第十八章：Spring 注解

小马哥 · mercyblitz

极客时间

扫码试看/订阅

《小马哥讲 Spring 核心编程思想》视频课程

Spring 注解

1. Spring 注解驱动编程发展历程
2. Spring 核心注解场景分类
3. Spring 注解编程模型
4. Spring 元注解（Meta-Annotations）
5. Spring 模式注解（Stereotype Annotations）
6. Spring 组合注解（Composed Annotations）
7. Spring 注解属性别名（Attribute Aliases）
8. Spring 注解属性覆盖（Attribute Overrides）
9. Spring @Enable 模块驱动
10. Spring 条件注解

Spring 注解

11. 课外资料
12. 面试题精选

Spring 注解驱动编程发展历程

- 注解驱动启蒙时代：Spring Framework 1.x
- 注解驱动过渡时代：Spring Framework 2.x
- 注解驱动黄金时代：Spring Framework 3.x
- 注解驱动完善时代：Spring Framework 4.x
- 注解驱动当下时代：Spring Framework 5.x

Spring 核心注解场景分类

- Spring 模式注解

Spring 注解	场景说明	起始版本
@Repository	数据仓储模式注解	2.0
@Component	通用组件模式注解	2.5
@Service	服务模式注解	2.5
@Controller	Web 控制器模式注解	2.5
@Configuration	配置类模式注解	3.0

Spring 核心注解场景分类

- 装配注解

Spring 注解	场景说明	起始版本
@ImportResource	替换 XML 元素 <import>	2.5
@Import	导入 Configuration 类	2.5
@ComponentScan	扫描指定 package 下标注 Spring 模式注解的类	3.1

- 依赖注入注解

Spring 注解	场景说明	起始版本
@Autowired	Bean 依赖注入，支持多种依赖查找方式	2.5
@Qualifier	细粒度的 @Autowired 依赖查找	2.5

Spring 注解编程模型

- 编程模型
  - 元注解（Meta-Annotations）
  - Spring 模式注解（Stereotype Annotations）
  - Spring 组合注解（Composed Annotations）
  - Spring 注解属性别名和覆盖（Attribute Aliases and Overrides）

https://github.com/spring-projects/spring-framework/wiki/Spring-Annotation-Programming-Model

Spring 元注解（Meta-Annotations）

- 官方 Wiki 原文

A meta-annotation is an annotation that is declared on another annotation. An annotation is therefore meta-annotated if it is annotated with another annotation. For example, any annotation that is declared to be documented is meta-annotated with @Documented from the java.lang.annotation package.

- 举例说明
  - java.lang.annotation.Documented
  - java.lang.annotation.Inherited
  - java.lang.annotation.Repeatable

Spring 模式注解（Stereotype Annotations）

- 官方 Wiki 原文

A **stereotype annotation** is an annotation that is used to declare the role that a component plays within the application. For example, the @Repository annotation in the Spring Framework is a marker for any class that fulfills the role or stereotype of a repository (also known as Data Access Object or DAO).

@Component is a generic stereotype for any Spring-managed component. Any component annotated with @Component is a candidate for component scanning. Similarly, any component annotated with an annotation that is itself meta-annotated with @Component is also a candidate for component scanning. For example, @Service is meta-annotated with @Component.

Core Spring provides several stereotype annotations out of the box, including but not limited to: @Component, @Service, @Repository, @Controller, @RestController, and @Configuration. @Repository, @Service, etc. are specializations of @Component.

Spring 模式注解（Stereotype Annotations）

- 理解 @Component “派生性”

  元标注 @Component 的注解在 XML 元素 <context:component-scan> 或注解 @ComponentScan 扫描中“派生”了 @Component 的特性，并且从 Spring Framework 4.0 开始支持多层次“派生性”。

- 举例说明
  - @Repository
  - @Service
  - @Controller
  - @Configuration
  - @SpringBootConfiguration（Spring Boot）

Spring 模式注解（Stereotype Annotations）

- @Component “派生性”原理
  - 核心组件 - org.springframework.context.annotation.ClassPathBeanDefinitionScanner
    - org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider
  - 资源处理 - org.springframework.core.io.support.ResourcePatternResolver
  - 资源-类元信息
    - org.springframework.core.type.classreading.MetadataReaderFactory
  - 类元信息 - org.springframework.core.type.ClassMetadata
    - ASM 实现 - org.springframework.core.type.classreading.ClassMetadataReadingVisitor
    - 反射实现 - org.springframework.core.type.StandardAnnotationMetadata
  - 注解元信息 - org.springframework.core.type.AnnotationMetadata
    - ASM 实现 - org.springframework.core.type.classreading.AnnotationMetadataReadingVisitor
    - 反射实现 - org.springframework.core.type.StandardAnnotationMetadata

Spring 组合注解（Composed Annotations）

- 官方 Wiki 原文

A composed annotation is an annotation that is meta-annotated with one or more annotations with the intent of combining the behavior associated with those meta-annotations into a single custom annotation. For example, an annotation named @TransactionalService that is meta-annotated with Spring's @Transactional and @Service annotations is a composed annotation that combines the semantics of @Transactional and @Service. @TransactionalService is technically also a custom stereotype annotation.

