container rather than managing them themselves.

Maintains Beans

Helps in → a) loose coupling
b) making code modular, testable and maintainable.

Beans are objects managed by spring

Eg -> @component
@service
@controller
etc

a) Autowire → On creating a bean, spring container automatically identifies its

dependencies and injects the appropriat
instances into it.