# Spring/Spring boot中使用自定义事件监听

2018年07月05日 11:53:48

阅读数：6 [编辑](https://mp.csdn.net/postedit/80924639)

在实际开发过程中，常常遇到这种场景：   
做完某一件事情以后，需要广播一些消息或者通知，告诉其他的模块进行一些事件处理，一般来说，可以一个一个发送请求去通知，但是有一种更好的方式，那就是事件监听，事件监听也是设计模式中 发布-订阅模式、观察者模式的一种实现。

观察者模式：在对象之间定义了一对多的依赖，这样一来，当一个对象改变状态，依赖它的对象会收到通知并自动更新。

Spring的事件为Bean和Bean之间的消息传递提供支持。当一个对象处理完某种任务后，通知另外的对象进行某些处理，常用的场景有进行某些操作后发送通知，消息、邮件等情况。   
Spring提供5种标准的事件监听：

1. 上下文更新事件（ContextRefreshedEvent）：该事件会在ApplicationContext被初始化或者更新时发布。也可以在调用ConfigurableApplicationContext接口中的refresh()方法时被触发。
2. 上下文开始事件（ContextStartedEvent）：当容器ConfigurableApplicationContext的Start()方法开始/重新开始容器时触发该事件。
3. 上下文停止事件（ContextStoppedEvent）：当容ConfigurableApplicationContext的Stop()方法停止容器时触发该事件。
4. 上下文关闭事件（ContextClosedEvent）：当ApplicationContext被关闭时触发该事件。容器被关闭时，其管理的所有单例Bean都被销毁。
5. 请求处理事件（RequestHandledEvent）：在Web应用中，当一个http请求（request）结束触发该事件。

不过也有些实际场景并用不上框架提供的标准事件，这个时候我们就需要自定义事件监听   
Spring的事件遵循的流程：

1. 自定义事件，继承ApplicationEvent（org.springframework.context.ApplicationEvent）
2. 定义监听事件，实现ApplicationListener（org.springframework.context.ApplicationListener）
3. 使用容器触发事件

Spring中书写事件监听有两种方式，编程式和注解式，这里两种都进行简单的说明。因为现在正好在看boot方面的东西，这里就以boot项目为基础开启事件监听，boot也是spring全家桶的一部分，所以配置和使用和传统spring项目也没什么区别，当然，boot提倡无xml，所以这里开启监听的方式也用注解的方式进行。

一：自定义事件

1. package com.cnjy.ecampus.modules.cms.domain;
2. import org.springframework.context.ApplicationEvent;
4. import com.cnjy.ecampus.modules.um.domain.User;
6. */\*\**
7. *\* 自定义监听方法类*
8. *\*/*
9. public class ListenerRabbitEvent extends ApplicationEvent{

12. */\*\**
13. *\* 操作对象 可以是对应的list*
14. *\*/*
15. private Object operationObj;
16. */\**
17. *\* 发送类型 add update delete*
18. *\*/*
19. private String sendType;
20. */\*\**
21. *\* 类型 class student teacher users*
22. *\*/*
23. private String type;
24. */\*\**
25. *\* 提示信息*
26. *\*/*
27. private String hitMsg;
28. */\*\**
29. *\* 登录用户*
30. *\*/*
31. private User loginUser;

34. */\*\**
35. *\* 在自定义事件的构造方法中除了第一个source参数，其他参数都可以去自定义，*
36. *\* 可以根据项目实际情况进行监听传参，这里就只定义个简单的String字符串的透传*
37. *\* @param source*
38. *\* @param operationObj sendType type hitMsg loginUser*
39. *\*/*
40. public ListenerRabbitEvent(Object source, Object operationObj, String sendType, String type
41. , String hitMsg, User loginUser) {
42. super(source);
43. this.operationObj = operationObj;
44. this.sendType = sendType;
45. this.type = type;
46. this.hitMsg = hitMsg;
47. this.loginUser = loginUser;
48. }

51. public Object getOperationObj() {
52. return operationObj;
53. }

56. public void setOperationObj(Object operationObj) {
57. this.operationObj = operationObj;
58. }

61. public String getSendType() {
62. return sendType;
63. }

66. public void setSendType(String sendType) {
67. this.sendType = sendType;
68. }

