

## 第三章：Spring AOP API 设计与实现

小马哥 (mercyblitz)



扫码试看/订阅

《小马哥讲 Spring AOP 编程思想》视频课程

# Spring AOP API 设计与实现

---

1. Spring AOP API 整体设计
2. 接入点接口 - Joinpoint
3. Joinpoint 条件接口 - Pointcut
4. Pointcut 操作 - ComposablePointcut
5. Pointcut 便利实现
6. Pointcut AspectJ 实现 - AspectJExpressionPointcut
7. Joinpoint 执行动作 - Advice
8. Joinpoint Before Advice 标准实现
9. Joinpoint Before Advice AspectJ 实现
10. Joinpoint After Advice 标准实现

# Spring AOP API 设计与实现

---

- 11. Joinpoint After Advice AspectJ 实现
- 12. Advice 容器接口 - Advisor
- 13. Pointcut 与 Advice 连接器 - PointcutAdvisor
- 14. Introduction 与 Advice 连接器 - IntroductionAdvisor
- 15. Advisor 的 Interceptor 适配器 - AdvisorAdapter
- 16. AdvisorAdapter 实现
- 17. AOP 代理接口 - AopProxy
- 18. AopProxy 工厂接口与实现
- 19. JDK AopProxy 实现 - JdkDynamicAopProxy
- 20. CGLIB AopProxy 实现 - CglibAopProxy

# Spring AOP API 设计与实现

---

- 21. AopProxyFactory 配置管理器 - AdvisedSupport
- 22. Advisor 链工厂接口与实现 - AdvisorChainFactory
- 23. 目标对象来源接口与实现 - TargetSource
- 24. 代理对象创建基础类 - ProxyCreatorSupport
- 25. AdvisedSupport 事件监听器 - AdvisedSupportListener
- 26. ProxyCreatorSupport 标准实现 - ProxyFactory
- 27. ProxyCreatorSupport IoC 容器实现 - ProxyFactoryBean
- 28. ProxyCreatorSupport AspectJ 实现 - AspectJProxyFactory
- 29. IoC 容器自动代理抽象 - AbstractAutoProxyCreator
- 30. IoC 容器自动代理标准实现

# Spring AOP API 设计与实现

---

- 31. IoC 容器自动代理 AspectJ 实现 - AspectJAwareAdvisorAutoProxyCreator
- 32. AOP Infrastructure Bean 接口 - AopInfrastructureBean
- 33. AOP 上下文辅助类 - AopContext
- 34. 代理工厂工具类 - AopProxyUtils
- 35. AOP 工具类 - AopUtils
- 36. AspectJ Enable 模块驱动实现 - @EnableAspectJAutoProxy
- 37. AspectJ XML 配置驱动实现 - <aop:aspectj-autoproxy/>
- 38. AOP 配置 Schema-based 实现 - <aop:config/>
- 39. Aspect Schema-based 实现 - <aop:aspect/>
- 40. Pointcut Schema-based 实现 - <aop:pointcut/>

# Spring AOP API 设计与实现

---

- 41. Around Advice Schema-based 实现 - `<aop:around/>`
- 42. Before Advice Schema-based 实现 - `<aop:before/>`
- 43. After Advice Schema-based 实现
- 44. Introduction Schema-based 实现 - `<aop:declare-parents/>`
- 45. 作用域代理 Schema-based 实现 - `<aop:scoped-proxy/>`
- 46. 面试题精选

# Spring AOP API 整体设计

- Join point - Joinpoint
- Pointcut - Pointcut
- Advice 执行动作 - Advice
- Advice 容器 - Advisor
- Introduction - IntroductionInfo
- 代理对象创建基础类 - ProxyCreatorSupport
- 代理工厂 - ProxyFactory、ProxyFactoryBean
- AopProxyFactory 配置管理器 - AdvisedSupport
- IoC 容器自动代理抽象 - AbstractAutoProxyCreator



# 接入点接口 - Joinpoint

- Interceptor 执行上下文 - Invocation
  - 方法拦截器执行上下文 - MethodInvocation
  - ~~• 构造器拦截器执行上下文 - ConstructorInvocation~~
- MethodInvocation 实现
  - 基于反射 - ReflectiveMethodInvocation
  - 基于 CGLIB - CglibMethodInvocation

