Spring Event

Introduction

Spring framework’s one of most unutilized area is the event handling. Spring has a support to event handling from the very beginning but still you will not find much of the articles in the web on that. I, in this article, will try to get a broader picture of event handling in spring. Whenever there is a talk of event handling one pattern comes into mind that’s observer pattern. Spring comes up with a solution which more or less works like observer pattern but which is more powerful when we can use it in entirety. In observer pattern you have something called observer or listener and something called observable or the publisher. In observer pattern observer is the class that implements java.util.observer interface. The observable is the class that extends java.util.observable class. In spring we have ApplicationEventPublisherAware which is an interface that need to be implemented to make something as publisher. The method that is used to send the message to the Listner is public void send(String text) .The listener is one which implements ApplicationListener. ApplicationEvent is the root class of all the events. You will have to extend this class to make your own event. The Listener will listen using the method public void onApplicationEvent(ApplicationEvent event). In spring Context itself works as the message broker. Now get started with the simple publisher and listener. In our first example we will have a listener that listens to every event. The SimpleEventPublisher is the class that will act as the Publisher in the first example.

Examples on Spring Event

Java code

CustomEvent.java

```
package com.ddlab.rnd.spring.events;
import org.springframework.context.ApplicationEvent;

public class CustomEvent extends ApplicationEvent{

	private static final long serialVersionUID = -8425548734429445606L;

	public CustomEvent(Object source) {
		super(source);
	}

	public String toString(){
		return "My Custom Event";
	}
}
```

CustomEventHandlerListener.java

```
package com.ddlab.rnd.spring.events;
import org.springframework.context.ApplicationListener;

public class CustomEventHandlerListener implements ApplicationListener<CustomEvent>{

   public void onApplicationEvent(CustomEvent event) {
      System.out.println("Event Name :::"+event.toString());
   }

}
```

CustomEventPublisher.java

```java
package com.ddlab.rnd.spring.events;
import org.springframework.context.ApplicationEventPublisher;
import org.springframework.context.ApplicationEventPublisherAware;
public class CustomEventPublisher implements ApplicationEventPublisherAware  {
   private ApplicationEventPublisher publisher;

   public void setApplicationEventPublisher
              (ApplicationEventPublisher publisher){
      this.publisher = publisher;
   }

   public void publish() {
      CustomEvent ce = new CustomEvent(this);
      publisher.publishEvent(ce);
   }
}
```

AnotherCustomEventHandlerListener.java

```java
package com.ddlab.rnd.spring.events;
import org.springframework.context.ApplicationListener;

public class AnotherCustomEventHandlerListener implements ApplicationListener<CustomEvent> {

      public void onApplicationEvent(CustomEvent event) {
            System.out.println("I am the second listener ....");
            System.out.println("Event Name :::"+event.toString());
      }
}
```

Test.java

```java
package com.ddlab.rnd.spring.events;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test
{
      public static void main( String[] args )
      {
            ApplicationContext context =
                        new ClassPathXmlApplicationContext("beans.xml");

            CustomEventPublisher cvp =
                        (CustomEventPublisher) context.getBean("customEventPublisher");
            cvp.publish();
            cvp.publish();
      }
}
```

Beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

       <bean id="customEventHandler" class="com.ddlab.rnd.spring.events.CustomEventHandlerListener" />

       <bean id="customEventHandler1"
             class="com.ddlab.rnd.spring.events.AnotherCustomEventHandlerListener" />

       <bean id="customEventPublisher" class="com.ddlab.rnd.spring.events.CustomEventPublisher" />

       <bean id="helloPublisher" class="com.ddlab.rnd.spring.events1.HelloPublisher" />
       <bean id="helloReceiver" class="com.ddlab.rnd.spring.events1.HelloReceiver" />
</beans>
```

Another Example

HelloEvent.java

```
package com.ddlab.rnd.spring.events1;
import org.springframework.context.ApplicationEvent;

public class HelloEvent extends ApplicationEvent {
    private String name = null;

    public HelloEvent(String name){
        super(name);
        this.name = name;
    }

    public String getName(){
        return name;
    }
}
```

HelloPublisher.java

```
package com.ddlab.rnd.spring.events1;

import org.springframework.context.ApplicationEventPublisher;
import org.springframework.context.ApplicationEventPublisherAware;

public class HelloPublisher implements ApplicationEventPublisherAware{
    private ApplicationEventPublisher aep;

    public void say(){
        HelloEvent he = new HelloEvent(HelloPublisher.class.getSimpleName());
        aep.publishEvent(he);
    }

    @Override
    public void setApplicationEventPublisher(ApplicationEventPublisher aep) {
        this.aep = aep;
    }
}
```

HelloReceiver.java

```java
package com.ddlab.rnd.spring.events1;

import org.springframework.context.ApplicationListener;

public class HelloReceiver implements ApplicationListener<HelloEvent>{

    @Override
    public void onApplicationEvent(HelloEvent e) {
        String name = HelloReceiver.class.getSimpleName();
        System.out.println(name + " heard: " + e.getName() + " said: Hello World!");
    }
}
```

HelloWorld.java