71. public String getType() {
72. return type;
73. }

76. public void setType(String type) {
77. this.type = type;
78. }

81. public String getHitMsg() {
82. return hitMsg;
83. }

86. public void setHitMsg(String hitMsg) {
87. this.hitMsg = hitMsg;
88. }

91. public User getLoginUser() {
92. return loginUser;
93. }

96. public void setLoginUser(User loginUser) {
97. this.loginUser = loginUser;
98. }
100. */\*\**
101. *\* 自定义监听器触发的透传打印方法*
102. *\* @param msg*
103. *\*/*
104. *// public void printMsg(String msg)*
105. *// {*
106. *// System.out.println("编程事件监听:" + msg);*
107. *// }*

110. }

二：事件监听

1. package com.cnjy.ecampus.ribbitmq;
2. import javax.annotation.Resource;
3. import javax.inject.Qualifier;
5. import org.springframework.beans.factory.annotation.Autowired;
6. import org.springframework.context.ApplicationListener;
7. import org.springframework.stereotype.Component;
9. import com.cnjy.ecampus.framework.util.LogUtils;
10. import com.cnjy.ecampus.modules.cms.domain.ListenerRabbitEvent;
12. */\*\**
13. *\* 测试用自定义监听器,监听事件为MyEvent*
14. *\*/*
15. @Component
16. public class LisenterRabbitMsg implements ApplicationListener<ListenerRabbitEvent>{
18. @Resource
19. private RibbitManage ribbitManage;
20. */\*\**
21. *\* 对监听到的事件进行处理*
22. *\* @param myEvent*
23. *\*/*
24. @Override
25. public void onApplicationEvent(ListenerRabbitEvent listenerRabbitEvent) {
27. LogUtils.info("=====================已经监听到并执行处理监听后的逻辑（此处根据自己业务不同进行不同的处理）=======================");
28. *//发送ribbit消息*
29. boolean result = ribbitManage.sendRibbitMsg(listenerRabbitEvent.getOperationObj(), listenerRabbitEvent.getSendType(),
30. listenerRabbitEvent.getType(), listenerRabbitEvent.getHitMsg(), listenerRabbitEvent.getLoginUser());
31. if (!result) {
32. LogUtils.info("发送ribbit消息失败---------------------------" );
33. LogUtils.info("sendType=" + listenerRabbitEvent.getSendType() + ",=" + listenerRabbitEvent.getType()
34. + ",hitMsg=" + listenerRabbitEvent.getHitMsg());
35. }

38. }

41. }

三：事件发布

1. package com.cnjy.ecampus.modules.cms.service;
3. import com.cnjy.ecampus.modules.um.domain.User;
5. public interface ListenerRabbitService{
6. public void publish(Object operationObj, String sendType, String type
7. , String hitMsg, User loginUser);
8. }
9. package com.cnjy.ecampus.modules.cms.service;
10. import org.springframework.context.ApplicationContext;
11. import org.springframework.stereotype.Service;
13. import com.cnjy.ecampus.framework.util.LogUtils;
14. import com.cnjy.ecampus.modules.cms.domain.ListenerRabbitEvent;
15. import com.cnjy.ecampus.modules.um.domain.User;
17. import javax.annotation.Resource;
19. @Service("listenerRabbitService")
20. public class ListenerRabbitServieImpl implements ListenerRabbitService{
22. */\*\**
23. *\* 上下文对象*
24. *\*/*
25. @Resource
26. private ApplicationContext applicationContext;
28. @Override
29. public void publish(Object operationObj, String sendType, String type
30. , String hitMsg, User loginUser) {
32. LogUtils.info("--发布监听");
33. *//通过上下文对象发布监听*
34. applicationContext.publishEvent(new ListenerRabbitEvent(this,operationObj, sendType, type,
35. hitMsg, loginUser));
36. }
37. }

测试类入口

1. package com.cnjy.ecampus.listen;
3. import javax.annotation.Resource;
5. import org.springframework.beans.factory.annotation.Autowired;
6. import org.springframework.beans.factory.annotation.Qualifier;
7. import org.springframework.web.bind.annotation.RequestMapping;
8. import org.springframework.web.bind.annotation.RestController;
10. import com.cnjy.ecampus.modules.cms.service.ListenerRabbitService;
11. import com.cnjy.ecampus.modules.um.domain.User;
13. */\*\**
14. *\* 事件监听测试*
15. *\*/*
16. @RestController
17. public class listenerTest {
19. @Resource
20. private ListenerRabbitService listenerRabbitService;
22. @RequestMapping("/listen/test")
23. public void testEvent()
24. {
25. System.out.println("------------");
26. *//此处发布监听*
27. Object obj = "obj";
28. User loginUser = new User();
29. listenerRabbitService.publish(obj, "add", "users", "需要添加的用户", loginUser);
30. }
31. }

java 技术交流群：317628801