# Joinpoint 条件接口 - Pointcut

- 核心组件
  - 类过滤器 - ClassFilter
  - 方法匹配器 - MethodMatcher

# Pointcut 操作

- 组合实现 - `org.springframework.aop.support.ComposablePointcut`
- 工具类
  - ClassFilter 工具类 - `ClassFilters`
  - MethodMatcher 工具类 - `MethodMatchers`
  - Pointcut 工具类 - `Pointcuts`

# Pointcut 便利实现

- 静态 Pointcut - StaticMethodMatcherPointcut
- 正则表达式 Pointcut - JdkRegexpMethodPointcut
- 控制流 Pointcut - ControlFlowPointcut

# Pointcut AspectJ 实现

- 实现类 - `org.springframework.aop.aspectj.AspectJExpressionPointcut`
- 指令支持 - `SUPPORTED_PRIMITIVES` 字段
- 表达式 - `org.aspectj.weaver.tools.PointcutExpression`

# Joinpoint 执行动作接口 - Advice

- Around Advice - Interceptor
  - 方法拦截器 - MethodInterceptor
  - 构造器拦截器 - ConstructorInterceptor
- 前置动作
  - 标准接口 - org.springframework.aop.BeforeAdvice
  - 方法级别 - org.springframework.aop.MethodBeforeAdvice
- 后置动作
  - org.springframework.aop.AfterAdvice
  - org.springframework.aop.AfterReturningAdvice
  - org.springframework.aop.ThrowsAdvice

# Joinpoint Before Advice 标准实现

- 接口
  - 标准接口 - `org.springframework.aop.BeforeAdvice`
  - 方法级别 - `org.springframework.aop.MethodBeforeAdvice`
- 实现
  - `org.springframework.aop.framework.adapter.MethodBeforeAdviceInterceptor`

# Joinpoint Before Advice AspectJ 实现

- 实现类 - `org.springframework.aop.aspectj.AspectJMethodBeforeAdvice`



# Joinpoint After Advice 标准实现

- 接口
  - `org.springframework.aop.AfterAdvice`
  - `org.springframework.aop.AfterReturningAdvice`
  - `org.springframework.aop.ThrowsAdvice`
- 实现
  - `org.springframework.aop.framework.adapter.ThrowsAdviceInterceptor`
  - `org.springframework.aop.framework.adapter.AfterReturningAdviceInterceptor`

# Joinpoint After Advice AspectJ 实现

- 接口
  - `org.springframework.aop.AfterAdvice`
  - `org.springframework.aop.AfterReturningAdvice`
  - `org.springframework.aop.ThrowsAdvice`
- 实现
  - `org.springframework.aop.aspectj.AspectJAfterAdvice`
  - `org.springframework.aop.aspectj.AspectJAfterReturningAdvice`
  - `org.springframework.aop.aspectj.AspectJAfterThrowingAdvice`

# Advice 容器接口 - Advisor

- 接口 - `org.springframework.aop.Advisor`
  - 通用实现 - `org.springframework.aop.support.DefaultPointcutAdvisor`

# Pointcut 与 Advice 连接器 - PointcutAdvisor

- 接口 - `org.springframework.aop.PointcutAdvisor`
  - 通用实现
    - `org.springframework.aop.support.DefaultPointcutAdvisor`
  - AspectJ 实现
    - `org.springframework.aop.aspectj.AspectJExpressionPointcutAdvisor`
    - `org.springframework.aop.aspectj.AspectJPointcutAdvisor`
  - 静态方法实现
    - `org.springframework.aop.support.StaticMethodMatcherPointcutAdvisor`
  - IoC 容器实现
    - `org.springframework.aop.support.AbstractBeanFactoryPointcutAdvisor`

# Introduction 与 Advice 连接器

- 接口 - `org.springframework.aop.IntroductionAdvisor`
  - 元信息
    - `org.springframework.aop.IntroductionInfo`
  - 通用实现
    - `org.springframework.aop.support.DefaultIntroductionAdvisor`
- AspectJ 实现
  - `org.springframework.aop.aspectj.DeclareParentsAdvisor`

# Advisor 的 Interceptor 适配器

- 接口 - `org.springframework.aop.framework.adapter.AdvisorAdapter`
  - MethodBeforeAdvice 实现
    - `org.springframework.aop.framework.adapter.MethodBeforeAdviceAdapter`
  - AfterReturningAdvice 实现
    - `org.springframework.aop.framework.adapter.AfterReturningAdviceAdapter`
  - ThrowsAdvice 实现
    - `org.springframework.aop.framework.adapter.ThrowsAdviceAdapter`