```java
package com.ddlab.rnd.spring.events1;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class HelloWorld {
    public static void main(String... argus){
        ApplicationContext ctx = new ClassPathXmlApplicationContext("beans.xml");
        HelloPublisher hp = (HelloPublisher)ctx.getBean("helloPublisher");
        hp.say();
    }
}
```

The same beans.xml file is used, for more details refer the below.

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

       <bean id="customEventHandler" class="com.ddlab.rnd.spring.events.CustomEventHandlerListener" />

       <bean id="customEventHandler1"
             class="com.ddlab.rnd.spring.events.AnotherCustomEventHandlerListener" />

       <bean id="customEventPublisher" class="com.ddlab.rnd.spring.events.CustomEventPublisher" />

       <bean id="helloPublisher" class="com.ddlab.rnd.spring.events1.HelloPublisher" />
       <bean id="helloReceiver" class="com.ddlab.rnd.spring.events1.HelloReceiver" />
</beans>
```

Spring Events

Java code

AtmMachine.java

```
package com.ddlab.rnd.spring.events;
public interface AtmMachine {

    public void insertCard(Card card);

    public int withdrawMoney(Card card,int amount);
}
```

AtmMachineImpl.java

```
package com.ddlab.rnd.spring.events;
import java.util.Date;
public class AtmMachineImpl implements AtmMachine {

    private EventPublisher eventPublisher;

    public void setEventPublisher(EventPublisher eventPublisher) {
        this.eventPublisher = eventPublisher;
    }

    @Override
    public void insertCard(Card card) {
        CardEvent event = new CardEvent(card);
        event.setCurrentDate(new Date());
        event.setOperation("Insertion");
        eventPublisher.publishEvent(event);
    }

    @Override
    public int withdrawMoney(Card card,int amount) {
       CardEvent event = new CardEvent(card);
       event.setCurrentDate(new Date());
       event.setOperation("Withdrawl");
         eventPublisher.publishEvent(event);
        return amount;
    }
}
```

Card.java

```
package com.ddlab.rnd.spring.events;
public interface Card {

    public String getAccountNo();

    public String getName();
}
```

CardImpl.java

```
package com.ddlab.rnd.spring.events;

public class CardImpl implements Card {

    public String getAccountNo() {
        return "1234";
    }

    public String getName() {
        return "John";
    }
}
```

<u>**CardEvent.java**</u>

```
package com.ddlab.rnd.spring.events;

import java.util.Date;

import org.springframework.context.ApplicationEvent;

public class CardEvent extends ApplicationEvent {

        private static final long serialVersionUID = 5136436079401730585L;
        private Card card;
        private Date currentDate;
        private String operation;

  public CardEvent(Card card) {
    super(card);
    this.card = card;
  }

  public Card getCard() {
    return card;
  }

        public Date getCurrentDate() {
                return currentDate;
        }

        public void setCurrentDate(Date currentDate) {
                this.currentDate = currentDate;
        }

        public String getOperation() {
                return operation;
        }

        public void setOperation(String operation) {
                this.operation = operation;
        }

}
```

EventListener.java

```
package com.ddlab.rnd.spring.events;

import org.springframework.context.ApplicationListener;
public class EventListener implements ApplicationListener<CardEvent> {

    @Override
    public void onApplicationEvent(CardEvent event) {
        System.out.println("Event :::"+event);
        Card card = event.getCard();
        System.out.println("Account No : "+card.getAccountNo());
        System.out.println("Date :::"+event.getCurrentDate());
        System.out.println("Operation :::"+event.getOperation());
    }
}
```

EventPublisher.java

```
package com.ddlab.rnd.spring.events;

import org.springframework.context.ApplicationEventPublisher;
import org.springframework.context.ApplicationEventPublisherAware;

public class EventPublisher implements ApplicationEventPublisherAware {

    private ApplicationEventPublisher publisher;

    @Override
    public void setApplicationEventPublisher(ApplicationEventPublisher publisher) {
        this.publisher = publisher;
    }

    public void publishEvent(CardEvent event) {
        publisher.publishEvent(event);
    }
}
```

Test.java

```
package com.ddlab.rnd.spring.events;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {

  public static void main(String[] ags) {

    ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
    Card card = (Card)context.getBean("card");
    AtmMachine atm = (AtmMachine) context.getBean("atmMachine");
    atm.insertCard(card);
    atm.withdrawMoney(card, 1000);

  }
}
```

Beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
             http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
aop-4.0.xsd">

    <bean id="publisher" class="com.ddlab.rnd.spring.events.EventPublisher"/>

    <bean id="atmMachine" class="com.ddlab.rnd.spring.events.AtmMachineImpl">
        <property name="eventPublisher" ref="publisher"/>
    </bean>

    <bean id="card" class="com.ddlab.rnd.spring.events.CardImpl"/>

    <bean id="eventListener" class="com.ddlab.rnd.spring.events.EventListener"/>

</beans>
```