

第十七章: Spring 事件

小马哥·mercyblitz





扫码试看/订阅

《小马哥讲 Spring 核心编程思想》 视频课程

Spring 事件

- 1. Java 事件/监听器编程模型
- 2. 面向接口的事件/监听器设计模式
- 3. 面向注解的事件/监听器设计模式
- 4. Spring 标准事件-ApplicationEvent
- 5. 基于接口的 Spring 事件监听器
- 6. 基于注解的 Spring 事件监听器
- 7. 注册 Spring ApplicationListener
- 8. Spring 事件发布器
- 9. Spring 层次性上下文事件传播
- 10. Spring 内建事件



Spring 事件

- 11. Spring 4.2 Payload 事件
- 12. 自定义 Spring 事件
- 13. 依赖注入 Application Event Publisher
- 14. 依赖查找 ApplicationEventMulticaster
- 15. ApplicationEventPublisher 底层实现
- 16. 同步和异步 Spring 事件广播
- 17. Spring 4.1 事件异常处理
- 18. Spring 事件/监听器实现原理
- 19. 课外资料
- 20. 面试题精选





Java 事件/监听器编程模型

- 设计模式 观察者模式扩展
 - 可观者对象(消息发送者) java.util.Observable
 - 观察者 java.util.Observer
- 标准化接口
 - 事件对象 java.util.EventObject
 - 事件监听器 java.util.EventListener



面向接口的事件/监听器设计模式

• 事件/监听器场景举例

Java 技术规范	事件接口	监听器接口	
JavaBeans	java.beans.PropertyChangeEvent	java.beans.PropertyChangeListener	单监听
Java AWT	java.awt.event.MouseEvent	java.awt.event.MouseListener	
Java Swing	javax.swing.event.MenuEvent	javax.swing.event.MenuListener	
Java Preference	java.util.prefs.PreferenceChangeEvent	java.util.prefs.PreferenceChangeListene r	

开阔视野

面向注解的事件/监听器设计模式

• 事件/监听器注解场景举例

Java 技术规范	事件注解	监听器注解
Servlet 3.0+		@javax.servlet.annotation.WebListener
JPA 1.0+	@javax.persistence.PostPersist	类似@ComponentScan
Java Common	@PostConstruct	
EJB 3.0+	@javax.ejb.PrePassivate	
JSF 2.0+	@javax.faces.event.ListenerFor	



Spring 标准事件 - ApplicationEvent

- Java 标准事件 java.util.EventObject 扩展
 - 扩展特性: 事件发生事件戳 时间戳(错别字)

- Spring 应用上下文 ApplicationEvent 扩展 ApplicationContextEvent
 - Spring 应用上下文 (ApplicationContext) 作为事件源
 - 具体实现:
 - org.springframework.context.event.ContextClosedEvent
 - org.springframework.context.event.ContextRefreshedEvent
 - org.springframework.context.event.ContextStartedEvent
 - org.springframework.context.event.ContextStoppedEvent



基于接口的 Spring 事件监听器

- Java 标准事件监听器 java.util.EventListener 扩展
 - 扩展接口 org.springframework.context.ApplicationListener
 - 设计特点: 单一类型事件处理 一次只处理一个类型
 - 处理方法: onApplicationEvent(ApplicationEvent)
 - 事件类型: org.springframework.context.ApplicationEvent



基于注解的 Spring 事件监听器

• Spring 注解 - @org.springframework.context.event.EventListener

特性	说明
设计特点	支持多 ApplicationEvent 类型,无需接口约束
注解目标	方法
是否支持异步执行	支持
是否支持泛型类型事件	支持
是指支持顺序控制	支持,配合@Order注解控制



注册 Spring ApplicationListener

• 方法一: ApplicationListener 作为 Spring Bean 注册

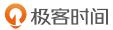
• 方法二: 通过 Configurable Application Context API 注册



Spring 事件发布器

- 方法一: 通过 Application Event Publisher 发布 Spring 事件
 - 获取 ApplicationEventPublisher
 - 依赖注入

- 方法二:通过 Application Event Multicaster 发布 Spring 事件 包含方法一语意
 - 获取 ApplicationEventMulticaster
 - 依赖注入
 - 依赖查找



Spring 层次性上下文事件传播

• 发生说明

当 Spring 应用出现多层次 Spring 应用上下文(ApplicationContext)时,如 Spring WebMVC、Spring Boot 或 Spring Cloud 场景下,由子 ApplicationContext 发起 Spring 事件可能会传递到其 Parent ApplicationContext(直到 Root)的过程

- 如何避免
 - 定位 Spring 事件源(ApplicationContext)进行过滤处理



Spring 内建事件

- ApplicationContextEvent 派生事件
 - ContextRefreshedEvent: Spring 应用上下文就绪事件
 - ContextStartedEvent: Spring 应用上下文启动事件
 - ContextStoppedEvent: Spring 应用上下文停止事件
 - ContextClosedEvent: Spring 应用上下文关闭事件



