********** Interview Questions **********

**1) What is Spring?**

Ans -> Spring is a framework and it makes programming quicker, easier, and safer for everybody. Spring's focus on speed, simplicity, and productivity

**2) What is the framework?**

Ans -> Frameworks are software that are developed and used by developers to build applications.

Since they are often built, tested, and optimized by several experienced software engineers and programmers, software frameworks are versatile, robust, and efficient.

Using a software framework to develop applications lets you focus on the high-level functionality of the application. This is because any low-level functionality is taken care of by the framework itself

**3) What is Spring boot?**

Ans -> Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

**4) What are the spring boot features ?**
Ans ->

1.Create stand-alone Spring applications

2.Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)

3.Provide opinionated 'starter' dependencies to simplify
your build configuration

4.Automatically configure Spring and 3rd party libraries whenever possible

5.Provide production-ready features such as metrics,
health checks, and externalized configuration

6.Absolutely no code generation and no
requirement for XML configuration

**5) What is inversion of control(ioc)?**

Inversion of Control is a principle in software engineering which transfers the control of objects or portions of a program to a container

In the Spring framework, the interface ApplicationContext represents the IoC container. The Spring container is responsible for instantiating, configuring and assembling objects known as beans, as well as managing their life cycles.

**6) What is dependency injection?**

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container "injects" objects into other objects or "dependencies".

Types of DI -

1. Constructor Injection
2. Setter Injection
3. Field Injection

**7) What is Gradle?**

Gradle is a build automation tool for multi-language software development. It controls the development process in the tasks of compilation and packaging to testing, deployment, and publishing.

## 10) Annotations -

**@Configuration** - annotation is used to mark a class as a source of the bean definitions.

**@Autowired** - To wire the application parts together, use the @Autowired on the fields, constructors, or methods in a component. Spring's dependency injection mechanism wires appropriate beans into the class members marked with @Autowired.

**@Component** - Another way to declare a bean is to **mark a class with a @Component** annotation

**@Bean**(means Java Object)-@Bean annotation tells that a **method produces a bean** to be managed by the Spring container.It is a method-level annotation.

**@Entity** - @Entity annotation specifies that the class is an entity and is mapped to a database table.

**@Id -** It specifies the **primary key**

**@GeneratedValue** -Provides for the **specification of generation strategies** for the values of primary keys.

**@Scope -** Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be prototype. Similarly, if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be singleton.

**@RequestMapping** - It is used to **map web** requests. It has many optional elements like consumes, header, method, name, params, path, produces, and value. We use it with the class as well as the method.

**@GetMapping:** It maps the **HTTP GET** requests on the specific handler method. It is used to create a web service endpoint that **fetches** It is used instead of using:
**@RequestMapping(method = RequestMethod.GET)**

**@PostMapping:** It maps the **HTTP POST** requests on the specific handler method. It is used to create a web service endpoint that **creates** It is used instead of using:
**@RequestMapping(method = RequestMethod.POST)**

**@PutMapping:** It maps the **HTTP PUT** requests on the specific handler method. It is used to create a web service endpoint that **creates** or **updates** It is used instead of using:
**@RequestMapping(method = RequestMethod.PUT)**

**@DeleteMapping:** It maps the **HTTP DELETE** requests on the specific handler method. It is used to create a web service endpoint that **deletes** a resource. It is used instead of using:
**@RequestMapping(method = RequestMethod.DELETE)**

**@RequestBody:** It is used to **bind** HTTP requests with an object in a method parameter. Internally it uses **HTTP MessageConverters** to convert the body of the request. When we annotate a method parameter with **@RequestBody,** the Spring framework binds the incoming HTTP request body to that parameter.

**@ResponseBody:** It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return object into JSON and XML format.

**@PathVariable:** It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple @PathVariable in a method.

**@RequestParam:** It is used to extract the query parameters from the URL. It is also known as a **query parameter**. It is most suitable for web applications. It can specify default values if the query parameter is not present in the URL.

**@RestController:** It can be considered as a combination of **@Controller** and **@ResponseBody** annotations**.** The @RestController annotation is itself annotated with the @ResponseBody annotation. It eliminates the need for annotating each method with @ResponseBody.