[spring事件监听](http://blog.csdn.net/sd0902/article/details/8394082)

标签： [java](http://www.csdn.net/tag/java)[Java](http://www.csdn.net/tag/Java)[JAVA](http://www.csdn.net/tag/JAVA)

2012-12-24 12:53 4089人阅读 [评论](http://blog.csdn.net/sd0902/article/details/8394082#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/sd0902/article/details/8394082#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

spring（7） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

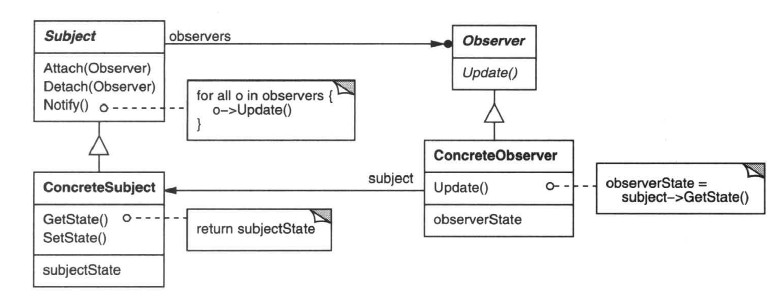
目录[(?)[+]](http://blog.csdn.net/sd0902/article/details/8394082)

**一、一些概念**

　事件是可以被控件识别的操作，如按下确定按钮，选择某个单选按钮或者复选框。每一种控件有自己可以识别的事件，如窗体的加载、单击、双击等事件，编辑框(文本框)的文本改变事，等等。事件有系统事件和用户事件。系统事件由系统激发，如时间每隔24小时，银行储户的存款日期增加一天。用户事件由用户激发，如用户点击按钮，在文本框中显示特定的文本。事件驱动控件执行某项功能。

 触发事件的对象称为事件发送者；接收事件的对象称为事件接收者。

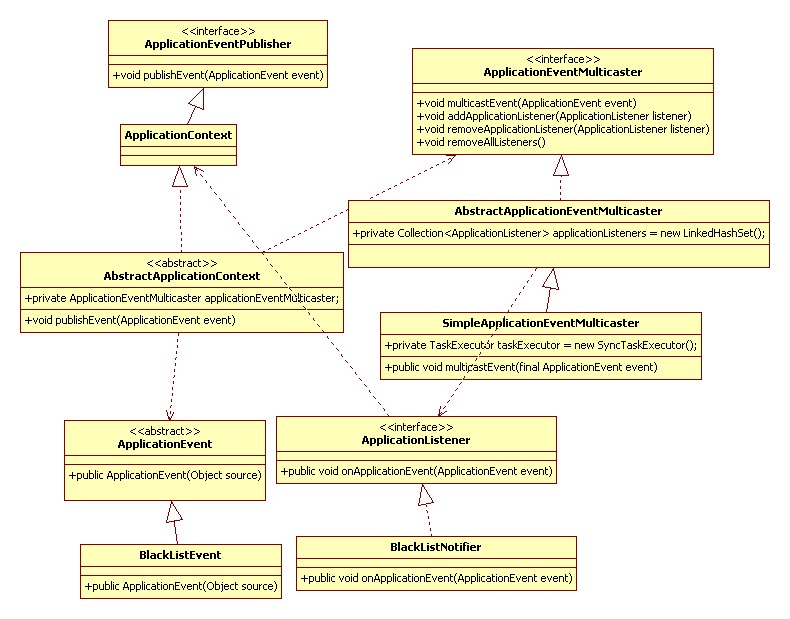
 事件模型一般用到了观察者模式，下面简单介绍下：



* SUBJECT
  + 目标知道它的观察者。可以有任意多个观察者观察同一个目标。
  + 提供注册和删除观察者对象的接口。
* Observer（观察者）
  + 为那些在目标发生改变时需获得通知的对象定义一个更新接口。
* ConcreteSubject（具体目标）
  + 将有关状态存入各ConcreteObserver对象。
  + 当它的状态发生改变时,向它的各个观察者发出通知。
* ConcreteObserver（具体观察者）
  + 维护一个指向ConcreteSubject对象的引用。
  + 存储有关状态，这些状态应与目标的状态保持一致。
  + 实现Observer的更新接口以使自身状态与目标的状态保持一致。

**二、spring事件机制**

首先上图一张：



* ApplicationListener 就是我们的 Observer，需要到容器中注册。他要关心他所关心的ApplicationEvent 。一般有如下代码：if (event instanceof BlackListEvent) {}
* ApplicationEventMulticaster是我们的SUBJECT一个代理。他会管理我们的 ApplicationListener 。
* ApplicationEvent 是事件，它就是媒介，充当介质的作用。

除了我们可以自己实现Spring容器的事件外，Spring也提供了几个内置事件：

          1、  ContextRefreshedEvent：ApplicationContext容器初始化或者刷新时触发该事件。

          2、  ContextStartedEvent：当使用ConfigurableApplicationContext接口的start()方法启动ApplicationContext容器时触发该事件。

