**Spring Annotations**

**Commonly used annotations in Spring Core, Spring REST API, Spring Boot, Spring Data JPA, and Spring Cloud Microservices**

Spring Core Annotations:

1. @Component: Marks a Java class as a Spring component to be managed by the Spring container.

2. @Repository: Indicates that a class is a repository, typically used for database access.

3. @Service: Marks a class as a service component in the business layer.

4. @Controller: Marks a class as a controller in the presentation layer for handling web requests.

5. @Autowired: Injects dependencies automatically by type.

6. @Qualifier: Specifies the qualifier when multiple beans of the same type are available.

7. @Primary: Indicates the primary bean when multiple beans of the same type are present.

8. @Scope: Defines the scope of a bean (singleton, prototype, etc.).

9. @Value: Injects values from properties files, environment variables, or SpEL expressions.

10. @PostConstruct: Executes a method after bean initialization.

**Spring REST API Annotations:**

11. @RestController: Combines @Controller and @ResponseBody, used for RESTful web services.

12. @RequestMapping: Maps HTTP requests to handler methods.

13. @GetMapping, @PostMapping, @PutMapping, @DeleteMapping: Shortcut annotations for HTTP methods.

14. @PathVariable: Extracts values from the URI path.

15. @RequestParam: Binds parameters from the request URL.

16. @RequestBody: Binds the HTTP request body to a method parameter.

17. @ResponseBody: Returns the method result directly as an HTTP response body.

18. @ResponseStatus: Sets the HTTP response status.

19. @RestControllerAdvice: Global exception handler for REST controllers.

20. @CrossOrigin: Configures Cross-Origin Resource Sharing (CORS) for the controller or method.

**Spring Boot Annotations:**

21. @SpringBootApplication: Main annotation to mark the main class of a Spring Boot application.

22. @EnableAutoConfiguration:Enables Spring Boot's auto-configuration mechanism.

23. @ComponentScan:Scans for Spring components in specified packages.

24. @Configuration: Indicates that a class provides bean definitions.

25. @ConditionalOnProperty: Conditionally configures beans based on specified properties.

26. @ConditionalOnClass: Conditionally configures beans based on the presence of certain classes.

27. @ConditionalOnBean: Conditionally configures beans based on the presence of certain beans.

28. @EnableConfigurationProperties: Enables binding of configuration properties to beans.

29. @SpringBootTest:Integration test annotation for loading the entire application context.

30. @MockBean: Creates a mock bean for testing purposes.

Spring Data JPA Annotations:

31. @Entity: Marks a class as an entity in JPA.

32. @Table: Specifies the table name for an entity.

33. @Column: Maps a field to a database column.

34. @Id:Marks a field as the primary key.

35. @GeneratedValue: Configures automatic generation of primary key values.

36. @OneToOne, @OneToMany, @ManyToOne, @ManyToMany: Defines relationships between entities.

37. @JoinColumn: Specifies the join column for a relationship.

38. @Repository: Indicates that a class is a repository for database access.

39. @Transactional: Marks a method, class, or interface as transactional.

40. @Query: Declares a query using JPQL or native SQL.

**Spring Cloud Microservices Annotations:**

41. @EnableDiscoveryClient: Makes a Spring Boot application discoverable by other services using service discovery.

42. @EnableEurekaClient: Enables a Spring Boot application to act as a Eureka client.

43. @EnableFeignClients: Enables Feign client in a Spring Boot application.

44. @EnableCircuitBreaker: Enables circuit breaker functionality.

45. @EnableHystrix: Enables Hystrix circuit breaker functionality.

46. @EnableZuulProxy: Enables the Zuul proxy and configures it to forward requests to registered services.

47. @EnableConfigServer: Enables the Spring Cloud Config server functionality.

48. @EnableDiscoveryClient: Enables service registration and discovery in the Spring Cloud application.

49. @FeignClient: Declares a Feign client interface.

50. @HystrixCommand: Marks a method to be wrapped in a circuit breaker.