what are all annotations used in spring and spring boot give me all annotations with explanation

ChatGPT

**ChatGPT**

Sure, here's a comprehensive list of commonly used annotations in Spring and Spring Boot along with their explanations:

**Core Spring Annotations**

1. **@Component**
   * Indicates that a class is a Spring component. This annotation is a generic stereotype for any Spring-managed component.
2. **@Repository**
   * Indicates that a class is a repository, typically a DAO (Data Access Object). This annotation translates any data access exceptions into Spring's DataAccessException.
3. **@Service**
   * Indicates that a class is a service, typically used in the service layer of an application.
4. **@Controller**
   * Indicates that a class is a controller, primarily used in Spring MVC for handling web requests.
5. **@RestController**
   * A specialization of **@Controller** that indicates the annotated class is a controller where **@ResponseBody** is applied to all methods by default, meaning it returns the object and object data directly into the HTTP response body.
6. **@Configuration**
   * Indicates that a class declares one or more **@Bean** methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime.
7. **@Bean**
   * Indicates that a method produces a bean to be managed by the Spring container.
8. **@Autowired**
   * Marks a constructor, field, setter method, or configuration method to be autowired by Spring's dependency injection facilities.
9. **@Qualifier**
   * Along with **@Autowired**, this annotation is used to avoid confusion when wiring beans with multiple implementations.
10. **@Value**
    * Annotation at the field or method/constructor parameter level that indicates a default value expression for the affected argument.
11. **@Primary**
    * Indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency.
12. **@Lazy**
    * Indicates that a bean should be instantiated lazily.
13. **@Scope**
    * Specifies the scope of a bean (e.g., singleton, prototype, request, session, etc.).

**Spring Boot Annotations**

1. **@SpringBootApplication**
   * A convenience annotation that adds **@Configuration**, **@EnableAutoConfiguration**, and **@ComponentScan** annotations with their default attributes.
2. **@EnableAutoConfiguration**
   * Enables Spring Boot's auto-configuration mechanism.
3. **@ComponentScan**
   * Configures component scanning directives for use with **@Configuration** classes.
4. **@ConfigurationProperties**
   * Binds and validates external configuration properties to a POJO.
5. **@EnableConfigurationProperties**
   * Enable **@ConfigurationProperties** annotated beans.
6. **@ConditionalOnProperty**
   * Conditional that checks whether a specific property has a specific value.
7. **@ConditionalOnClass**
   * Conditional that checks whether the specified classes are on the classpath.
8. **@ConditionalOnMissingBean**
   * Conditional that only matches when the specified bean class is not contained in the BeanFactory.
9. **@ConditionalOnBean**
   * Conditional that only matches when the specified bean class is contained in the BeanFactory.

**Spring Web Annotations**

1. **@RequestMapping**
   * Maps HTTP requests to handler methods of MVC and REST controllers.
2. **@GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping**
   * Specializations of **@RequestMapping** for HTTP GET, POST, PUT, DELETE, and PATCH methods.
3. **@RequestParam**
   * Binds HTTP request parameters to method parameters in controller methods.
4. **@PathVariable**
   * Binds URI template variables to method parameters in controller methods.
5. **@RequestBody**
   * Binds the HTTP request body to a method parameter in a controller.
6. **@ResponseBody**
   * Indicates that the return value of a method should be used as the response body of the request.
7. **@RestControllerAdvice**
   * A specialization of **@ControllerAdvice** that is used to define **@ExceptionHandler**, **@InitBinder**, and **@ModelAttribute** methods that apply to all **@RestController** controllers.
8. **@ExceptionHandler**
   * Handles exceptions thrown by request handling methods.
9. **@ResponseStatus**
   * Sets the HTTP status code on the HTTP response.

**Spring Data Annotations**

1. **@Entity**
   * Indicates that the class is an entity. This annotation is used to mark Java objects that are persisted to the database.
2. **@Repository**
   * Marks the class as a Data Access Object (DAO).
3. **@Transactional**
   * Marks a method, class, or interface as transactional.
4. **@Query**
   * Specifies a JPQL or SQL query to execute.

**Spring Security Annotations**

1. **@EnableWebSecurity**
   * Enables Spring Security’s web security support and provides the Spring MVC integration.
2. **@Secured**
   * Defines a method-level security annotation.
3. **@PreAuthorize, @PostAuthorize**
   * Defines expression-based pre/post authorization for methods.
4. **@EnableGlobalMethodSecurity**
   * Enables Spring Security global method security.

**Spring Batch Annotations**

1. **@EnableBatchProcessing**
   * Enables Spring Batch features.
2. **@StepScope**
   * Configures a bean definition to be lazily initialized and scoped to the lifetime of a Step execution.

**Testing Annotations**

1. **@RunWith**
   * Specifies the class to run the tests.
2. **@SpringBootTest**
   * Configures the tests to run Spring Boot application.
3. **@WebMvcTest**
   * Configures the tests for Spring MVC controllers.
4. **@DataJpaTest**
   * Configures the tests for Spring Data JPA.
5. **@MockBean**
   * Adds mock beans to the Spring application context.

These annotations provide powerful features and configurations for developing Spring and Spring Boot applications. Each annotation serves a specific purpose, from managing beans and dependencies to configuring web endpoints and security.