          3、  ContextClosedEvent：当使用ConfigurableApplicationContext接口的close()方法关闭ApplicationContext容器时触发该事件。

          4、  ContextStopedEvent: 当使用ConfigurableApplicationContext接口的stop()方法停止ApplicationContext容器时触发该事件。

在spring中,容器管理所有的 bean。是ApplicationEvent 驱动的，一个ApplicationEvent  publish了，观察这个事件的监听者就会送到通知。

具体核心代码如下: ApplicationEventMulticaster 会遍历所有的 监听器，再启动一个线程调用监听器的onApplicationEvent方法。

**[java]** [view plaincopy](http://blog.csdn.net/bxyz1203/article/details/6253541)

1. **public** **void** multicastEvent(**final** ApplicationEvent event) {
2. **for** (Iterator it = getApplicationListeners().iterator(); it.hasNext();) {
3. **final** ApplicationListener listener = (ApplicationListener) it.next();
4. getTaskExecutor().execute(**new** Runnable() {
5. **public** **void** run() {
6. listener.onApplicationEvent(event);
7. }
8. });
9. }
10. }

**三、实例代码**

          场景：程序发现一个有问题的用户，再会把这个的用户记录下来，采取 事件模型就这样设计

* + 发现有问题的用户
  + 把问题记录下来
  + 媒介也就是事件

**1、先来一个事件**

**[java]** [view plaincopy](http://blog.csdn.net/bxyz1203/article/details/6253541)

* 1. **package** spring;
  2. **import** org.springframework.context.ApplicationEvent;
  3. **public** **class** BlackListEvent **extends** ApplicationEvent {
  4. **private** **static** **final** **long** serialVersionUID = 1000L;
  5. **private** String            address;
  6. **public** String getAddress() {
  7. **return** address;
  8. }
  9. **public** BlackListEvent(String address) {
  10. **super**(address);
  11. **this**.address = address;
  12. }
  13. }

**2、一个监听器**

**[java]** [view plaincopy](http://blog.csdn.net/bxyz1203/article/details/6253541)

* 1. **package** spring;
  2. **import** org.springframework.context.ApplicationEvent;
  3. **import** org.springframework.context.ApplicationListener;
  4. **public** **class** BlackListNotifier **implements** ApplicationListener {
  5. **public** **void** onApplicationEvent(ApplicationEvent event) {
  6. System.out.println(event);
  7. **if** (event **instanceof** BlackListEvent) {
  8. System.out.println(((BlackListEvent) event).getAddress());
  9. //TODO 处理这个邮件地址
  10. }
  11. }
  12. }

**3、发现有问题的用户，并且publish**

**[java]** [view plaincopy](http://blog.csdn.net/bxyz1203/article/details/6253541)

* 1. **package** spring;
  2. **import** java.util.List;
  3. **import** org.springframework.beans.BeansException;
  4. **import** org.springframework.context.ApplicationContext;
  5. **import** org.springframework.context.ApplicationContextAware;
  6. **public** **class** EmailBean **implements** ApplicationContextAware {
  7. **private** List<String>       blackList;
  8. **private** ApplicationContext ctx;
  9. **public** **void** setBlackList(List<String> blackList) {
  10. **this**.blackList = blackList;
  11. }
  12. **public** **void** setApplicationContext(ApplicationContext applicationContext) **throws** BeansException {
  13. **this**.ctx = applicationContext;
  14. }
  15. **public** **void** sendEmail(String address) {
  16. **if** (blackList.contains(address)) {
  17. BlackListEvent event = **new** BlackListEvent(address);
  18. ctx.publishEvent(event);
  19. **return**;
  20. }
  21. }
  22. }

**4、配置文件**

**[xhtml]** [view plaincopy](http://blog.csdn.net/bxyz1203/article/details/6253541)

* 1. **<?xml** version="1.0" encoding="UTF-8"**?>**
  2. **<beans** xmlns="http://www.springframework.org/schema/beans"
  3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  4. xsi:schemaLocation="http://www.springframework.org/schema/beans
  5. http://www.springframework.org/schema/beans/spring-beans-2.5.xsd"
  6. default-autowire="byName"**>**
  8. **<bean** id="emailer" class="spring.EmailBean"**>**
  9. **<property** name="blackList"**>**
  10. **<list>**
  11. **<value>**black@list.org**</value>**
  12. **<value>**white@list.org**</value>**
  13. **<value>**john@doe.org**</value>**
  14. **</list>**
  15. **</property>**
  16. **</bean>**
  17. **<bean** id="blackListListener" class="spring.BlackListNotifier" **/>**
  18. **</beans>**

**5、调用方法**

**[java]** [view plaincopy](http://blog.csdn.net/bxyz1203/article/details/6253541)

* 1. **package** spring;
  2. **import** org.springframework.context.ApplicationContext;
  3. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
  4. **public** **class** Test {
  5. **public** **static** **void** main(String[] args) {
  6. ApplicationContext ctx = **new** ClassPathXmlApplicationContext("spring.xml");
  7. EmailBean email = (EmailBean) ctx.getBean("emailer");
  8. email.sendEmail("black@list.org");
  9. }
  10. }