Spring Boot leverages a wide array of annotations from the Spring framework, each designed to simplify and automate configuration, dependency injection, and other aspects of application development. Below is a comprehensive list of 50 key annotations used in Spring Boot, along with detailed explanations.

Core Annotations

1. **@SpringBootApplication**
   - **Description: Combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`. It is the primary annotation that marks the main class of a Spring Boot application.**
   - **Usage: Applied on the main application class to bootstrap a Spring Boot application.**
2. **@Configuration**
   - **Description: Indicates that a class declares one or more `@Bean` methods and may be processed by the Spring container to generate bean definitions.**
   - **Usage: Applied on classes to define beans and configuration.**
3. **@Component**
   - **Description: Marks a class as a Spring-managed component, allowing Spring to automatically detect and register it as a bean.**
   - **Usage: Applied on any Spring-managed component class.**
4. **@Service**
   - **Description: A specialization of `@Component`, denoting a service layer class.**
   - **Usage: Applied on service layer classes.**
5. **@Repository**
   - **Description: A specialization of `@Component`, used to indicate that a class is a Data Access Object (DAO).**
   - **Usage: Applied on DAO classes.**
6. **@Controller**
   - **Description: A specialization of `@Component`, marking a class as a Spring MVC controller.**
   - **Usage: Applied on web controller classes.**
7. **@RestController**
   - **Description: A combination of `@Controller` and `@ResponseBody`, used to create RESTful web services.**
   - **Usage: Applied on REST API controller classes.**
8. **@RequestMapping**
   - **Description: Maps HTTP requests to handler methods of MVC and REST controllers.**
   - **Usage: Applied at both class and method levels to define request paths.**
9. **@GetMapping**
   - **Description: A shortcut for `@RequestMapping(method = RequestMethod.GET)`; maps HTTP GET requests.**
   - **Usage: Applied on methods to handle GET requests.**

10. **@PostMapping**
    - **Description: A shortcut for `@RequestMapping(method = RequestMethod.POST)`; maps HTTP POST requests.**
    - **Usage: Applied on methods to handle POST requests.**
11. **@PutMapping**
    - **Description: A shortcut for `@RequestMapping(method = RequestMethod.PUT)`; maps HTTP PUT requests.**
    - **Usage: Applied on methods to handle PUT requests.**
12. **@DeleteMapping**
    - **Description: A shortcut for `@RequestMapping(method = RequestMethod.DELETE)`; maps HTTP DELETE requests.**
    - **Usage: Applied on methods to handle DELETE requests.**
13. **@PatchMapping**
    - **Description: A shortcut for `@RequestMapping(method = RequestMethod.PATCH)`; maps HTTP PATCH requests.**
    - **Usage: Applied on methods to handle PATCH requests.**

Dependency Injection and Bean Management

14. **@Autowired**
    - **Description: Marks a constructor, field, setter method, or config method as to be autowired by Spring's dependency injection.**
    - **Usage: Applied on dependencies that should be injected by Spring.**
15. **@Qualifier**
    - **Description: Used in conjunction with `@Autowired` to specify which bean should be injected when multiple candidates exist.**
    - **Usage: Applied on fields, constructors, or setter methods.**
16. **@Primary**
    - **Description: Indicates that a bean should be given preference when multiple candidates are qualified to be autowired.**
    - **Usage: Applied on beans defined in `@Configuration` or `@Component` classes.**
17. **@Bean**
    - **Description: Indicates that a method produces a bean to be managed by the Spring container.**
    - **Usage: Applied on methods in configuration classes.**
18. **@Lazy**
    - **Description: Marks a bean to be lazily initialized, meaning it is not created until it is needed.**
    - **Usage: Applied on beans and injection points.**
19. **@Scope**
    - **Description: Specifies the scope of a bean, such as `singleton`, `prototype`, `request`, or `session`.**
    - **Usage: Applied on beans or component classes.**
20. **@Value**
    - **Description: Used to inject property values into Spring-managed beans.**
    - **Usage: Applied on fields, setter methods, or constructors.**

21. **@PostConstruct**
    - **Description: Marks a method to be called after the bean has been initialized.**
    - **Usage: Applied on methods in bean classes.**
22. **@PreDestroy**
    - **Description: Marks a method to be called before the bean is destroyed.**
    - **Usage: Applied on methods in bean classes.**

