Top 50 Spring Boot Annotations Explained

1. **@SpringBootApplication**
    - **Explanation**: Marks the main class of a Spring Boot application and combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`.

2. **@RestController**
    - **Explanation**: Indicates that a class is a RESTful controller, combining `@Controller` and `@ResponseBody`.

3. **@RequestMapping**
    - **Explanation**: Maps web requests to specific handler methods, can be applied at the class or method level.

4. **@GetMapping**
    - **Explanation**: Shortcut for `@RequestMapping(method = RequestMethod.GET)`, maps HTTP GET requests to handler methods.

5. **@PostMapping**
    - **Explanation**: Shortcut for `@RequestMapping(method = RequestMethod.POST)`, maps HTTP POST requests to handler methods.

6. **@PutMapping**
    - **Explanation**: Shortcut for `@RequestMapping(method = RequestMethod.PUT)`, maps HTTP PUT requests to handler methods.

7. **@DeleteMapping**

- **Explanation**: Shortcut for `@RequestMapping(method = RequestMethod.DELETE)`, maps HTTP DELETE requests to handler methods.

8. **@PatchMapping**
   - **Explanation**: Shortcut for `@RequestMapping(method = RequestMethod.PATCH)`, maps HTTP PATCH requests to handler methods.

9. **@Autowired**
   - **Explanation**: Automatically wires dependencies in Spring beans, can be applied to fields, constructors, or methods.

10. **@Component**
    - **Explanation**: Marks a Java class as a Spring bean.

11. **@Service**
    - **Explanation**: Indicates that a class contains business logic, a specialization of `@Component`.

12. **@Repository**
    - **Explanation**: Indicates that a class is a data access object (DAO), a specialization of `@Component`.

13. **@Controller**
    - **Explanation**: Marks a class as a web controller in a Spring MVC application.

14. **@RequestBody**
    - **Explanation**: Maps the HTTP request body to a method parameter in a controller.

15. **@ResponseBody**

- **Explanation**: Maps the return value of a method to the HTTP response body.

16. **@PathVariable**
    - **Explanation**: Binds a method parameter to a URI template variable.

17. **@RequestParam**
    - **Explanation**: Binds a method parameter to a web request parameter.

18. **@RequestHeader**
    - **Explanation**: Binds a method parameter to a web request header.

19. **@CookieValue**
    - **Explanation**: Binds a method parameter to a cookie value.

20. **@ModelAttribute**
    - **Explanation**: Binds a method parameter or method return value to a named model attribute and exposes it to a web view.

21. **@SessionAttributes**
    - **Explanation**: Indicates the names of model attributes that should be stored in the session.

22. **@ExceptionHandler**
    - **Explanation**: Indicates the method to be invoked when an exception is thrown.

23. **@ControllerAdvice**
    - **Explanation**: Allows the handling of exceptions across the whole application in a single global handler.

24. **@CrossOrigin**
    - **Explanation**: Enables Cross-Origin Resource Sharing (CORS) on a method or class.

25. **@Configuration**
    - **Explanation**: Declares a class as a configuration class, typically used with `@Bean` methods.

26. **@Bean**
    - **Explanation**: Indicates that a method produces a bean to be managed by the Spring container.

27. **@Primary**
    - **Explanation**: Indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency.

28. **@Value**
    - **Explanation**: Injects values from properties files or other sources into Spring beans.

29. **@PropertySource**
    - **Explanation**: Provides a convenient and declarative mechanism for adding a set of PropertySources to Spring's Environment.

30. **@EnableAutoConfiguration**
    - **Explanation**: Enables Spring Boot's auto-configuration mechanism.

31. **@Conditional**
    - **Explanation**: Conditionally includes or excludes beans based on a condition.

32. **@Profile**
    - **Explanation**: Specifies the profiles a bean is eligible for registration in, controlling which beans are loaded in which environments.

33. **@Scope**
    - **Explanation**: Configures the scope of a bean, such as singleton, prototype, request, session, etc.

34. **@Lazy**
    - **Explanation**: Delays the initialization of a bean until it is first requested.

35. **@Async**
    - **Explanation**: Indicates that a method should be executed asynchronously.

36. **@Scheduled**
    - **Explanation**: Schedules a method to be run at fixed intervals.

37. **@EnableScheduling**
    - **Explanation**: Enables Spring's scheduled task execution capability.

38. **@Transactional**
    - **Explanation**: Demarcates transactional boundaries on a method or class.

39. **@EnableTransactionManagement**
    - **Explanation**: Enables Spring's annotation-driven transaction management.

40. **@Entity**
    - **Explanation**: Specifies that a class is an entity and is mapped to a database table.

41. **@Table**
    - **Explanation**: Specifies the primary table for the annotated entity.

42. **@Id**
    - **Explanation**: Specifies the primary key of an entity.

43. **@GeneratedValue**
    - **Explanation**: Specifies the generation strategy for the primary key values.

44. **@Column**
    - **Explanation**: Specifies the mapped column for a persistent property or field.

45. **@OneToMany, @ManyToOne, @OneToOne, @ManyToMany**
    - **Explanation**: Defines various types of relationships between entities.

46. **@JoinColumn**
    - **Explanation**: Specifies the foreign key column for a relationship.

47. **@JsonIgnoreProperties**
    - **Explanation**: Specifies properties to ignore during JSON serialization and deserialization.

48. **@JsonProperty**
    - **Explanation**: Specifies the property name to be used during JSON serialization and deserialization.

49. **@SpringBootTest**
    - **Explanation**: Used to bootstrap the entire container and start the full Spring context for integration tests.

50. **@Test**
    - **Explanation**: Marks a method as a test method in a JUnit test class.