# AdvisorAdapter 实现

- MethodBeforeAdvice 实现
  - `org.springframework.aop.framework.adapter.MethodBeforeAdviceAdapter`
- AfterReturningAdvice 实现
  - `org.springframework.aop.framework.adapter.AfterReturningAdviceAdapter`
- ThrowsAdvice 实现
  - `org.springframework.aop.framework.adapter.ThrowsAdviceAdapter`

# AOP 代理接口 - AopProxy

- 接口 - `org.springframework.aop.framework.AopProxy`
- 实现
  - JDK 动态代理
    - `org.springframework.aop.framework.JdkDynamicAopProxy`
  - CGLIB 字节码提升
    - `org.springframework.aop.framework.CglibAopProxy`
      - `org.springframework.aop.framework.ObjenesisCglibAopProxy`



# AopProxy 工厂接口与实现

- 接口 - `org.springframework.aop.framework.AopProxyFactory`
- 默认实现: `org.springframework.aop.framework.DefaultAopProxyFactory`
  - 返回类型
    - `org.springframework.aop.framework.JdkDynamicAopProxy`
    - `org.springframework.aop.framework.CglibAopProxy`
      - `org.springframework.aop.framework.ObjenesisCglibAopProxy`

# JDK AopProxy 实现 - JdkDynamicAopProxy

- 实现 - `org.springframework.aop.framework.JdkDynamicAopProxy`
  - 配置 - `org.springframework.aop.framework.AdvisedSupport`
  - 来源 - `org.springframework.aop.framework.DefaultAopProxyFactory`

# CGLIB AopProxy 实现 - CglibAopProxy

- 实现 - `org.springframework.aop.framework.CglibAopProxy`
  - 配置 - `org.springframework.aop.framework.AdvisedSupport`
  - 来源 - `org.springframework.aop.framework.DefaultAopProxyFactory`

# AopProxyFactory 配置管理器

- 核心 API - `org.springframework.aop.framework.AdvisedSupport`
  - 语义 - 代理配置
  - 基类 - `org.springframework.aop.framework.ProxyConfig`
  - 实现接口 - `org.springframework.aop.framework.Advised`
  - 使用场景 - `org.springframework.aop.framework.AopProxy` 实现

# Advisor 链工厂接口与实现

- 核心 API - `org.springframework.aop.framework.AdvisorChainFactory`
  - 特殊实现 - `org.springframework.aop.framework.InterceptorAndDynamicMethodMatcher`
  - 默认实现 - `org.springframework.aop.framework.DefaultAdvisorChainFactory`

# 目标对象来源接口与实现

- 核心 API - `org.springframework.aop.TargetSource`
  - 实现
    - `org.springframework.aop.target.HotSwappableTargetSource`
    - `org.springframework.aop.target.AbstractPoolingTargetSource`
    - `org.springframework.aop.target.PrototypeTargetSource`
    - `org.springframework.aop.target.ThreadLocalTargetSource`
    - `org.springframework.aop.target.SingletonTargetSource`

# 代理对象创建基础类

- 核心 API - `org.springframework.aop.framework.ProxyCreatorSupport`
  - 语义 - 代理对象创建基类
- 基类 - `org.springframework.aop.framework.AdvisedSupport`

# AdvisedSupport 事件监听器

- 核心 API - `org.springframework.aop.framework.AdvisedSupportListener`
  - 事件对象 - `org.springframework.aop.framework.AdvisedSupport`
  - 事件来源 - `org.springframework.aop.framework.ProxyCreatorSupport`
  - 激活事件触发 - `ProxyCreatorSupport#createAopProxy`
  - 变更事件触发 - 代理接口变化时、Advisor 变化时、配置复制



# ProxyCreatorSupport 标准实现

- 核心 API - `org.springframework.aop.framework.ProxyFactory`
  - 基类 - `org.springframework.aop.framework.ProxyCreatorSupport`
- 特性增强
  - 提供一些便利操作

# ProxyCreatorSupport IoC 容器实现

- 核心 API - `org.springframework.aop.framework.ProxyFactoryBean`
  - 基类 - `org.springframework.aop.framework.ProxyCreatorSupport`
- 特点
  - Spring IoC 容器整合
    - `org.springframework.beans.factory.BeanClassLoaderAware`
    - `org.springframework.beans.factory.BeanFactoryAware`
- 特性增强
  - 实现 `org.springframework.beans.factory.FactoryBean`

