





# **Spring, Spring Boot Annotations Cheat sheet**

As known there are a number of Annotations provided by Java's Spring, Spring Boot Framework, and it would be quite difficult to remember all. Hence I had comeup with below Spring, Spring Boot Annotations Cheat Sheet, to help the readers.













# <u>Click here to explore my Spring, Spring Boot course with 12 hours Video, with downloadable source code examples</u>

#### **Spring Core related Annotations:**

@Bean — method annotated with @Bean creates and returns Bean. Spring Container calls such methods, automatically.

@Configuration — Class annotated with @Configuration has methods annotated with @Bean or has data members annotated with @Value

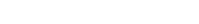
@Scope — indicates Scope of a Bean such as Singleton, Prototype, Session, etc...

@Lazy — indicates that Bean needs to be created on Demand only, i..e when there is explicit request

@Autowired — indicates Bean needs to be automatically created by Spring Container.

@Qualifier — used along with @Bean or @Autowired to avoid ambiguity







@Component — indicates a class as Component, so that it can be recognized by @ComponentScan, automatically. As known, all Component classes are automatically scanned and loaded by Spring Container.

@ComponentScan — scans one or more packages/subpackages for Components.

@Service — Components in Service Layer need to be annotated with @Service

@Repository — Components in Repository Layer need to be annotated with @Repository

@SpringBootApplication — This annotation is used with main class of Spring Boot Application

@Value — Data members of a Configuration class are automatically loaded from Configuration file(such as application.properties)

@ConfigurationProperties — Class annotated with @ConfigurationProperties automatically loads data members(with matching name)from Configuration file(such as application.properties)









@RestController — Class annotated with @RestController has REST end points.

@RequestBody — used with method parameter of REST end point. This annotation automatically describilizes the body(of Http request) into a Model or Entity object.

@PathVariable — used with method parameter of REST end point. It automatically retrieves a Path variable into the method parameter of REST end point.

@RequestParam — used with method parameter of REST end point. It automatically retrieves a Query parameter into the method parameter of REST end point.

REST End points are annotated with any of below annotation, to indicate specific HTTP method

- 1. @RequestMapping
- 2. @GetMapping
- 3. @PostMapping









### @ControllerAdvice — to Handle REST API Exceptions

@Valid — used with @RequestBody , to automatically validate the data members during descrialization. This annotation works along with Validation rules such as @NotNull, @Max, etc...

<u>Click here to explore my Spring, Spring Boot course with 12 hours Video, with downloadable source code examples</u>

#### **Spring Boot Data JPA related annotations:**

@Entity — class which need to be mapped with underlying DB Table

@Table — Used along with @Entity annotated, to specify custom name for DB Table(by default DB Table has same name as Entity Class name)

@Column — Used with Data members of Entity class, to indicate a Column of DB Table.

Data field Validation related — @NotNull, @Max, @Min, @Positive, @Negative, etc...







## **Security related Annotations:**

@CrossOrigin — Can be used with Class or method(s), indicating by which Origins(domain name or domain name patterns) the REST end points can be invoked.

Below annotations used for method level Security



- 1. @Secured
- 2. @PreAuthorize
- 3. @PermitAll

### **Caching related Annotations:**

@EnableCaching — Used along with @SpringBootApplication, which enables the application to perform Cache related operations

@Cacheable — Adds an entry to the Cache

@CachePut — Updates an existing entry in the Cache











@Transactional — Used by class/interface or method, indicates the method(s) is executed under a Transaction

<u>Click here to explore my Spring, Spring Boot course with 12 hours Video, with downloadable source code examples</u>

Thanks for reading this Post, and Happy Learning!!!

Read my other post(s):

<u>Create Custom Collections by extending an existing Collection class</u>

I will be sharing Spring Cloud related Annotations, shortly.





