* Spring framework is a light weight application
* Spring is loosely coupled application
* Spring mostly works on HAS-A Relationship.
* IOC container (inversion of control)

1. Spring core item
2. It will creates the bean and it will configure the bean and manages the bean till its usage.
3. It helps read the data directly from Property files/ configuration file.

* Dependency injection:

This makes the application loosely coupled with the help of IOC container.

Types:

1. Setter injection: Data will be injected in setter method
2. Constructor Injection: data will be injected in constructor

* Bean: bean is nothing but an object in spring, it is created by IOC

By default it is singleton object.

It is used on Method level i.e., @Bean

* Component: it is also a bean but which makes entire class as a single component. 🡺 @Component
* Spring boot has embedded tomcat server.
* Spring boot is auto-configurer
* Spring boot has actuators 🡺 monitoring

/actuator/health

Actuator/info 🡺

* Spring boot handles Dependency management and version control issues. 🡺 starter POM

Controller layer 🡺 http://localhost:8080/student/1

Service layer 🡺 Busineess logic

Repository Layer/ DAO layer (data access object) 🡺 data base related.

Hello h = new Hello();

Hello h;

A🡺 B 🡺 C

@EnableAutoConfiguration

@SpringBootConfiguration

@ComponentScan

@Springbootapplication

@Controller 🡺

@ResponseBody 🡺 java object response converts to json object 🡺 httpresponse

@RequestBody 🡺 HttPRequest Data json object to java object converting

@RestController == @Controller + @ResponseBody

@RequestMapping (“/student”)🡺 urls

@Service

@Repository

@Transactional

@Bean

@Component

@Configuration

@Autowired

@Qualifier

@PostMapping🡺 save

@GetMapping 🡺 get

@PutMapping 🡺 update

@DeleteMapping 🡺 delete

@PathVariable 🡺 reading data from url

@RequestParam 🡺 reading data from url with key value pair

Employee 🡺 data add, fetching, update, delete

Address

Company

Projects

Student

Address

College

Course

Diagram

Description automatically generated