Spring 4.2 Payload 事件

- Spring Payload 事件 org.springframework.context.PayloadApplicationEvent
 - 使用场景:简化 Spring 事件发送,关注事件源主体
 - 发送方法
 - ApplicationEventPublisher#publishEvent(java.lang.Object)



自定义 Spring 事件

1 • 扩展 org.springframework.context.ApplicationEvent

2 • 实现 org.springframework.context.ApplicationListener

3 • 注册 org.springframework.context.ApplicationListener



依赖注入 Application Event Publisher

• 通过 Application Event Publisher Aware 回调接口

• 通过 @Autowired ApplicationEventPublisher



依赖查找 Application Event Multicaster

底层接口,一般不给开发人员使用

一对多,类似于发布订阅

- 查找条件
 - Bean 名称: "applicationEventMulticaster"
 - Bean 类型: org.springframework.context.event.ApplicationEventMulticaster



ApplicationEventPublisher 底层实现

- 底层实现
 - 接口: org.springframework.context.event.ApplicationEventMulticaster
 - 抽象类: org.springframewor
 - k.context.event.AbstractApplicationEventMulticaster
 - 实现类: org.springframework.context.event.SimpleApplicationEventMulticaster



同步和异步 Spring 事件广播

• 基于实现类 - org.springframework.context.event.SimpleApplicationEventMulticaster

• 模式切换: setTaskExecutor(java.util.concurrent.Executor) 方法

• 默认模式: 同步

• 异步模式:如 java.util.concurrent.ThreadPoolExecutor

• 设计缺陷: 非基于接口契约编程



同步和异步 Spring 事件广播

• 基于注解 - @org.springframework.context.event.EventListener

• 模式切换

• 默认模式: 同步

• 异步模式:标注@org.springframework.scheduling.annotation.Async

缺陷: 方法上的局部生效, 非单例

• 实现限制:无法直接实现同步/异步动态切换



Spring 4.1 事件异常处理

- Spring 3.0 错误处理接口 org.springframework.util.ErrorHandler
 - 使用场景
 - Spring 事件(Events)
 - SimpleApplicationEventMulticaster Spring 4.1 开始支持
 - Spring 本地调度 (Scheduling)
 - org.springframework.scheduling.concurrent.ConcurrentTaskScheduler
 - org.springframework.scheduling.concurrent.ThreadPoolTaskScheduler



Spring 事件/监听器实现原理

- 核心类 org.springframework.context.event.SimpleApplicationEventMulticaster
 - 设计模式:观察者模式扩展
 - 被观察者 org.springframework.context.ApplicationListener
 - API 添加
 - 依赖查找
 - 通知对象 org.springframework.context.ApplicationEvent
 - 执行模式: 同步/异步
 - 异常处理: org.springframework.util.ErrorHandler
 - 泛型处理: org.springframework.core.ResolvableType



课外资料

• Spring Boot 事件

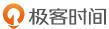
事件类型	发生时机
ApplicationStartingEvent	当 Spring Boot 应用已启动时
ApplicationStartedEvent	当 Spring Boot 应用已启动时
ApplicationEnvironmentPreparedEvent	当 Spring Boot Environment 实例已准备时
ApplicationPreparedEvent	当 Spring Boot 应用预备时
ApplicationReadyEvent	当 Spring Boot 应用完全可用时
ApplicationFailedEvent	当 Spring Boot 应用启动失败时



课外资料

• Spring Cloud 事件

事件类型	发生时机
EnvironmentChangeEvent	当 Environment 示例配置属性发生变化时
HeartbeatEvent	当 DiscoveryClient 客户端发送心跳时
InstancePreRegisteredEvent	当服务实例注册前
InstanceRegisteredEvent	当服务实例注册后
RefreshEvent	当 RefreshEndpoint 被调用时
RefreshScopeRefreshedEvent	当 Refresh Scope Bean 刷新后



面试题

沙雕面试题 - Spring 事件核心接口/组件?



我真的没笑

答:

- Spring 事件 org.springframework.context.ApplicationEvent
- Spring 事件监听器 org.springframework.context.ApplicationListener
- Spring 事件发布器 org.springframework.context.ApplicationEventPublisher
- Spring 事件广播器 org.springframework.context.event.ApplicationEventMulticaster



面试题

996 面试题 - Spring 同步和异步事件处理的使用场景?



答:

- Spring 同步事件 绝大多数 Spring 使用场景,如 ContextRefreshedEvent
- Spring 异步事件 主要 @EventListener 与 @Asyc 配合,实现异步处理,不阻塞主线程,比如长时间的数据计算任务等。不要轻易调整 SimpleApplicationEventMulticaster 中关联的 taskExecutor 对象,除非使用者非常了解 Spring 事件机制,否则容易出现异常行为。



面试题

劝退面试题 - @EventListener 的工作原理?

答: 答案下章揭晓







扫码试看/订阅

《小马哥讲 Spring 核心编程思想》 视频课程