JAVA实现接口回调

熟悉MS－Windows和X Windows事件驱动设计模式的开发人员，通常是把一个方法的指针传递给事件源，当某一事件发生时来调用这个方法（也称为“回调”）。Java的面向对象的模型目前不支持方法指针，似乎不能使用这种方便的机制。

Java支持interface，通过interface可以实现相同的回调。其诀窍就在于定义一个简单的interface，申明一个被希望回调的方法。

例如，假定当某一事件发生时会得到通知，我们可以定义一个interface：  
public interface InterestingEvent {  
    // 这只是一个普通的方法，可以接收参数、也可以返回值  
    public void interestingEvent();  
}  
  
这样我们就有了任何一个实现了这个接口类对象的手柄grip。  
  
当一事件发生时，需要通知实现**InterestingEvent** 接口的对象，并调用**interestingEvent()** 方法。  
class EventNotifier {  
    private InterestingEvent ie;  
    private boolean somethingHappened;  
  
    public EventNotifier(InterestingEvent event) {  
        ie = event;  
        somethingHappened = false;  
    }  
  
    public void doWork() {  
        if (somethingHappened) {  
            // 事件发生时，通过调用接口的这个方法来通知  
            ie.interestingEvent();  
        }          
    }  
}  
  
在这个例子中，用**somethingHappened** 来标志事件是否发生。  
  
希望接收事件通知的类必须要实现**InterestingEvent** 接口，而且要把自己的引用传递给事件的通知者。  
public class CallMe implements InterestingEvent {  
    private EventNotifier en;  
  
    public CallMe() {  
        // 新建一个事件通知者对象，并把自己传递给它  
        en = new EventNotifier(this);  
    }  
  
    // 实现事件发生时，实际处理事件的方法  
    public void interestingEvent() {  
        // 这个事件发生了，进行处理  
    }  
}  
  
以上是通过一个非常简单的例子来说明Java中的回调的实现。

当然，也可以在事件管理或事件通知者类中，通过注册的方式来注册多个对此事件感兴趣的对象。1. 定义一个接口**InterestingEvent** ，回调方法**nterestingEvent(String event)** 简单接收一个**String** 参数。

interface InterestingEvent {  
    public void interestingEvent(String event);  
}

2. 实现InterestingEvent接口，事件处理类

class CallMe implements InterestingEvent {  
    private String name;  
    public CallMe(String name){  
        this.name = name;  
    }      
    public void interestingEvent(String event) {  
        System.out.println(name + ":[" +event  + "] happened");  
    }  
}

3. 事件管理者，或事件通知者

class EventNotifier {  
    private List<CallMe> callMes = new ArrayList<CallMe>();  
      
    public void regist(CallMe callMe){  
        callMes.add(callMe);  
    }  
      
    public void doWork(){  
        for(CallMe callMe: callMes) {  
            callMe.interestingEvent("sample event");  
        }  
    }      
}

4. 测试

public class CallMeTest {  
    public static void main(String[] args) {  
        EventNotifier ren = new EventNotifier();  
        CallMe a = new CallMe("CallMe A");  
        CallMe b = new CallMe("CallMe B");  
  
        // regiest  
        ren.regist(a);  
        ren.regist(b);  
          
        // test  
        ren.doWork();          
    }  
}

来源:博客园