Frequently Used Annotations in Spring Boot Applications

This article contains a non-comprehensive list of frequently used annotations in Spring Boot applications. This list is meant to be a quick lookup, for detailed and comprehensive information please read official Java docs and documentation.

Core Spring

- @Bean (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Bean.html) - Annotated method produces a bean managed by the Spring IoC container
- Stereotype annotations
  - @Component (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Component.html) - Marks annotated class as a bean found by the component-scanning and loaded into the application context
  - @Controller (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Controller.html) - Marks annotated class as a bean for Spring MVC containing request handler
  - @RestController (https://docs.spring.io/spring-framework/docs/current/javadoc-

api/org/springframework/web/bind/annotation/RestController.html) - Marks annotated class as a `@Controller` bean and adds `@ResponseBody` to serialize returned results as messages

- @Configuration (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Configuration.html) - Marks annotated class as a Java configuration defining beans
- @Service (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Service.html) - Marks annotated class as a bean (as convention usually containing business logic)
- @Repository (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Repository.html) - Marks annotated class as a bean (as convention usually providing data access) and adds auto-translation from `SQLException` to `DataAccessExceptions`

Bean State

- @PostConstruct (https://docs.oracle.com/javaee/7/api/javax/annotation/PostConstruct.html) - Annotated method is executed after dependency injection is done to perform initialization
- @PreDestroy (https://docs.oracle.com/javaee/7/api/javax/annotation/PreDestroy.html) - Annotated method is executed before the bean is destroyed, e.g. on the shutdown

Configuration

- @Import (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Import.html) - Imports one or

more Java configuration classes `@Configuration`

- @PropertySource (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/PropertySource.html) - Indicates the location of `applicaiton.properties` file to add key-value pairs to Spring `Environment`
- @Value (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Value.html) - Annotated fields and parameters values will be injected
- @ComponentScan (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/ComponentScan.html) - Configures component scanning `@Compenent`, `@Service`, etc.

Bean Properties

- @Lazy (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Lazy.html) - Annotated bean will be lazily initialized on the first usage
- @Profile (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Profile.html) - Indicates that beans will be only initialized if the defined profiles are active
- @Scope (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Scope.html) - Defines bean creation scope, e.g. prototype, singleton, etc.
- @DependsOn (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/DependsOn.html) - Explicitly defines a dependency to other beans in terms of creation order
- @Order (https://docs.spring.io/spring-framework/docs/current/javadoc-

@Order (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/core/annotation/Order.html) - Defines sorting order if injecting a list of beans, but it does not resolve the priority if only a single bean is expected

- @Primary (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Primary.html) - Annotated bean will be picked if multiple beans can be autowired
- @Conditional (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Conditional.html)- Annotated bean is created only if conditions are satisfied
    - Additionally available in Spring Boot:
        - @ConditionalOnBean (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/condition/ConditionalOnBean.html)
        - @ConditionalOnMissingBean (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/condition/ConditionalOnMissingBean.html)
        - @ConditionalOnClass (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/condition/ConditionalOnClass.html)
        - @ConditionalOnMissingClass (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/condition/ConditionalOnMissingClass.html)
        - @ConditionalOnProperty (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/condition/ConditionalOnProperty.html)
        - @ConditionalOnMissingProperty (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/condition/ConditionalOnProperty.html)

Bean Injection

- @Autowired (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html) - Beans are injected into annotated setters, fields, or constructor params.
- @Qualifier (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Qualifier.html) - Specifies the name of a bean as an additional condition to identify a unique candidate for autowiring

Spring Boot

- @SpringBootConfiguration (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/SpringBootConfiguration.html) - Indicates Spring Boot application `@Configuration`
- @EnableAutoConfiguration (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/EnableAutoConfiguration.html) - Enables application context auto-configuration to provide possibly needed beans based on the classpath
- @ConfigurationProperties (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/context/properties/ConfigurationProperties.html) - Provides external binding of key value properties
- @ConfigurationPropertiesScan (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/context/properties/ConfigurationPropertiesScan.html) - Enables auto-detection of `@ConfigurationProperties` classes
- @SpringBootApplication (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/SpringBootA

pplication.html) - Combination of `@SpringBootConfiguration`, `@EnableAutoConfiguration`, `@ConfigurationPropertiesScan` and `@ComponentScan`

