**Android Touch 事件分发流程分析**

**▐ 事件分发：public boolean dispatchTouchEvent(MotionEvent ev)**

Touch 事件发生时 Activity 的 dispatchTouchEvent(MotionEvent ev) 方法会以隧道方式（从根元素依次往下传递直到最内层子元素或在中间某一元素中由于某一条件停止传递）将事件传递给最外层 View 的 dispatchTouchEvent(MotionEvent ev) 方法，并由该 View 的 dispatchTouchEvent(MotionEvent ev) 方法对事件进行分发。dispatchTouchEvent 的事件分发逻辑如下：

* 如果 **return true**，事件会分发给当前 View 并由 dispatchTouchEvent 方法进行消费，同时事件会停止向下传递；
* 如果 **return false**，事件分发分为两种情况：

1. 如果当前 View 获取的事件直接来自 Activity，则会将事件返回给 Activity 的 onTouchEvent 进行消费；
2. 如果当前 View 获取的事件来自外层父控件，则会将事件返回给父 View 的  onTouchEvent 进行消费。

* 如果返回系统默认的 **super**.dispatchTouchEvent(ev)，事件会自动的分发给当前 View 的 onInterceptTouchEvent 方法。

**▐ 事件拦截：public boolean onInterceptTouchEvent(MotionEvent ev)**

在外层 View 的 dispatchTouchEvent(MotionEvent ev) 方法返回系统默认的 super.dispatchTouchEvent(ev) 情况下，事件会自动的分发给当前 View 的 onInterceptTouchEvent 方法。onInterceptTouchEvent 的事件拦截逻辑如下：

* 如果 onInterceptTouchEvent 返回 **true**，则表示将事件进行拦截，并将拦截到的事件交由当前 View 的 onTouchEvent 进行处理；
* 如果 onInterceptTouchEvent 返回 **false**，则表示将事件放行，当前 View 上的事件会被传递到子 View 上，再由子 View 的 dispatchTouchEvent 来开始这个事件的分发；
* 如果 onInterceptTouchEvent 返回 **super**.onInterceptTouchEvent(ev)，事件默认会被拦截，并将拦截到的事件交由当前 View 的 onTouchEvent 进行处理。注意这一点，若点击的是非子控件那么情况是像这样，但是如果是如果点击的是子控件，那么则事件还是不会被拦截，继续分发到子控件。

**▐ 事件响应：public boolean onTouchEvent(MotionEvent ev)**

在 dispatchTouchEvent 返回 super.dispatchTouchEvent(ev) 并且 onInterceptTouchEvent 返回 true 或返回 super.onInterceptTouchEvent(ev) 的情况下 onTouchEvent 会被调用。onTouchEvent 的事件响应逻辑如下：

* 如果事件传递到当前 View 的 onTouchEvent 方法，而该方法返回了 false，那么这个事件会从当前 View 向上传递，并且都是由上层 View 的 onTouchEvent 来接收，如果传递到上面的 onTouchEvent 也返回 false，这个事件就会“消失”，而且接收不到下一次事件。
* 如果返回了 true 则会接收并消费该事件。
* 如果返回 super.onTouchEvent(ev) 默认处理事件的逻辑和返回 false 时相同。

到这里，与 Touch 事件相关的三个方法就分析完毕了。下面的内容会通过各种不同的的测试案例来验证上文中三个方法对事件的处理逻辑。

**注意、**onClick事件是在View、ViewGroup的onTouchEvent(MotionEvent ev)里面实现的，如果没有调用父类的super. onTouchEvent(MotionEvent ev)那么这个时候onClick事件是不会响应的。**这点千万要注意**。

TouchListener