# 事件的参与者

先抛开Java语言的特殊性，事件机制的开创者们将一件事情的参与者抽象出以下三种角色：

**事件对象（Event Object）**：它是事件发生的“当事人”之一，可以存储某件事情发生时候一些状态的值。比如Java事件中的KeyEvent，它记录了键盘中的哪个建被按下。

**事件监听者（Event Listener）**：它们是专门“围观看热闹”的，但他们并不是一直痴痴地等待事件的发生，而是当事件发生的时候，有“人”会告诉它们，这个负责“通风报信”的人就是事件机制的第三个角色——事件源。

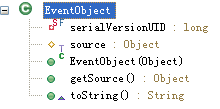
**事件源（Event Source）**：它是事件的“发起人”，比如一个Java中的Button按钮类，它可以激发ActionEvent事件。事件源有一项重要的工作就是当事件发生的时候，告诉所有的事件监听者“XX事件发生了，你们该干嘛干嘛吧”。

注意，正如篇首所言，这三者并不是Java特有的，你手里掌握的所有面向对象语言应该都能实现。

# Java中的事件机制

Java为事件机制中的三个角色中的前两个角色定义了两个父类型，分别是EventObject类和EventListener接口。

## EventObject



EventObject的构造函数接受一个Object对象，Java是希望将第三个角色“事件源”在EventObject中有所记录。“冤有头，债有主”，EventObject记录的第一个状态便是该事件的发起者了。

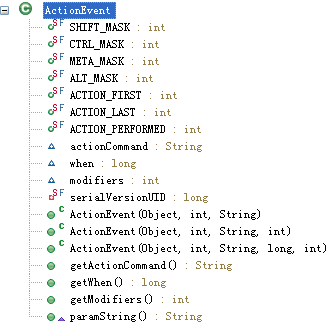
## EventListener

这个接口非常“懒”，什么都没有实现。它是作为一个“标签”而存在的，在Java中，所有的事件监听者都将实现该接口以标明它是一个监听者。

## Java事件举例

Java中的事件太多了，用个比较熟悉的ActionEvent来举例说明吧。

（1）ActionEvent



这个类自然是EventObject的子类，它会记录事件的激发者。同时该类还记录了一个“actionCommand”字符串，按照Java的说法，它是用来更有效的指示是哪个对象激发了该事件，能够补充更多关于“事件源”的信息。

此外，ActionEvent还记录了事件被激发的事件，可以通过getWhen方法获得；记录了事件被激发时有没有一些其它附加的操作，比如“Shift”键被按下、“Ctrl”键被按下等等，可以通过getModifiers方法获得附加操作的掩码。

（2）ActionListener



根据Java的约定，它首先是要实现EventListener接口的。除此之外ActionListener一定是要有一个方法用来对“ActionEvent”的发生出现反应的，否则它就没有存在的意义了。

actionPerformed便是事件发生时“监听者”所执行的反馈动作，这个反馈动作的执行是“事件源”驱动的，驱动的方式就是“事件源”对该方法的调用，在该方法参数列表中拥有一个ActionEvent参数，指示了事件本身，这也应该是“事件源”为其注入的。