# **Spring监听器之ApplicationListener原理及源码解析及实例**



[林海静](https://blog.csdn.net/jinhaijing" \t "https://blog.csdn.net/jinhaijing/article/details/_blank) 2018-11-12 20:32:09 IMG_257 12469 IMG_258 收藏 15

分类专栏： [spring](https://blog.csdn.net/jinhaijing/category_7723292.html" \t "https://blog.csdn.net/jinhaijing/article/details/_blank) [spring监听器](https://blog.csdn.net/jinhaijing/category_8340984.html" \t "https://blog.csdn.net/jinhaijing/article/details/_blank) 文章标签： [Spring扩展原理之ApplicationListener](https://so.csdn.net/so/search/s.do?q=Spring%E6%89%A9%E5%B1%95%E5%8E%9F%E7%90%86%E4%B9%8BApplicationListener&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/jinhaijing/article/details/_blank)[ApplicationListener原理](https://so.csdn.net/so/search/s.do?q=ApplicationListener%E5%8E%9F%E7%90%86&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/jinhaijing/article/details/_blank) [ApplicationListener源码解析](https://so.csdn.net/so/search/s.do?q=ApplicationListener%E6%BA%90%E7%A0%81%E8%A7%A3%E6%9E%90&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/jinhaijing/article/details/_blank) [ApplicationListener实例](https://so.csdn.net/so/search/s.do?q=ApplicationListener%E5%AE%9E%E4%BE%8B&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/jinhaijing/article/details/_blank)

版权

### **一、原理及源码解析**

****事件：****ContextRefreshedEvent、IOCTest\_Ext$1[source=我发布的事件]、ContextClosedEvent；  
 \*  1）、ContextRefreshedEvent事件：  
 \*      1）、容器创建对象：refresh()；  
 \*      2）、finishRefresh();容器刷新完成会发布ContextRefreshedEvent事件  
 \*  2）、自己发布事件；  
 \*  3）、容器关闭会发布ContextClosedEvent；  
 \*    
****\*  【事件发布流程】源码执行流程：****  
 \*      3）、publishEvent(new ContextRefreshedEvent(this));  
 \*              1）、获取事件的多播器（派发器）：getApplicationEventMulticaster()  
 \*              2）、multicastEvent派发事件：  
 \*              3）、获取到所有的ApplicationListener；  
 \*                  for (final ApplicationListener<?> listener : getApplicationListeners(event, type)) {  
 \*                  1）、如果有Executor，可以支持使用Executor进行异步派发；  
 \*                      Executor executor = getTaskExecutor();  
 \*                  2）、否则，同步的方式直接执行listener方法；invokeListener(listener, event);  
 \*                   拿到listener回调onApplicationEvent方法；  
 \*    
****\*  【事件多播器（派发器）】源码执行流程：****  
 \*      1）、容器创建对象：refresh();  
 \*      2）、initApplicationEventMulticaster();初始化ApplicationEventMulticaster；  
 \*          1）、先去容器中找有没有id=“applicationEventMulticaster”的组件；  
 \*          2）、如果没有this.applicationEventMulticaster = new SimpleApplicationEventMulticaster(beanFactory);  
 \*              并且加入到容器中，我们就可以在其他组件要派发事件，自动注入这个applicationEventMulticaster；  
 \*    
****\*  【容器中有哪些监听器】源码执行流程：****  
 \*      1）、容器创建对象：refresh();  
 \*      2）、registerListeners();  
 \*          从容器中拿到所有的监听器，把他们注册到applicationEventMulticaster中；  
 \*          String[] listenerBeanNames = getBeanNamesForType(ApplicationListener.class, true, false);  
 \*          //将listener注册到ApplicationEventMulticaster中  
 \*          getApplicationEventMulticaster().addApplicationListenerBean(listenerBeanName);  
 \*            
****\*   SmartInitializingSingleton 原理：****->afterSingletonsInstantiated();  
 \*           1）、ioc容器创建对象并refresh()；  
 \*           2）、finishBeanFactoryInitialization(beanFactory);初始化剩下的单实例bean；  
 \*               1）、先创建所有的单实例bean；getBean();  
 \*               2）、获取所有创建好的单实例bean，判断是否是SmartInitializingSingleton类型的；  
 \*                   如果是就调用afterSingletonsInstantiated();  
 \*

