

# 第五章：Spring AOP 在 Spring Framework 内部应用

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 在 Spring Framework 内部应用

---

1. Spring AOP 在 Spring 事件 (Events)
2. Spring AOP 在 Spring 事务 (Transactions) 理论基础
3. Spring AOP 在 Spring 事务 (Transactions) 源码分析
4. Spring AOP 在 Spring 缓存 (Caching)
5. Spring AOP 在 Spring 本地调度 (Scheduling)
6. 面试题精选
7. 结束语

# Spring AOP 在 Spring 事件 (Events)

- 核心 API - `org.springframework.context.event.EventPublicationInterceptor`

- 特性描述

当 Spring AOP 代理 Bean 中的 JoinPoint 方法执行后, Spring ApplicationContext 将发布一个自定义事件 (ApplicationEvent 子类)

- 使用限制

- EventPublicationInterceptor 关联的 ApplicationEvent 子类必须存在单参数的构造器
- EventPublicationInterceptor 需要被声明为 Spring Bean

# Spring AOP 在 Spring 事务 (Transactions)

- 核心 API
  - Spring 事务 @Enable 模块驱动 - @EnableTransactionManagement
  - Spring 事务注解 - @Transactional
  - Spring 事务事件监听器 - @TransactionalEventListener
  - Spring 事务定义 - TransactionDefinition
  - Spring 事务状态 - TransactionStatus
  - Spring 平台事务管理器 - PlatformTransactionManager
  - Spring 事务代理配置 - ProxyTransactionManagementConfiguration
  - Spring 事务 PointcutAdvisor 实现 - BeanFactoryTransactionAttributeSourceAdvisor
  - Spring 事务 MethodInterceptor 实现 - TransactionInterceptor
  - Spring 事务属性源 - TransactionAttributeSource

# Spring AOP 在 Spring 事务 (Transactions)

- 理解 TransactionDefinition (Spring 事务定义)
  - 说明: Interface that defines Spring-compliant transaction properties. Based on the propagation behavior definitions analogous to EJB CMT attributes.
- 核心方法
  - `getIsolationLevel()`: 获取隔离级别, 默认值 `ISOLATION_DEFAULT` 常量, 参考 `org.springframework.transaction.annotation.Isolation`
  - `getPropagationBehavior()`: 获取事务传播, 默认值: `PROPAGATION_REQUIRED` 常量, 参考 `org.springframework.transaction.annotation.Propagation`
  - `getTimeout()`: 获取事务执行超时时间, 默认值: `TIMEOUT_DEFAULT` 常量
  - `isReadOnly()`: 是否为只读事务, 默认值: `false`

# Spring AOP 在 Spring 事务 (Transactions)

- 理解 TransactionStatus (Spring 事务状态)
  - 说明: Interface that specifies an API to programmatically manage transaction savepoints in a generic fashion. Extended by TransactionStatus to expose savepoint management functionality for a specific transaction.
- 核心方法
  - isNewTransaction(): 当前事务执行是否在新的事务
  - setRollbackOnly(): 将当前事务设置为只读
  - isRollbackOnly(): 当前事务是否为只读
  - isCompleted(): 当前事务是否完成

# Spring AOP 在 Spring 事务 (Transactions)

- 理解 PlatformTransactionManager (平台事务管理器)
  - 说明: the central interface in Spring's transaction infrastructure. Applications can use this directly, but it is not primarily meant as API: Typically, applications will work with either TransactionTemplate or declarative transaction demarcation through AOP.
- 核心方法
  - getTransaction(TransactionDefinition): 获取事务状态
  - commit(TransactionStatus): 提交事务
  - rollback(TransactionStatus): 回滚事务



# Spring AOP 在 Spring 事务 (Transactions)

- 理解 Spring 事务传播 (Transaction Propagation)
  - 官方文档: <https://docs.spring.io/spring-framework/docs/current/reference/html/data-access.html#tx-propagation>

# Spring AOP 在 Spring 缓存 (Caching)

- 核心 API
  - Spring 缓存 @Enable 模块驱动 - @EnableCaching
  - 缓存操作注解 - @Caching、@Cachable、@CachePut、@CacheEvict
  - 缓存配置注解 - @CacheConfig
  - 缓存注解操作数据源 - AnnotationCacheOperationSource
  - Spring 缓存注解解析器 - SpringCacheAnnotationParser
  - Spring 缓存管理器 - CacheManager
  - Spring 缓存接口 - Cache
  - Spring 缓存代理配置 - ProxyCachingConfiguration
  - Spring 缓存 PointcutAdvisor 实现 - BeanFactoryCacheOperationSourceAdvisor
  - Spring 缓存 MethodInterceptor 实现 - CacheInterceptor

# Spring AOP 在 Spring 本地调度 (Scheduling)

- 核心 API
  - Spring 异步 @Enable 模块驱动 - @EnableAsync
  - Spring 异步注解 - @Async
  - Spring 异步配置器 - AsyncConfigurer
  - Spring 异步代理配置 - ProxyAsyncConfiguration
  - Spring 异步 PointcutAdvisor 实现 - AsyncAnnotationAdvisor
  - Spring 异步 MethodInterceptor 实现 - AnnotationAsyncExecutionInterceptor

# 面试题精选

**沙雕面试题** - 请举例说明 Spring AOP 在 Spring Framework 特性运用？

答：

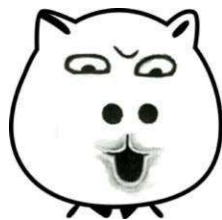
- Spring 事件 (Events)
- Spring 事务 (Transaction)
- Spring 缓存 (Caching)
- Spring 本地调度 (Scheduling)
- Spring 远程 (Remoting)



我真的没笑

# 面试题精选

**996 面试题** - 请解释 Spring 事务传播的原理?



答:

# 面试题精选

**劝退面试题** - 请总结 Spring AOP 与 IoC 功能整合的设计模式？



答：

- 实现 Advice 或 MethodInterceptor
- 实现 PointcutAdvisor
- 实现 Spring AOP 代理配置类
- （可选）实现注解和注解元信息的解析以及处理
- （可选）实现 XML 与其元信息的解析以及处理



扫码试看/订阅

《小马哥讲 Spring AOP 编程思想》视频课程