Top 50 Spring Boot Annotations Explained

- **@SpringBootApplication**
- **Explanation**: Marks the main class of a Spring Boot application and combines @Configuration, @EnableAutoConfiguration, and @ComponentScan.

- **@RestController**
- **Explanation**: Indicates that a class is a RESTful controller, combining @Controller and @ResponseBody.

- **@RequestMapping**
- **Explanation**: Maps web requests to specific handler methods, can be applied at the class or method level.

- **@GetMapping**
- **Explanation**: Shortcut for @RequestMapping(method = RequestMethod.GET), maps HTTP GET requests to handler methods.

- **@PostMapping**
- **Explanation**: Shortcut for @RequestMapping(method = RequestMethod.POST), maps HTTP POST requests to handler methods.

- **@PutMapping**
- **Explanation**: Shortcut for @RequestMapping(method = RequestMethod.PUT), maps HTTP PUT requests to handler methods.

- **@DeleteMapping**
- **Explanation**: Shortcut for @RequestMapping(method = RequestMethod.DELETE), maps HTTP DELETE requests to handler methods.

- **@PatchMapping**
- **Explanation**: Shortcut for @RequestMapping(method = RequestMethod.PATCH), maps HTTP PATCH requests to handler methods.

- **@Autowired**
- **Explanation**: Automatically wires dependencies in Spring beans, can be applied to fields, constructors, or methods.

- **@Component**
- **Explanation**: Marks a Java class as a Spring bean.

- **@Service**

- **Explanation**: Indicates that a class contains business logic, a specialization of @Component.

- **@Repository**
- **Explanation**: Indicates that a class is a data access object (DAO), a specialization of @Component.

- **@Controller**
- **Explanation**: Marks a class as a web controller in a Spring MVC application.

- **@RequestBody**
- **Explanation**: Maps the HTTP request body to a method parameter in a controller.

- **@ResponseBody**
- **Explanation**: Maps the return value of a method to the HTTP response body.

- **@PathVariable**
- **Explanation**: Binds a method parameter to a URI template variable.

- **@RequestParam**
- **Explanation**: Binds a method parameter to a web request parameter.

- **@RequestHeader**
- **Explanation**: Binds a method parameter to a web request header.

- **@CookieValue**
- **Explanation**: Binds a method parameter to a cookie value.

- **@ModelAttribute**
- **Explanation**: Binds a method parameter or method return value to a named model attribute and exposes it to a web view.

- **@SessionAttributes**
- **Explanation**: Indicates the names of model attributes that should be stored in the session.

- **@ExceptionHandler**
- **Explanation**: Indicates the method to be invoked when an exception is thrown.

- **@ControllerAdvice**

- **Explanation**: Allows the handling of exceptions across the whole application in a single global handler.

- **@CrossOrigin**
- **Explanation**: Enables Cross-Origin Resource Sharing (CORS) on a method or class.

- **@Configuration**
- **Explanation**: Declares a class as a configuration class, typically used with @Bean methods.

- **@Bean**
- **Explanation**: Indicates that a method produces a bean to be managed by the Spring container.

- **@Primary**
- **Explanation**: Indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency.

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

- **@PropertySource**
- **Explanation**: Provides a convenient and declarative mechanism for adding a set of PropertySources to Spring's Environment.

- **@EnableAutoConfiguration**
- **Explanation**: Enables Spring Boot's auto-configuration mechanism.

- **@Conditional**
- **Explanation**: Conditionally includes or excludes beans based on a condition.

- **@Profile**
- **Explanation**: Specifies the profiles a bean is eligible for registration in, controlling which beans are loaded in which environments.

- **@Lazy**
- **Explanation**: Delays the initialization of a bean until it is first requested.

- **@Async**
- **Explanation**: Indicates that a method should be executed asynchronously.

- **@Scheduled**
- **Explanation**: Schedules a method to be run at fixed intervals.

- **@EnableScheduling**
- **Explanation**: Enables Spring’s scheduled task execution capability.

- **@Transactional**
- **Explanation**: Demarcates transactional boundaries on a method or class.

- **@EnableTransactionManagement**
- **Explanation**: Enables Spring’s annotation-driven transaction management.

- **@Entity**
- **Explanation**: Specifies that a class is an entity and is mapped to a database table.

- **@Table**
- **Explanation**: Specifies the primary table for the annotated entity.

- **@Id**
- **Explanation**: Specifies the primary key of an entity.

- **@GeneratedValue**
- **Explanation**: Specifies the generation strategy for the primary key values.

- **@Column**
- **Explanation**: Specifies the mapped column for a persistent property or field.

- **@OneToMany, @ManyToOne, @OneToOne, @ManyToMany**
- **Explanation**: Defines various types of relationships between entities.

- **@JoinColumn**
- **Explanation**: Specifies the foreign key column for a relationship.

- **@JsonIgnoreProperties**
- **Explanation**: Specifies properties to ignore during JSON serialization and deserialization.

- **@JsonProperty**
- **Explanation**: Specifies the property name to be used during JSON serialization and deserialization.

- **@SpringBootTest**
- **Explanation**: Used to bootstrap the entire container and start the full Spring context for integration tests.

- **@Test**
- **Explanation**: Marks a method as a test method in a JUnit test class.

- **@Builder**
- **Explanation**: Eliminates the need to manually write builder classes, constructors, and setter methods.