### **二、实例**

ApplicationListener：监听容器中发布的事件。事件驱动模型开发；  
 \*       public interface ApplicationListener<E extends ApplicationEvent>  
 \*         监听 ApplicationEvent 及其下面的子事件；  
 \*   
 \*      步骤：  
 \*         1）、写一个监听器（ApplicationListener实现类）来监听某个事件（ApplicationEvent及其子类）  
 \*             @EventListener;  
 \*             原理：使用EventListenerMethodProcessor处理器来解析方法上的@EventListener；  
 \*   
 \*         2）、把监听器加入到容器；  
 \*         3）、只要容器中有相关事件的发布，我们就能监听到这个事件；  
 \*                 ContextRefreshedEvent：容器刷新完成（所有bean都完全创建）会发布这个事件；  
 \*                 ContextClosedEvent：关闭容器会发布这个事件；  
 \*         4）、发布一个事件：  
 \*                 applicationContext.publishEvent()；

****1、自定义监听器****

定义MyApplicationListener 类并实现ApplicationListener<ApplicationEvent>接口

用@Component装配到Spring容器中

package com.atguigu.ext;

import org.springframework.context.ApplicationEvent;

import org.springframework.context.ApplicationListener;

import org.springframework.context.event.EventListener;

import org.springframework.stereotype.Component;

@Component

public class MyApplicationListener implements ApplicationListener<ApplicationEvent> {

*//当容器中发布此事件以后，方法触发*

@Override

public void onApplicationEvent(ApplicationEvent event) {

*// TODO Auto-generated method stub*

System.out.println("收到事件："+event);

}

}

或者使用@EventListener注解来定义一个监听器

注：@EventListener(classes={ApplicationEvent.class})中classes监听的类可以是多个，以逗号分开

package com.atguigu.ext;

import org.springframework.context.ApplicationEvent;

import org.springframework.context.event.EventListener;

import org.springframework.stereotype.Service;

@Service

public class UserService {

@EventListener(classes={ApplicationEvent.class})

public void listen(ApplicationEvent event){

System.out.println("UserService。。监听到的事件："+event);

}

}

****2、定义配置类，加载IOC容器****

@ComponentScan("com.atguigu.ext")

@Configuration

public class ExtConfig {

@Bean

public Blue blue(){

return new Blue();

}

}

****3、测试****

自定义发布事件使用了匿名内部类

package com.atguigu.test;

import org.junit.Test;

import org.springframework.context.ApplicationEvent;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.atguigu.ext.ExtConfig;

public class IOCTest\_Ext {

@Test

public void test01(){

AnnotationConfigApplicationContext applicationContext = new AnnotationConfigApplicationContext(ExtConfig.class);

//发布事件；

applicationContext.publishEvent(new ApplicationEvent(new String("我发布的事件")) {

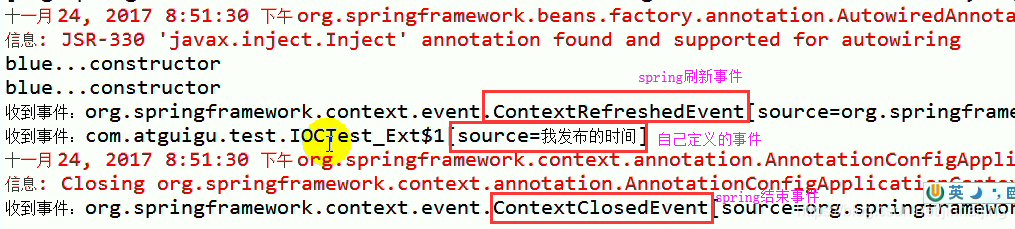
});

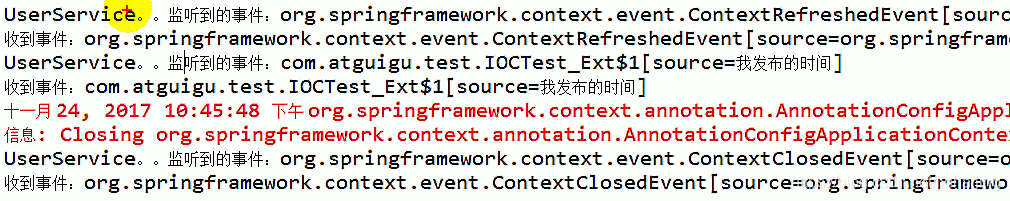
applicationContext.close();

