Event Handling in Spring

Spring Framework also provide an another feature to promote loose coupling which is Application Event handling. Using Events, an Event publisher object can communicate with other objects without even knowing which object is listen. and event listener can work to event without knowing which object publish the events. Publisher Object that object who publish the event or call the event and Listener always listen the events that are occurs.

To make a custom event extends the class with ApplicationEvent class. The ApplicationEventPublisher has contain publishEvent() method that enable to publish ApplicationEvents (Custom Events).Any ApplicationListener that is registered with the onApplicationEvent() method in application,listen the events.

To understand The Event handling Lets illustrate a simple example-

**Circle.java**

1. **package** com.dineshonjava.sdnext.eventHandling.tutorial;
3. **import** javax.annotation.Resource;
5. **import** org.springframework.context.ApplicationEventPublisher;
6. **import** org.springframework.context.ApplicationEventPublisherAware;
7. **import** org.springframework.context.MessageSource;
8. **import** org.springframework.stereotype.Controller;
10. /\*\*
11. \* @author
12. \*
13. \*/
14. @Controller
15. **public** **class** Circle **implements** ApplicationEventPublisherAware
16. {
17. **private** Point             center;
18. **private** ApplicationEventPublisher publisher;
20. /\*\*
21. \* @param center the center to set
22. \*/
23. @Resource(name="pointB")
24. **public** **void** setCenter(Point center)
25. {
26. **this**.center = center;
27. }
29. **public** **void** draw()
30. {
31. System.out.println("Circle is Drawn");
32. DrawEvent drawEvent = **new** DrawEvent(**this**);
33. publisher.publishEvent(drawEvent);
34. }
36. @Override
37. **public** **void** setApplicationEventPublisher(ApplicationEventPublisher publisher)
38. {
39. **this**.publisher = publisher;
40. }
41. }

**Point.java**

[view plainprint?](http://www.dineshonjava.com/2012/07/event-handling-in-spring.html)

1. **package** com.dineshonjava.sdnext.eventHandling.tutorial;
3. /\*\*
4. \* @author
5. \*
6. \*/
7. **public** **class** Point
8. {
9. **private** **int** x;
10. **private** **int** y;
11. /\*\*
12. \* @return the x
13. \*/
14. **public** **int** getX()
15. {
16. **return** x;
17. }
18. /\*\*
19. \* @param x the x to set
20. \*/
21. **public** **void** setX(**int** x)
22. {
23. **this**.x = x;
24. }
25. /\*\*
26. \* @return the y
27. \*/
28. **public** **int** getY()
29. {
30. **return** y;
31. }
32. /\*\*
33. \* @param y the y to set
34. \*/
35. **public** **void** setY(**int** y)
36. {
37. **this**.y = y;
38. }
39. }

**MyEventListener.java**

[view plainprint?](http://www.dineshonjava.com/2012/07/event-handling-in-spring.html)

1. **package** com.dineshonjava.sdnext.eventHandling.tutorial;
3. **import** org.springframework.context.ApplicationEvent;
4. **import** org.springframework.context.ApplicationListener;
5. **import** org.springframework.stereotype.Component;
7. @Component
8. **public** **class** MyEventListener **implements** ApplicationListener
9. {
11. @Override
12. **public** **void** onApplicationEvent(ApplicationEvent event)
13. {
14. System.out.println(event.toString());
15. }
17. }

**DrawEvent.java**

[view plainprint?](http://www.dineshonjava.com/2012/07/event-handling-in-spring.html)

1. **package** com.dineshonjava.sdnext.eventHandling.tutorial;
3. **import** org.springframework.context.ApplicationEvent;
5. **public** **class** DrawEvent **extends** ApplicationEvent
6. {
8. /\*\*
9. \*
10. \*/
11. **private** **static** **final** **long** serialVersionUID = 6973014356268900607L;
13. **public** DrawEvent(Object source)
14. {
15. **super**(source);
16. }
18. **public** String toString()
19. {
20. **return** "Draw event occurred";
21. }
22. }

**spring.xml**

[view plainprint?](http://www.dineshonjava.com/2012/07/event-handling-in-spring.html)

1. **<beans** xmlns:context="http://www.springframework.org/schema/context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.springframework.org/schema/beans" xsi:schemalocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
2. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"**>**
3. **<context:annotation-config></context:annotation-config>**
4. **<bean** class="com.sdnext.dineshonjava.eventHandling.tutorial.Point" id="pointB"**>**
5. **<property** name="x" value="10"**></property>**
6. **<property** name="y" value="20"**></property>**
7. **</bean>**
8. **<context:component-scan** base-package="com.sdnext.dineshonjava.eventHandling.tutorial"**></context:component-scan>**
9. **</beans>**

**DrawingApp.java**

[view plainprint?](http://www.dineshonjava.com/2012/07/event-handling-in-spring.html)

1. **package** com.dieshonjava.sdnext.eventHandling.tutorial;
3. **import** org.springframework.context.ApplicationContext;
4. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
6. /\*\*
7. \* @author
8. \*
9. \*/
10. **public** **class** DrawingApp
11. {
13. /\*\*
14. \* @param args
15. \*/
16. **public** **static** **void** main(String[] args)
17. {
18. ApplicationContext context = **new** ClassPathXmlApplicationContext("spring.xml");
19. Circle circle = (Circle)context.getBean("circle");
20. circle.draw();
21. }
22. }