Annotations

1.@Bean

The ****@Bean**** annotations are used at the method level and indicate that a method produces a bean that is to be managed by the Spring container. It is an alternative to the XML<bean> tag.

1. @Service

It is used at the class level. It shows that the annotated class is a service class, such as business basic logic, and call external APIs.

1. @Repository

It is a Data Access Object (DAO) that accesses the database directly. It indicates that the annotated class is a repository. The repository annotation indicates the class has the capability of storage, retrieval, updating, deletion, search and data manipulation, querying, and providing a structured approach to database operations.

1. @Configuration

It is used as a source of bean definitions. It is a class-level annotation.The configuration annotation represents the class having one or more @Bean methods.

1. @Controller

Among all Spring Boot annotations in Java, @Controller is regarded as one of the most important annotations in Spring Boot. The annotation is used to indicate that the class is a web request handler. It is often used to present web pages. It is most commonly used with @RequestMapping annotation. By marking a class with @Controller, developers explicitly state that this class is responsible for processing HTTP requests and often involves the presentation of web pages. This annotation is frequently paired with the @RequestMapping annotation to define the URL patterns that map to specific methods within the controller.

1. @RequestMapping

RequestMapping is used to map the HTTP request. It is used with the class as well as the method. It has many other optional elements like consumes, name, method, request, path, etc.It is one of the most used annotations in Spring Boot and can be applied at both the class and method levels, allowing developers to define URL patterns for handling incoming requests.

1. @Autowired

This annotation is used to auto-wire spring bean on setter methods, constructor and instance variable. It injects object dependency implicitly. When we use this annotation, the spring container auto-wires the bean by its matching data type.

1. @Component

@Component is also regarded as one of the most used annotations in Spring Boot.It is a class-level annotation that turns the class into Spring bean at the auto-scan time.The component annotation can automatically detect custom beans. It represents that the framework could autodetect these classes for dependency injection.

1. @SpringBootApplication

* @Configuration
* @ComponentScan
* @EnableAutoConfiguration

The class annotated with @SpringBootApplication is kept in the base package. This annotation does the component scan. However, only the sub-packages are scanned.

* Auto-configuration –The @EnableAutoConfiguration annotation is a core component of @SpringBootApplication. It triggers Spring Boot’s auto-configuration mechanism, which automatically configures the application based on the dependencies present in the classpath. Auto-configuration aims to reduce the need for explicit configuration by providing sensible defaults, making it easier for developers to get started with Spring Boot projects.
* Spring Boot Configuration –The @SpringBootConfiguration annotation is synonymous with @Configuration and indicates that the annotated class contains configuration methods. Configuration methods, marked with @Bean, define and configure beans in the Spring application context. This annotation is essential for organizing and centralizing the configuration of the application.
* Component Scan –The @ComponentScan annotation automatically discovers and registers Spring components within specified packages, such as controllers, services, and repositories. By default, it scans the class package annotated with @SpringBootApplication, making all components in that package and its sub-packages available for dependency injection. Component scanning enhances modularity and allows for the seamless integration of Spring components without explicit configuration.

1. @EnableAutoConfiguration

It’s one of the most important annotations in Spring Boot that simplifies application context initialization and setup by automating configuration based on classpath settings and added dependencies.It is placed on the main application class. Based on classpath settings, other beans, and various property settings, this annotation instructs SpringBoot to start adding beans.

1. @ComponentScan

It is used to scan a package of beans. It is used with the annotation @Configuration to allow Spring to know the packages to be scanned for annotated components. This annotation is also used to specify base packages.

1. @Required

This annotation is applied to bean setter methods. It indicates that the required property must be filled at the configuration time in the affected bean, or else it throws an exception: BeanInitializationException.

1. @Qualifier

It is used along with @Autowired annotation. It is used when more control is required over the dependency injection process. Individual constructor arguments or method parameters can be specified by using this annotation. Confusion arises when more than one bean of the same type is created, and only one of them is to be wired with a property, @Qualifier is used to get rid of the confusion.

1. @CookieValue

It is used at the method parameter level as an argument of the request mapping method. For a given cookie name, the HTTP cookie is bound to a @CookieValue parameter.The cookie value annotation is used to get the value of any HTTP cookie. Cookies in spring boot are used to show the personalised content to the users and they can be retrieved using the cookie value annotation.

1. @Lazy

It is used in the component class. At startup, all auto-wired dependencies are created and configured. But a @Lazy annotation can be created if a bean is to be initialized lazily. This means that only if it is requested for a bean will be created. It can also be used on @Configuartion classes. It’s an indication that all @Bean methods within that @Configuration should be lazily initialized.