**Annotation**

**Annotations** are a form of metadata that provide data about a program but are not part of the program itself. Annotations have no direct effect on the operation of the code they annotate. They are used to provide information to the compiler, compile-time and deployment-time processing tools, and runtime libraries.

In Java, annotations can be used for various purposes:

1. **Compiler Instructions**: Annotations can give the compiler instructions, e.g., @Override tells the compiler that a method is intended to override a method in a superclass.
2. **Runtime Processing**: Annotations can be retained at runtime and can be accessed via reflection to perform various operations. For example, JUnit uses annotations like @Test to identify test methods.
3. **Deployment Descriptors**: Annotations can replace or supplement XML configuration files. For example, in Spring, @Component or @Service annotations are used to declare a class as a Spring component.

Common annotations in Java include:

@Override

@Deprecated

@SuppressWarnings

In the context of frameworks like Spring, there are annotations such as:

@Autowired

@Component

@Service

@Repository

@Configuration

@Bean

**Spring Boot**

**Spring Boot** is a framework built on top of the Spring Framework. It simplifies the development of stand-alone, production-grade Spring applications by providing a range of non-functional features commonly used in application development, such as embedded servers, security, metrics, health checks, and externalized configuration.

Spring Boot is designed to get you up and running as quickly as possible with minimal configuration. Key features include:

1. **Auto-Configuration**: Automatically configures your Spring application based on the dependencies you have added. For example, if you include a database dependency, Spring Boot will automatically set up a connection to the database.
2. **Starter POMs**: Provide a convenient way to include common dependencies in your project. For instance, spring-boot-starter-web brings in all dependencies needed to build a web application.
3. **Embedded Servers**: You can run a Spring Boot application directly using an embedded server like Tomcat, Jetty, or Undertow, which means you don’t need to deploy WAR files to an external server.
4. **Production-Ready Features**: Includes features like health checks, metrics, and externalized configuration out-of-the-box.
5. **Spring Boot CLI**: Allows you to quickly prototype with Groovy.