Suresh Kumar
Nov 6, 2021 · 4 min read

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.

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

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 during Bean creation by Spring Container

@Primary — When there are multiple qualified Beans, priority is given to the Bean annotated with @Primary

@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.

Suresh Kumar
104 Followers
With about 23 Years of experience in Software development, currently into Corporate Training on Java Full Stack, Microservices, Angular, Design Patterns
Follow

@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)

REST API related Annotations:

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

@RequestBody — used with method parameter of REST end point. This annotation automatically deserializes 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.

@RequestHeader — used with method parameter of REST end point. It automatically retrieved value from a specified HTTP header and populates the value into the method parameter.

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

1. @RequestMapping
2. @GetMapping
3. @PostMapping
4. @PutMapping
5. @DeleteMapping , etc…

@ControllerAdvice, @ExceptionHandler — to Handle REST API Exceptions

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

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

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…

@Query — to specify Custom Query String(native or JPQL query), along with method declaration in Repository interface.

Entity class relationships — @OnetoOne, @OnetoMany, @ManytoOne, @ManytoMany

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

AOP related Annotations: Aspect Oriented Programming is used to separate Cross Cutting concerns(such as Logging, Security, etc…), from Business Logic. AOP is used only in selected Spring Boot Projects.

@Aspect — to specify that a class is Aspect, which holds Cross cutting concerns

@Pointcut — to specify Pointcut expressions

@Before — to specify a method is Before Advice

@After — to specify a method is After Advice

@Around — to specify a method is around Advice

As known, all advice methods are in an Aspect class.

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

@CacheEvict — Removes one or more entries from the Cache

Transaction related Annotations:

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

Click here to read my post on, Commonly used Spring Boot Starter dependencies, classes, interfaces

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

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

Read my other post(s):

Create Custom Collections by extending an existing Collection class

I will be sharing Spring Cloud related Annotations, shortly.

Spring Annotation · Spring Boot Annotation · Spring Cheat Sheet · Spring Boot Cheat Sheet · Spring

-- 2