Important Spring 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
`@Component`	Marks a class as a Spring-managed component. Generic stereotype for Spring beans.
`@Service`	Specialization of `@Component`, used for service-layer beans.
`@Repository`	Specialization of `@Component`, used for data access and exception translation.
`@Configuration`	Marks a class as a source of Spring bean definitions.
`@Bean`	Declares a bean in a `@Configuration` class.
`@Autowired`	Automatically wires dependencies by type.
`@Qualifier`	Used with `@Autowired` to resolve ambiguity when multiple beans of the same type exist.
`@Primary`	Indicates the primary bean to use when multiple beans of the same type exist.
`@Scope`	Defines the scope of a bean (`singleton`, `prototype`, etc.).
`@Lazy`	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
`@Transactional`	Marks a method or class to participate in a transaction.
`@PersistenceContext`	Injects a JPA `EntityManager`.
`@Repository`	Marks a class as a DAO component with exception translation.
`@Query`	Defines custom queries in Spring Data repositories.

`@Modifying`	Marks a query method as an update or delete operation.
`@EnableJpaRepositories`	Enables scanning of JPA repositories.
`@NamedQuery`	Defines a named query at the entity level.
`@EnableTransactionManagement`	Enables annotation-driven transaction management.
`@Id`	Marks a field as the primary key in JPA.
`@GeneratedValue`	Specifies the generation strategy for primary keys.
`@Entity`	Marks a class as a JPA entity.
`@Table`	Specifies the database table for a JPA entity.
`@Column`	Specifies the mapping of a field to a database column.

4. Aspect-Oriented Programming (AOP) Annotations

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

5. Scheduling and Async Annotations

Annotation	Purpose
`@Scheduled`	Schedules tasks to run periodically or at specific times.
`@EnableScheduling`	Enables scheduling support in the application.
`@Async`	Marks a method to be executed asynchronously.
`@EnableAsync`	Enables asynchronous processing in the application.

6. Spring Boot-Specific Annotations

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

7. Security Annotations

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

8. Cloud and Microservices Annotations

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

`@RefreshScope`	Refreshes bean definitions at runtime when configuration changes.

9. Testing Annotations

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