

Important Spring Annotations

Comprehensive List of Spring Annotations by Categories

1. Core Annotations
2. Spring MVC Annotations
3. Data Access Annotations (Spring Data)
4. Aspect-Oriented Programming (AOP) Annotations
5. Scheduling and Async Annotations
6. Spring Boot-Specific Annotations
7. Security Annotations
8. Cloud and Microservices Annotations
9. Testing Annotations

Comprehensive List of Spring Annotations by Categories

Spring offers a vast array of annotations across its modules. Below is a categorized list covering **core**, **MVC**, **data access**, **AOP**, and other key areas of Spring.

1. Core Annotations

Annotation	Purpose
<code>@Component</code>	Marks a class as a Spring-managed component. Generic stereotype for Spring beans.
<code>@Service</code>	Specialization of <code>@Component</code> , used for service-layer beans.
<code>@Repository</code>	Specialization of <code>@Component</code> , used for data access and exception translation.
<code>@Configuration</code>	Marks a class as a source of Spring bean definitions.
<code>@Bean</code>	Declares a bean in a <code>@Configuration</code> class.
<code>@Autowired</code>	Automatically wires dependencies by type.
<code>@Qualifier</code>	Used with <code>@Autowired</code> to resolve ambiguity when multiple beans of the same type exist.
<code>@Primary</code>	Indicates the primary bean to use when multiple beans of the same type exist.
<code>@Scope</code>	Defines the scope of a bean (<code>singleton</code> , <code>prototype</code> , etc.).
<code>@Lazy</code>	Specifies lazy initialization for a bean.

@Value	Injects values from property files or environment variables.
@PostConstruct	Indicates a method to be executed after bean initialization.
@PreDestroy	Indicates a method to be executed before bean destruction.
@EventListener	Handles Spring application events.
@PropertySource	Specifies the location of property files.

2. Spring MVC Annotations

Annotation	Purpose
@Controller	Marks a class as a Spring MVC controller.
@RestController	Combines @Controller and @ResponseBody . Used for RESTful web services.
@RequestMapping	Maps HTTP requests to handler methods or classes.
@GetMapping	Shortcut for @RequestMapping with GET method.
@PostMapping	Shortcut for @RequestMapping with POST method.
@PutMapping	Shortcut for @RequestMapping with PUT method.
@DeleteMapping	Shortcut for @RequestMapping with DELETE method.
@PatchMapping	Shortcut for @RequestMapping with PATCH method.
@RequestParam	Binds query parameters to method arguments.
@PathVariable	Binds URL path variables to method arguments.
@RequestBody	Maps the request body to a method argument.
@ResponseBody	Maps the return value of a method to the HTTP response body.
@ModelAttribute	Binds a model attribute to a method parameter or return value.
@SessionAttributes	Specifies attributes to store in the HTTP session.
@CrossOrigin	Enables Cross-Origin Resource Sharing (CORS) for RESTful services.
@ExceptionHandler	Handles exceptions thrown by controller methods.
@InitBinder	Customizes data binding for request parameters.
@ControllerAdvice	A global exception handler for all controllers.

3. Data Access Annotations (Spring Data)

Annotation	Purpose
------------	---------

<code>@Transactional</code>	Marks a method or class to participate in a transaction.
<code>@PersistenceContext</code>	Injects a JPA <code>EntityManager</code> .
<code>@Repository</code>	Marks a class as a DAO component with exception translation.
<code>@Query</code>	Defines custom queries in Spring Data repositories.
<code>@Modifying</code>	Marks a query method as an update or delete operation.
<code>@EnableJpaRepositories</code>	Enables scanning of JPA repositories.
<code>@NamedQuery</code>	Defines a named query at the entity level.
<code>@EnableTransactionManagement</code>	Enables annotation-driven transaction management.
<code>@Id</code>	Marks a field as the primary key in JPA.
<code>@GeneratedValue</code>	Specifies the generation strategy for primary keys.
<code>@Entity</code>	Marks a class as a JPA entity.
<code>@Table</code>	Specifies the database table for a JPA entity.
<code>@Column</code>	Specifies the mapping of a field to a database column.

4. Aspect-Oriented Programming (AOP) Annotations

Annotation	Purpose
<code>@Aspect</code>	Marks a class as an aspect for defining cross-cutting concerns.
<code>@Before</code>	Defines advice to run before a method execution.
<code>@After</code>	Defines advice to run after a method execution.
<code>@Around</code>	Defines advice that wraps around a method execution.
<code>@AfterReturning</code>	Defines advice to run after a method returns successfully.
<code>@AfterThrowing</code>	Defines advice to run after a method throws an exception.
<code>@Pointcut</code>	Declares reusable pointcut expressions.

5. Scheduling and Async Annotations

Annotation	Purpose
<code>@Scheduled</code>	Schedules tasks to run periodically or at specific times.
<code>@EnableScheduling</code>	Enables scheduling support in the application.
<code>@Async</code>	Marks a method to be executed asynchronously.

<code>@EnableAsync</code>	Enables asynchronous processing in the application.
---------------------------	---

6. Spring Boot-Specific Annotations

Annotation	Purpose
<code>@SpringBootApplication</code>	Combines <code>@Configuration</code> , <code>@EnableAutoConfiguration</code> , and <code>@ComponentScan</code> .
<code>@EnableAutoConfiguration</code>	Enables Spring Boot's auto-configuration feature.
<code>@RestController</code>	Combines <code>@Controller</code> and <code>@ResponseBody</code> for RESTful APIs.
<code>@ConditionalOnProperty</code>	Enables configuration based on the presence of a specific property.
<code>@ConditionalOnMissingBean</code>	Configures a bean only if a specific bean is not already defined.
<code>@Value</code>	Injects properties from configuration files or environment variables.
<code>@EnableConfigurationProperties</code>	Binds configuration properties to POJOs.

7. Security Annotations

Annotation	Purpose
<code>@EnableWebSecurity</code>	Enables Spring Security for the application.
<code>@Secured</code>	Specifies role-based security for methods.
<code>@PreAuthorize</code>	Adds pre-authorization checks to methods.
<code>@PostAuthorize</code>	Adds post-authorization checks to methods.
<code>@RolesAllowed</code>	Specifies allowed roles for accessing a method.

8. Cloud and Microservices Annotations

Annotation	Purpose
<code>@EnableDiscoveryClient</code>	Enables service discovery for Spring Cloud applications.
<code>@FeignClient</code>	Declares a Feign client for HTTP-based microservice communication.
<code>@EnableCircuitBreaker</code>	Enables circuit breaker functionality.
<code>@HystrixCommand</code>	Annotates methods with fallback mechanisms for resilience.

<code>@RefreshScope</code>	Refreshes bean definitions at runtime when configuration changes.
----------------------------	---

9. Testing Annotations

Annotation	Purpose
<code>@SpringBootTest</code>	Loads the full Spring context for integration testing.
<code>@WebMvcTest</code>	Focuses testing on Spring MVC components (e.g., controllers).
<code>@MockBean</code>	Creates mock beans in the Spring context.
<code>@TestConfiguration</code>	Defines test-specific configuration classes.
<code>@DataJpaTest</code>	Tests JPA repositories with an in-memory database.