# ProxyCreatorSupport AspectJ 实现

- 核心 API - `org.springframework.aop.aspectj.annotation.AspectJProxyFactory`
  - 基类 - `org.springframework.aop.framework.ProxyCreatorSupport`
- 特点
  - AspectJ 注解整合
- 相关 API
  - AspectJ 元数据 - `org.springframework.aop.aspectj.annotation.AspectMetadata`
  - AspectJ Advisor 工厂 - `org.springframework.aop.aspectj.annotation.AspectJAdvisorFactory`

# IoC 容器自动代理抽象

- API -  
`org.springframework.aop.framework.autoproxy.AbstractAutoProxyCreator`
  - 基类 - `org.springframework.aop.framework.ProxyProcessorSupport`
- 特点
  - 与 Spring Bean 生命周期整合
    - `org.springframework.beans.factory.config.SmartInstantiationAwareBeanPostProcessor`

# IoC 容器自动代理标准实现

- 基类 -  
`org.springframework.aop.framework.autoproxy.AbstractAutoProxyCreator`
  - 默认实现 - `DefaultAdvisorAutoProxyCreator`
  - Bean 名称匹配实现 - `BeanNameAutoProxyCreator`
  - Infrastructure Bean 实现 - `InfrastructureAdvisorAutoProxyCreator`

# IoC 容器自动代理 AspectJ 实现

- `org.springframework.aop.aspectj.autoproxy.AspectJAwareAdvisorAutoProxyCreator`
  - 基类 - `org.springframework.aop.framework.autoproxy.AbstractAdvisorAutoProxyCreator`

# AOP Infrastructure Bean 接口

- 接口 - `org.springframework.aop.framework.AopInfrastructureBean`
  - 语义 - Spring AOP 基础 Bean 标记接口
- 实现
  - `org.springframework.aop.framework.autoproxy.AbstractAutoProxyCreator`
  - `org.springframework.aop.scope.ScopedProxyFactoryBean`
- 判断逻辑
  - `AbstractAutoProxyCreator#isInfrastructureClass`
  - `ConfigurationClassUtils#checkConfigurationClassCandidate`

# AOP 上下文辅助类

- API - `org.springframework.aop.framework.AopContext`
  - 语义 - `ThreadLocal` 的扩展，临时存储 AOP 对象



# AOP 代理工具类

- API - `org.springframework.aop.framework.AopProxyUtils`
  - 代表方法
    - `getSingletonTarget` - 从实例中获取单例对象
    - `ultimateTargetClass` - 从实例中获取最终目标类
    - `completeProxiedInterfaces` - 计算 `AdvisedSupport` 配置中所有被代理的接口
    - `proxiedUserInterfaces` - 从代理对象中获取代理接口

# AOP 工具类

- API - `org.springframework.aop.support.AopUtils`
  - 代表方法
    - `isAopProxy` - 判断对象是否为代理对象
    - `isJdkDynamicProxy` - 判断对象是否为 JDK 动态代理对象
    - `isCglibProxy` - 判断对象是否为 CGLIB 代理对象
    - `getTargetClass` - 从对象中获取目标类型
    - `invokeJoinpointUsingReflection` - 使用 Java 反射调用 Joinpoint（目标方法）

# AspectJ Enable 模块驱动实现

- 注解 - `org.springframework.context.annotation.EnableAspectJAutoProxy`
  - 属性方法
    - `proxyTargetClass` - 是否已类型代理
    - `exposeProxy` - 是否将代理对象暴露在 `AopContext` 中
  - 设计模式 - `@Enable` 模块驱动
    - `ImportBeanDefinitionRegistrar` 实现 - `org.springframework.context.annotation.AspectJAutoProxyRegistrar`
  - 底层实现
    - `org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator`

# AspectJ XML 配置驱动实现

- XML 元素 - `<aop:aspectj-autoproxy/>`
  - 属性
    - `proxy-target-class` - 是否已类型代理
    - `expose-proxy` - 是否将代理对象暴露在 `AopContext` 中
  - 设计模式 - Extensible XML Authoring
  - 底层实现
    - `org.springframework.aop.config.AspectJAutoProxyBeanDefinitionParser`

