# Annotations

**1. What is @SpringBootApplication?**
It is a combination of @Configuration, @EnableAutoConfiguration, and @ComponentScan. It boots the Spring application with auto configuration and component scanning. It reduces boilerplate because you don’t need to add all three annotations separately.

**2. What is @ComponentScan?**
It tells Spring where to search for beans. It scans packages for @Component, @Service, @Controller, and @Repository. I use it when my packages are outside the default base package.

**3. What is @EnableAutoConfiguration?**
It tells Spring Boot to automatically configure beans based on dependencies. For example, if Spring Web is found, it configures DispatcherServlet automatically.

**4. What is @Configuration?**
It marks a class as a source of bean definitions. Methods annotated with @Bean create Spring-managed objects. It is used to configure custom beans manually.

**5. What is @Component?**
A generic Spring-managed bean. It is the base annotation for creating any custom component. Used for classes that don’t fit service or repository layers.

**6. What is @Service?**
Marks a business logic class. It helps Spring identify service layer beans and enables service-related features like transaction handling.

**7. What is @Controller?**
Used for MVC controllers that return views. It is used when we work with templates like Thymeleaf.

**8. What is @RestController?**
A combination of @Controller and @ResponseBody. It returns JSON responses directly. I use it for building REST APIs.

**9. What is @Repository?**
Marks data access classes. It converts database exceptions into Spring DataAccessException. It is used with JPA or JDBC access.

**10. What is @Bean?**
Defines a method that returns a bean managed by Spring. Useful for configuring third-party or custom objects that cannot use annotations.

**11. What is @Autowired?**
Injects dependencies automatically by type. I generally use constructor injection as it is recommended for testability and immutability.

**12. What is @Qualifier?**
Used with @Autowired to pick the exact bean when multiple beans of the same type exist. It avoids ambiguity during dependency injection.

**13. What is @Lazy?**
Makes a bean load only when first used instead of loading at startup. It improves performance when a bean is heavy or rarely used.

**14. What is @Value?**
Injects values from properties into fields. Often used for simple values like URLs, usernames, or limits.

**15. What is @PropertySource?**
Loads a custom properties file into the application environment. Useful when properties are kept outside application.properties.

**16. What is @ConfigurationProperties?**
Binds multiple configuration values into a Java class. It is used to group settings like database or mail configs. It avoids writing many @Value annotations.

**17. What is @Profile?**
Activates beans only for specific environments like dev, test, or prod. I use it to load different configurations for different environments.

**18. What is @Scope?**
Defines bean scope like singleton or prototype. Used when we need a new instance every time instead of a shared one.

**19. What is @RequestMapping?**
Defines the base URL for controllers or endpoints. Can set method type and path. It is the parent of GetMapping and others.

**20. What is @GetMapping?**
Handles HTTP GET requests. It is used to fetch data from the server.

**21. What is @PostMapping?**
Handles HTTP POST requests. It is used to create or submit data.

**22. What is @PutMapping?**
Handles HTTP PUT requests. It is used to update existing data.

**23. What is @DeleteMapping?**
Handles HTTP DELETE requests. It is used to delete records.

**24. What is @RequestBody?**
Maps incoming JSON from the request body to a Java object. Useful for POST and PUT APIs.

**25. What is @PathVariable?**
Extracts a value directly from the URL path. Used when the value is part of the endpoint like users 10.

**26. What is @RequestParam?**
Extracts values from query parameters like users id=10. Used when data comes after the question mark.

**27. What is @ControllerAdvice?**
Provides global exception handling for all controllers. I use it to return uniform error responses across all APIs.

**28. What is @ExceptionHandler?**
Handles specific exceptions inside a controller or ControllerAdvice. It gives full control over error messages.

**29. What is @Entity?**
Marks a class as a JPA entity mapped to a table. Each instance represents a database row.

**30. What is @Table?**
Specifies the table name and settings. Useful when the class name and table name differ.

**31. What is @Column?**
Specifies column properties like name, length, unique, or nullable. It is used to customize database mappings.

**32. What is @Transactional?**
Executes a method within a database transaction. If an exception occurs, it automatically rolls back changes. I mostly use it in service layer methods.

**33. What is @OneToOne?**
Defines a one to one relationship between two entities. Used when one record is linked to one other record.

**34. What is @OneToMany?**
Defines a one to many relationship. One parent entity has multiple child entities. Common in parent-child tables.

**35. What is @ManyToOne?**
Many child records point to one parent record. This is the inverse side of OneToMany.

**36. What is @ManyToMany?**
Defines a many to many relationship using a join table. Used when multiple records relate to multiple records.

## Miscellaneous

## Hibernate

## Security