- 基本定义

Spring 组合注解（Composed Annotations）中的元注允许是 Spring 模式注解（Stereotype Annotation）与其他 Spring 功能性注解的任意组合。

Spring 注解属性别名（Attribute Aliases）

- 官方 Wiki 原文

An attribute alias is an alias from one annotation attribute to another annotation attribute. Attributes within a set of aliases can be used interchangeably and are treated as equivalent. Attribute aliases can be categorized as follows.

- Explicit Aliases: if two attributes in one annotation are declared as aliases for each other via @AliasFor, they are explicit aliases.
- Implicit Aliases: if two or more attributes in one annotation are declared as explicit overrides for the same attribute in a meta-annotation via @AliasFor, they are implicit aliases.
- Transitive Implicit Aliases: given two or more attributes in one annotation that are declared as explicit overrides for attributes in meta-annotations via @AliasFor, if the attributes effectively override the same attribute in a meta-annotation following the law of transitivity, they are transitive implicit aliases.

Spring 注解属性覆盖（Attribute Overrides）

类似于继承中子类覆盖父类

- 官方 Wiki 原文

An attribute override is an annotation attribute that overrides (or shadows) an annotation attribute in a meta-annotation. Attribute overrides can be categorized as follows.

- Implicit Overrides: given attribute A in annotation @One and attribute A in annotation @Two, if @One is meta-annotated with @Two, then attribute A in annotation @One is an implicit override for attribute A in annotation @Two based solely on a naming convention (i.e., both attributes are named A).
- Explicit Overrides: if attribute A is declared as an alias for attribute B in a meta-annotation via @AliasFor, then A is an explicit override for B.
- Transitive Explicit Overrides: if attribute A in annotation @One is an explicit override for attribute B in annotation @Two and B is an explicit override for attribute C in annotation @Three, then A is a transitive explicit override for C following the law of transitivity.

Spring @Enable 模块驱动

- @Enable 模块驱动

@Enable 模块驱动是以 @Enable 为前缀的注解驱动编程模型。所谓“模块”是指具备相同领域的功能组件集合，组合所形成一个独立的单元。比如 Web MVC 模块、AspectJ代理模块、Caching（缓存）模块、JMX（Java 管理扩展）模块、Async（异步处理）模块等。

- 举例说明
  - @EnableWebMvc
  - @EnableTransactionManagement
  - @EnableCaching
  - @EnableMBeanExport
  - @EnableAsync

Spring @Enable 模块驱动

- @Enable 模块驱动编程模式
  - 驱动注解：@EnableXXX
  - 导入注解：@Import 具体实现
  - 具体实现
    - 基于 Configuration Class
    - 基于 ImportSelector 接口实现
    - 基于 ImportBeanDefinitionRegistrar 接口实现

Spring 条件注解

- 基于配置条件注解 - @org.springframework.context.annotation.Profile
  - 关联对象 - org.springframework.core.env.Environment 中的 Profiles
  - 实现变化：从 Spring 4.0 开始，@Profile 基于 @Conditional 实现

- 基于编程条件注解 - @org.springframework.context.annotation.Conditional
  - 关联对象 - org.springframework.context.annotation.Condition 具体实现

Spring 条件注解

- @Conditional 实现原理
  - 上下文对象 - org.springframework.context.annotation.ConditionContext
  - 条件判断 - org.springframework.context.annotation.ConditionEvaluator
  - 配置阶段 - org.springframework.context.annotation.ConfigurationCondition.ConfigurationPhase
  - 判断入口 - org.springframework.context.annotation.ConfigurationClassPostProcessor
    - org.springframework.context.annotation.ConfigurationClassParser

课外资料

- Spring Boot 注解

注解	场景说明	起始版本
@SpringBootConfiguration	Spring Boot 配置类	1.4.0
@SpringBootApplication	Spring Boot 应用引导注解	1.2.0
@EnableAutoConfiguration	Spring Boot 激活自动转配	1.0.0

课外资料

- Spring Cloud 注解

注解	场景说明	起始版本
@SpringCloudApplication	Spring Cloud 应用引导注解	1.0.0
@EnableDiscoveryClient	Spring Cloud 激活服务发现客户端注解	1.0.0
@EnableCircuitBreaker	Spring Cloud 激活熔断注解	1.0.0

面试题

沙雕面试题 - Spring 模式注解有哪些?

答:

- @org.springframework.stereotype.Component
- @org.springframework.stereotype.Repository
- @org.springframework.stereotype.Service
- @org.springframework.stereotype.Controller
- @org.springframework.context.annotation.Configuration

面试题

996 面试题 - @EventListener 的工作原理？

答：

- 源码导读 - org.springframework.context.event.EventListenerMethodProcessor

面试题

劝退面试题 - @PropertySource 的工作原理？

答：答案下章揭晓

极客时间

扫码试看/订阅

《小马哥讲 Spring 核心编程思想》视频课程