# AOP 配置 Schema-based 实现

- XML 元素 - <aop:config/>
  - 属性
    - proxy-target-class - 是否已类型代理
    - expose-proxy - 是否将代理对象暴露在 AopContext 中
  - 嵌套元素
    - pointcut
    - advisor
    - aspect
- 底层实现
  - org.springframework.aop.config.ConfigBeanDefinitionParser

# Aspect Schema-based 实现

- XML 元素 - <aop:aspect/>
  - 父元素 - <aop:config/>
  - 属性
    - ref - Spring Bean 引用的名称
    - order - Aspect 顺序数
  - 嵌套元素
    - pointcut
    - declare-parents
    - before
    - after
    - after-returning
    - after-throwing
    - around

# Pointcut Schema-based 实现

- XML 元素 - `<aop:pointcut/>`
  - 父元素 - `<aop:aspect/>` 或 `<aop:config/>`
  - 属性
    - `id` - Pointcut ID
    - `expression` - （必须）AspectJ 表达式
- 底层实现
  - `org.springframework.aop.Pointcut`

# Around Advice Schema-based 实现

- XML 元素 - `<aop:around/>`
  - 父元素 - `<aop:aspect/>`
  - 属性
    - `pointcut` - AspectJ Pointcut 表达式
    - `pointcut-ref` - 引用的 AspectJ Pointcut 名称
    - `method` - 拦截目标方法
    - `arg-names` - 目标方法参数名称



# Before Advice Schema-based 实现

- XML 元素 - `<aop:before/>`
  - 父元素 - `<aop:aspect/>`
  - 属性
    - `pointcut` - AspectJ Pointcut 表达式
    - `pointcut-ref` - 引用的 AspectJ Pointcut 名称
    - `method` - 拦截目标方法
    - `arg-names` - 目标方法参数名称

# After Advice Schema-based 实现

- XML 元素 - `<aop:after/>`
  - 父元素 - `<aop:aspect/>`
  - 属性
    - `pointcut` - AspectJ Pointcut 表达式
    - `pointcut-ref` - 引用的 AspectJ Pointcut 名称
    - `method` - 拦截目标方法
    - `arg-names` - 目标方法参数名称

# After Returning Advice Schema-based 实现

- XML 元素 - `<aop:after-returning/>`
  - 父元素 - `<aop:aspect/>`
  - 属性
    - `pointcut` - AspectJ Pointcut 表达式
    - `pointcut-ref` - 引用的 AspectJ Pointcut 名称
    - `method` - 拦截目标方法
    - `arg-names` - 目标方法参数名称
    - `returning` - 方法参数名称

# After Throwing Advice Schema-based 实现

- XML 元素 - `<aop:after-throwing/>`
  - 父元素 - `<aop:aspect/>`
  - 属性
    - `pointcut` - AspectJ Pointcut 表达式
    - `pointcut-ref` - 引用的 AspectJ Pointcut 名称
    - `method` - 拦截目标方法
    - `arg-names` - 目标方法参数名称
    - `throwing` - 方法参数名称

# Advisor Schema-based 实现

- XML 元素 - <aop:advisor/>
  - 父元素 - <aop:config/>
  - 属性
    - advice-ref - Advice Bean 引用
    - pointcut - AspectJ Pointcut 表达式
    - pointcut-ref - AspectJ Pointcut Bean 引用
    - order - Advisor 顺序数

# Introduction Schema-based 实现

- XML 元素 - <aop:declare-parents/>
  - 父元素 - <aop:aspect/>
  - 属性
    - types-matching - 是否已类型代理
    - implement-interface - 实现接口全类名
    - default-impl - 默认实现全类名
    - delegate-ref - 委派实现 Bean 引用

# 作用域代理 Schema-based 实现

- XML 元素 - `<aop:scoped-proxy/>`
  - 属性
    - `proxy-target-class` - 是否已类型代理

# 面试题精选

沙雕面试题 - Spring AOP Advice XML 标签有哪些？



我真的没笑

答：

- Around Advice : `<aop:around/>`
- Before Advice : `<aop:before/>`
- After: `<aop:after/>`
- AfterReturning : `<aop:after-returning/>`
- AfterThrowing : `<aop:after-throwing/>`



## 面试题精选

**996 面试题** - 请解释 Spring @EnableAspectJAutoProxy 的原理?



答: 参考 @EnableAspectJAutoProxy

## 面试题精选

**劝退面试题** - Spring Configuration Class CGLIB 提升  
与 AOP 类代理的关系?



答:



扫码试看/订阅

《小马哥讲 Spring AOP 编程思想》视频课程