- @EntityScan (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/domain/EntityScan.html) - Configures base packages to scan for entity classes
- @EnableJpaRepositories (https://docs.spring.io/spring-data/data-jpa/docs/current/api/org/springframework/data/jpa/repository/config/EnableJpaRepositories.html) - Enables auto-configuration of jpa repositories

Spring Boot Tests

- @SpringBootTest (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/context/SpringBootTest.html) - Annotated test class will load the entire application context for integration tests
- @WebMvcTest (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/autoconfigure/web/servlet/WebMvcTest.html) - Annotated test class will load only the web layer (service and data layer are ignored)
- @DataJpaTest (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/autoconfigure/orm/jpa/DataJpaTest.html) - Annotated class will load only the JPA components
- @MockBean (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/mock/mockito/MockBean.html) - Marks annotated field as a mock and loads it as a bean into the application context
- @SpyBean (https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/mock/mockito/SpyBea

n.html) - Allows partial mocking of beans

- @Mock (https://www.javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mock.html) - Defines the annotated field as a mock

Spring Test

- @ContextConfiguration (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/context/ContextConfiguration.html) - Defines `@Configuration` to load application context for integration test
- @ExtendWith (https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/extension/ExtendWith.html) - Defines extensions to execute the tests with, e.g. MockitoExtension
- @SpringJUnitConfig (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/context/junit/jupiter/SpringJUnitConfig.html) - Combines `@ContextConfiguration` and `@ExtendWith(SpringExtension.class)`
- @TestPropertySource (https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/test/context/TestPropertySource.html) - Defines the location of property files used in integration tests
- @DirtiesContext (https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/test/annotation/DirtiesContext.html) - Indicates that annotated tests dirty the application context and will be cleaned after each test
- @ActiveProfiles (https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/test/context/ActiveProfiles.html) - Defines which active bean definition should be loaded when initializing the test application context
- @Sql (https://docs.spring.io/spring-framework/docs/current/javadoc-

- @Sql (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/context/jdbc/Sql.html) - Allows defining SQL scripts and statements to be executed before and after tests

Transactions

- @EnableTransactionManagement (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/transaction/annotation/EnableTransactionManagement.html) - Enables annotation-driven transaction declaration `@Transactional`
- @Transactional (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/transaction/annotation/Transactional.html) - Annotated methods will be executed in a transactional manner

Spring JPA and Hibernate

- @Id (https://docs.oracle.com/javaee/7/api/javax/persistence/Id.html) - Marks annotated field as a primary key of an entity
- @GeneratedValue (https://docs.oracle.com/javaee/7/api/javax/persistence/GeneratedValue.html) - Provides generation strategy of primary keys
- @Entity (https://docs.oracle.com/javaee/7/api/javax/persistence/Entity.html) - Marks annotated class as an entity
- @Column (https://docs.oracle.com/javaee/7/api/javax/persistence/Column.html) - Provides additional configuration for a field, e.g. column name
- @Table (https://docs.oracle.com/javaee/7/api/javax/persistence/Table.html) - Provides additional configuration for an entity, e.g. table name
- @PersistenceContext

(https://docs.oracle.com/javaee/7/api/javax/persistence/PersistenceContext.html) - `EntityManger` is injected into annotated setters and fields