}

}

****结果：****





****说明：****spring自己有自定义监听器，如容器有刷新和关闭事件发布就会被监听到；自己定义了监听器并装配到spring容器中，一旦有相关事件发布也可以被监听到。

# ======以下于你或许是个好消息======

****好消息就是：****欢迎访问下面的博客网站哈哈哈......

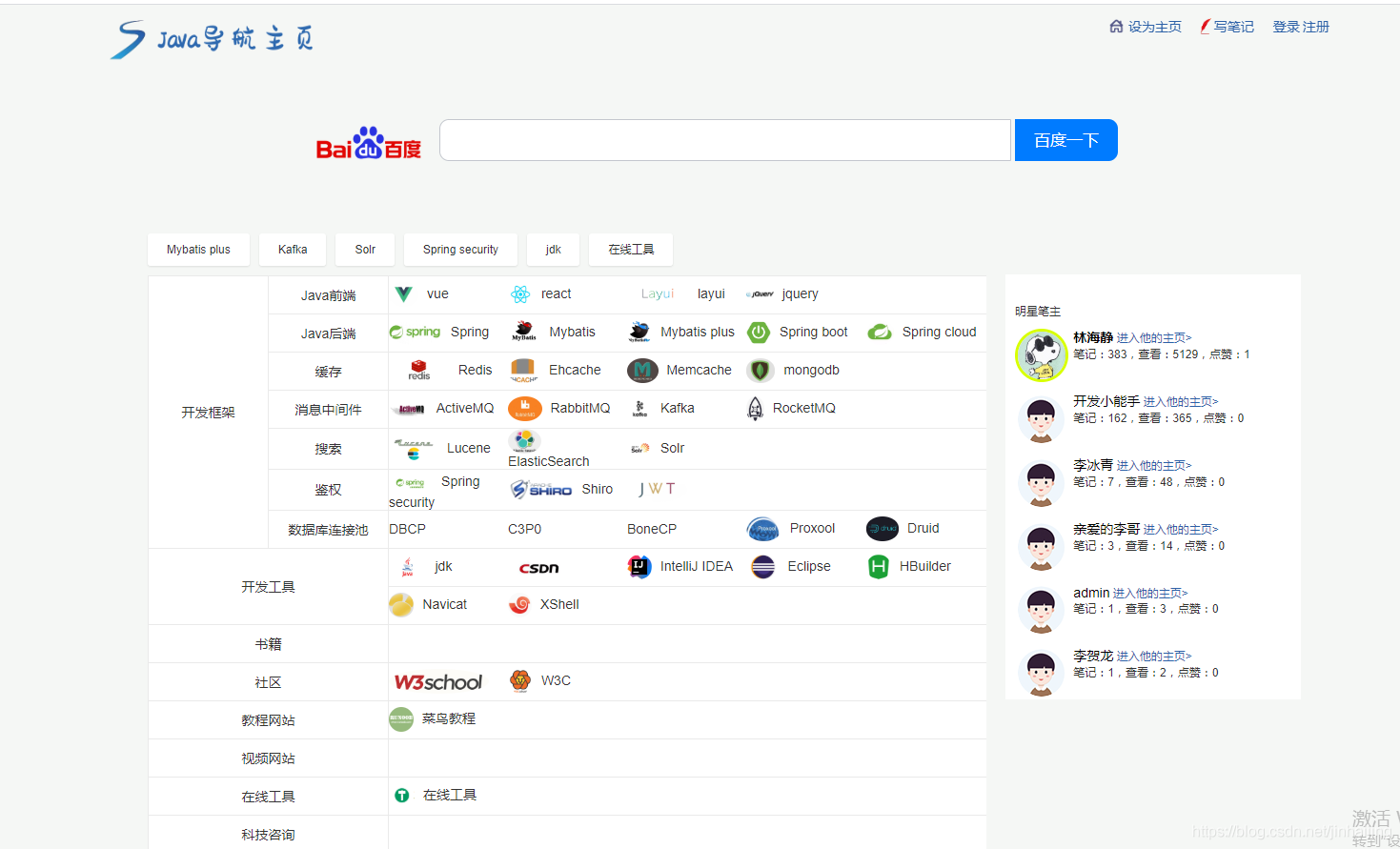
****网站名称：****[Java学习笔记网](https://www.javaxxbj.com/) (点击进入)

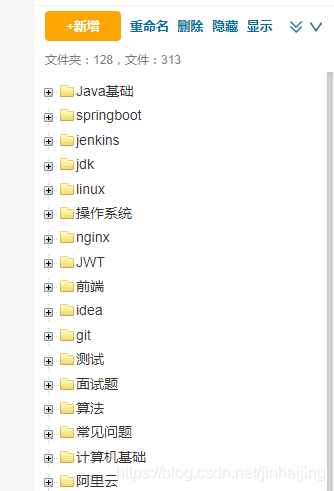
****url：****<https://www.javaxxbj.com/> (点击进入)

****网站特点：****

1. java主要网站的导航目录
2. 你可以记录自己的博客，并可以控制显示和隐藏，可利于管理啦！！！
3. 可以添加收藏各个网站的链接！！！
4. 甚至也可以文章收藏，点赞，关注，查看我的消息等功能哦！！1

****看一小点点的截图：****





或可一试哦！