Data Access and Transaction Management

23. **@Transactional**
    - **Description: Indicates that a method or class should be executed within a transaction context.**
    - **Usage: Applied on service methods or classes that involve database operations.**
24. **@Entity**
    - **Description: Marks a class as a JPA entity, meaning it will be mapped to a database table.**
    - **Usage: Applied on domain model classes.**
25. **@Id**
    - **Description: Specifies the primary key of an entity.**
    - **Usage: Applied on fields in entity classes.**
26. **@GeneratedValue**
    - **Description: Specifies how the primary key should be generated (e.g., auto, sequence).**
    - **Usage: Applied on primary key fields in entity classes.**
27. **@Table**
    - **Description: Specifies the table name in the database for a particular entity.**
    - **Usage: Applied on entity classes.**
28. **@Column**
    - **Description: Specifies the mapping between a field and a database column.**
    - **Usage: Applied on fields in entity classes.**
29. **@OneToOne**
    - **Description: Defines a one-to-one relationship between two entities.**
    - **Usage: Applied on fields in entity classes.**
30. **@OneToMany**
    - **Description: Defines a one-to-many relationship between two entities.**
    - **Usage: Applied on fields in entity classes.**
31. **@ManyToOne**
    - **Description: Defines a many-to-one relationship between two entities.**
    - **Usage: Applied on fields in entity classes.**
32. **@ManyToMany**
    - **Description: Defines a many-to-many relationship between two entities.**
    - **Usage: Applied on fields in entity classes.**
33. **@JoinColumn**
    - **Description: Specifies the foreign key column in a relationship mapping.**
    - **Usage: Applied on fields in entity classes that represent relationships.**

34. **@Fetch**
    - **Description: Specifies the fetching strategy (e.g., EAGER, LAZY) for a relationship.**
    - **Usage: Applied on relationship fields in entity classes.**
35. **@Query**
    - **Description: Defines a JPQL or SQL query in a repository method.**
    - **Usage: Applied on methods in repository interfaces.**
36. **@Modifying**
    - **Description: Indicates that a repository query method is an update or delete operation.**
    - **Usage: Applied on methods in repository interfaces.**

Validation and Exception Handling

37. **@Valid**
    - **Description: Marks a method parameter or return value for validation.**
    - **Usage: Applied on method parameters or return values.**
38. **@NotNull**
    - **Description: Ensures that a field or parameter is not null.**
    - **Usage: Applied on fields or method parameters.**
39. **@NotEmpty**
    - **Description: Ensures that a field or parameter is not empty (for collections, arrays, or strings).**
    - **Usage: Applied on fields or method parameters.**
40. **@Size**
    - **Description: Specifies the size constraints for a field or parameter (e.g., string length, collection size).**
    - **Usage: Applied on fields or method parameters.**
41. **@Min**
    - **Description: Specifies the minimum value for a numeric field or parameter.**
    - **Usage: Applied on numeric fields or method parameters.**
42. **@Max**
    - **Description: Specifies the maximum value for a numeric field or parameter.**
    - **Usage: Applied on numeric fields or method parameters.**
43. **@ExceptionHandler**
    - **Description: Defines a method to handle exceptions thrown by controller methods.**
    - **Usage: Applied on methods in `@Controller` or `@RestController` classes.**
44. **@ControllerAdvice**
    - **Description: Allows centralized exception handling across multiple controllers.**
    - **Usage: Applied on classes that handle exceptions globally.**

Security and Scheduling

45. **@Secured**

- **Description: Specifies that a method can only be invoked by users with specific roles.**
    - **Usage: Applied on methods in service or controller classes.**

46. **@PreAuthorize**
    - **Description: A more flexible alternative to `@Secured`, allowing complex security expressions.**
    - **Usage: Applied on methods in service or controller classes.**
47. **@Scheduled**
    - **Description: Marks a method to be scheduled for execution at a fixed interval or cron expression.**
    - **Usage: Applied on methods in service or component classes.**
48. **@Async**
    - **Description: Indicates that a method should run asynchronously in a separate thread.**
    - **Usage: Applied on methods in service or component classes.**
49. **@EnableScheduling**
    - **Description: Enables support for scheduling tasks.**
    - **Usage: Applied on configuration classes.**

50

4o
Continue generating