**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.

@GetMapping("/{id}")

public String getCustomerById(@PathVariable("id") int customerId) {

return "Customer ID: " + customerId;

}

1. **@RequestParam**
   * **Explanation**: Binds a method parameter to a web request parameter.
   * is used to **extract query parameters** (the key-value pairs in a URL **after ?**) from the HTTP request.

Request: GET /accounts?branch=Hyd&active=true

@GetMapping("/accounts")

public String getAccounts(@RequestParam String branch, @RequestParam boolean active) {

return "Branch: " + branch + ", Active: " + active;

}

1. **@RequestHeader**
   * **Explanation**: Binds a method parameter to a web request header.
2. **@CookieValue**
   * **Explanation**: Binds a method parameter to a cookie value.
3. **@ModelAttribute**
   * **Explanation**: Binds a method parameter or method return value to a named model attribute and exposes it to a web view.
4. **@SessionAttributes**
   * **Explanation**: Indicates the names of model attributes that should be stored in the session.
5. **@ExceptionHandler**
   * **Explanation**: Indicates the method to be invoked when an exception is thrown.
6. **@ControllerAdvice**
   * **Explanation**: Allows the handling of exceptions across the whole application in a single global handler.
7. **@CrossOrigin**
   * **Explanation**: Enables Cross-Origin Resource Sharing (CORS) on a method or class.
8. **@Configuration**
   * **Explanation**: Declares a class as a configuration class, typically used with @Bean methods.
9. **@Bean**
   * **Explanation**: Indicates that a method produces a bean to be managed by the Spring container.
10. **@Primary**

* **Explanation**: When you have **multiple beans of the same type**, Spring doesn’t know which one to inject automatically.  
  You can mark **one bean as the default** using @Primary.
* Then, **that bean is chosen automatically** unless a specific @Qualifier is used.

@Service

@Primary

public class EmailNotificationService implements NotificationService { }

@Service

public class SMSNotificationService implements NotificationService { }

@Component

public class NotificationController {

@Autowired

private NotificationService notificationService;

// ✅ Injects EmailNotificationService (the @Primary one)

}

1. @Qualifier
   * When you have **multiple beans** of the same type in Spring’s ApplicationContext, the framework doesn’t know which one to inject — that causes an **ambiguity**.
   * 👉 @Qualifier helps **resolve this ambiguity** by specifying **which bean** to inject.

@Autowired

@Qualifier("emailNotificationService")

private NotificationService notificationService;

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

@Value("${bank.name}")

private String name;

1. **@PropertySource**

* **Explanation**: Provides a convenient and declarative mechanism for adding a set of PropertySources to Spring’s Environment.
* Spring Boot automatically loads **application.properties** and **application.yml** —  
  so you **don’t need** @PropertySource for those.
* You only use @PropertySource when you have **extra or custom property files**.

1. **@EnableAutoConfiguration**
   * **Explanation**: Enables Spring Boot’s auto-configuration mechanism.
2. **@Conditional**
   * **Explanation**: Conditionally includes or excludes beans based on a condition.
3. **@Profile**
   * **Explanation**: Specifies the profiles a bean is eligible for registration in, controlling which beans are loaded in which environments.
4. **@Scope**
   * **Explanation**: Configures the scope of a bean, such as singleton, prototype, request, session, etc.
5. **@Lazy**
   * **Explanation**: Delays the initialization of a bean until it is first requested.
6. **@Async**
   * **Explanation**: Indicates that a method should be executed asynchronously.
7. **@Scheduled**
   * **Explanation**: Schedules a method to be run at fixed intervals.
   * Before using @Scheduled, you need to **enable scheduling** in your Spring configuration: @EnableScheduling

@Component

public class ReportService {

@Scheduled(fixedRate = 5000)

public void generateReport() {

System.out.println("Report generated at: " + new java.util.Date());

}

}

* fixedRate = 5000 → runs every **5 seconds**, counted from the **start time** of the last execution.
* @Scheduled(cron = "0 0/1 \* \* \* ?")

second minute hour day-of-month month day-of-week

| **Expression** | **Meaning** |
| --- | --- |
| 0 0 \* \* \* ? | Every hour at minute 0 |
| 0 0 9-17 \* \* MON-FRI | Every hour between 9 AM and 5 PM on weekdays |
| 0 0/15 \* \* \* ? | Every 15 minutes |

1. **@EnableScheduling**
   * **Explanation**: Enables Spring’s scheduled task execution capability.
2. **@Transactional**
   * **Explanation**: Demarcates transactional boundaries on a method or class.
3. **@EnableTransactionManagement**
   * **Explanation**: Enables Spring’s annotation-driven transaction management.
4. **@Entity**
   * **Explanation**: Specifies that a class is an entity and is mapped to a database table.
5. **@Table**
   * **Explanation**: Specifies the primary table for the annotated entity.
6. **@Id**
   * **Explanation**: Specifies the primary key of an entity.
7. **@GeneratedValue**
   * **Explanation**: Specifies the generation strategy for the primary key values.
8. **@Column**
   * **Explanation**: Specifies the mapped column for a persistent property or field.
9. **@OneToMany, @ManyToOne, @OneToOne, @ManyToMany**
   * **Explanation**: Defines various types of relationships between entities.
10. **@JoinColumn**
    * **Explanation**: Specifies the foreign key column for a relationship.
11. **@JsonIgnoreProperties**
    * **Explanation**: Specifies properties to ignore during JSON serialization and deserialization.
12. **@JsonProperty**
    * **Explanation**: Specifies the property name to be used during JSON serialization and deserialization.
13. **@SpringBootTest**
    * **Explanation**: Used to bootstrap the entire container and start the full Spring context for integration tests.
14. **@Test**
    * **Explanation**: Marks a method as a test method in a JUnit test class.