- @Embedded (https://docs.oracle.com/javaee/7/api/javax/persistence/Embedded.html) - Annotated field is instantiated as a value of an `Embeddable` class
- @Embeddable (https://docs.oracle.com/javaee/7/api/javax/persistence/Embeddable.html) - Instances of an annotated class are stored as part of an entity
- @EmbeddedId (https://docs.oracle.com/javaee/7/api/javax/persistence/EmbeddedId.html) - Marks annotated property as a composite key mapped by an embeddable class
- @AttributeOverride (https://docs.oracle.com/javaee/7/api/javax/persistence/AttributeOverride.html) - Overrides the default mapping of a field
- @Transient (https://docs.oracle.com/javaee/7/api/javax/persistence/Transient.html) - Annotated field is not persistent
- @CreationTimestamp (https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/annotations/CreationTimestamp.html) - Annotated field contains the timestamp when an entity was stored for the first time
- @UpdateTimestamp (https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/annotations/UpdateTimestamp.html) - Annotated field contains the timestamp when an entity was updated last time
- @ManyToOne (https://docs.oracle.com/javaee/7/api/javax/persistence/ManyToOne.html) - Indicates N:1 relationship, the entity containing annotated field has a single

relation to an entity of other class, but the other class has multiple relations

- @JoinColumn (https://docs.oracle.com/javaee/7/api/javax/persistence/JoinColumn.html) - Indicates a column for joining entities in `@ManyToOne` or `@OneToOne` relationships at the owning side or unidirectional `@OneToMany`
- @OneToOne (https://docs.oracle.com/javaee/7/api/javax/persistence/OneToOne.html) - Indicates 1:1 relationship
- @MapsId (https://docs.oracle.com/javaee/7/api/javax/persistence/MapsId.html) - References joining columns of owning side of `@ManyToOne` or `@OneToOne` relationships to be the primary key of referencing and referenced entities
- @ManyToMany (https://docs.oracle.com/javaee/7/api/javax/persistence/ManyToMany.html) - Indicates N:M relationship
- @JoinTable (https://docs.oracle.com/javaee/7/api/javax/persistence/JoinTable.html) - Specifies an association using a join table
- @BatchSize (https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/annotations/BatchSize.html) - Defines size to lazy load a collection of annotated entities
- @FetchMode (https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/annotations/FetchMode.html) - Defines fetching strategy for an association, e.g. loading all entities in a single subquery

Spring Security

- @EnableWebSecurity (https://docs.spring.io/spring

- @EnableWebSecurity (https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/EnableWebSecurity.html) - Enables web security
- @EnableGlobalMethodSecurity (https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/method/configuration/EnableGlobalMethodSecurity.html) - Enables method security
- @PreAuthorize (https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/access/prepost/PreAuthorize.html) - Defines access-control expression using SpEL, which is evaluated before invoking a protected method
- @PostAuthorize (https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/access/prepost/PostAuthorize.html) - Defines access-control expression using SpEL, which is evaluated after invoking a protected method
- @RolesAllowed (https://docs.oracle.com/javaee/7/api/javax/annotation/security/RolesAllowed.html) - Specifies a list of security roles allowed to invoke protected method
- @Secured (https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/access/annotation/Secured.html) - Java 5 annotation for defining method level security

Spring AOP

- @EnableAspectJAutoProxy (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/EnableAspectJAutoProxy.html) - Enables support for handling components marked with `@Aspect`
- @Aspect

(https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/Aspect.html) - Declares an annotated component as an aspect containing pointcuts and advices

- @Before (https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/Before.html) - Declares a pointcut executed before the call is propagated to the join point
- @AfterReturning (https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/AfterReturning.html) - Declares a pointcut executed if the join point successfully returns a result
- @AfterThrowing (https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/AfterThrowing.html) - Declares a pointcut executed if the join point throws an exception
- @After (https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/After.html) - Declares a pointcut executed if the join point successfully returns a result or throws an exception
- @Around (https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/Around.html) - Declares a pointcut executed before the call giving control over the execution of the join point to the advice
- @Pointcut (https://javadoc.io/static/org.aspectj/aspectjrt/1.9.5/org/aspectj/lang/annotation/Pointcut.html) - Externalized definition a pointcut expression