

[首页](#) [ANDROID](#) [互联网](#) [杂乱无章](#) [科技资讯](#) [程序员人生](#) [程序员笑话](#) [编程技术](#) [网址导航](#)

3.2 基于回调的事件处理机制

分类 [Android 基础入门教程](#)

本节引言

在3.1中我们对Android中的一个事件处理机制——基于监听的事件处理机制进行了学习,简单的说就是 为我们的事件源(组件)添加一个监听器,然后当用户触发了事件后,交给监听器去处理,根据不同的事件 执行不同的操作;那么基于回调的事件处理机制又是什么样的原理呢?好吧,还有一个问题:你知道 什么是**方法回调**吗?知道吗?相信很多朋友都是了解,但又说不出来吧!好了,带着这些疑问我们 对android事件处理机制中的回调事件处理机制进行解析吧!

1.什么是方法回调?

文字表述:

答:是将功能定义与功能分开的一种手段,一种解耦合的设计思想;在Java中回调是通过接口来实现的,作为一种系统架构,必须要有自己的运行环境,且需要为用户提供实现接口;实现依赖于客户,这样就可以 达到接口统一,实现不同,系统通过在不同的状态下"回调"我们的实现类,从而达到接口和实现的分离!

举个简单例子:

比如:你周五放学回家,你问你老妈煮好饭没,你妈说还没煮;然后你跟她说了:老妈,我看下喜羊羊,你煮好饭叫我哈! **分析:**你和老妈约定了一个接口,你通过这个接口叫老妈煮饭,当饭煮好了的时候,你老妈 又通过这个接口来反馈你,"饭煮好了"!

2.Android回调的事件处理机制详解:

在Android中基于回调的事件处理机制使用场景有两个:

1) 自定义view

当用户在GUI组件上激发某个事件时,组件有自己特定的方法会负责处理该事件 通常用法:继承基本的GUI组件,重写该组件的事件处理方法,即自定义view 注意:在xml布局中使用自定义的view时,需要使用"全限定类名"

Android 基础入门教程(Q群号: 153836263)

- 1.0 Android基础入门教程
 - 1.0.1 2015年最新Android基...
 - 1.1 背景相关与系统架构分析
 - 1.2 开发环境搭建
 - 1.2.1 使用Eclipse + ADT + S...
 - 1.2.2 使用Android Studio开...
 - 1.3 SDK更新不了问题解决
 - 1.4 Genymotion模拟器安装
 - 1.5.1 Git使用教程之本地仓...
 - 1.5.2 Git之使用GitHub搭建...
 - 1.6 .9(九妹)图片怎么玩
 - 1.7 界面原型设计
 - 1.8 工程相关解析(各种文件...
 - 1.9 Android程序签名打包
 - 1.11 反编译APK获取代码&...
 - 2.1 View与ViewGroup的概念
 - 2.2.1 LinearLayout(线性布局)
 - 2.2.2 RelativeLayout(相对布...
 - 2.2.3 TableLayout(表格布局)
 - 2.2.4 FrameLayout(帧布局)
 - 2.2.5 GridLayout(网格布局)
 - 2.2.6 AbsoluteLayout(绝对...
 - 2.3.1 TextView(文本框)详解
 - 2.3.2 EditText(输入框)详解
 - 2.3.3 Button(按钮)与ImageB...
 - 2.3.4 ImageView(图像视图)
 - 2.3.5.RadioButton(单选按钮...
 - 2.3.6 开关按钮ToggleButton...
 - 2.3.7 ProgressBar(进度条)
 - 2.3.8 SeekBar(拖动条)
 - 2.3.9 RatingBar(星级评分条)
 - 2.4.1 ScrollView(滚动条)

常见View组件的回调方法：

android为GUI组件提供了一些事件处理的回调方法,以View为例,有以下几个方法

- ①在该组件上触发屏幕事件: boolean **onTouchEvent**(MotionEvent event);
- ②在该组件上按下某个按钮时: boolean **onKeyDown**(int keyCode, KeyEvent event);
- ③松开组件上的某个按钮时: boolean **onKeyUp**(int keyCode, KeyEvent event);
- ④长按组件某个按钮时: boolean **onKeyLongPress**(int keyCode, KeyEvent event);
- ⑤键盘快捷键事件发生: boolean **onKeyShortcut**(int keyCode, KeyEvent event);
- ⑥在组件上触发轨迹球屏事件: boolean **onTrackballEvent**(MotionEvent event);
- *⑦当组件的焦点发生改变,和前面的6个不同,这个方法只能够在View中重写哦! protected void **onFocusChanged**(boolean gainFocus, int direction, Rect previously FocusedRect)

另外,这了解释下什么是轨迹球,不过用处不大,在以前黑莓的手机上可以看到;当我们浏览网页的时候,可以把轨迹球看作鼠标,不过这样的操作,我们用onTouchEvent就可以解决了,而且不够美观,所以现在用的很好,基本不用,如果你有兴趣想看看的话,可以在原始Android模拟器按f6就可以看到了!

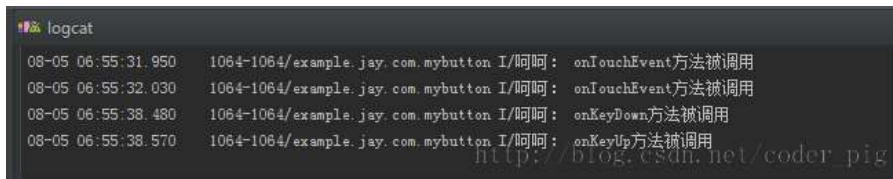


代码示例：我们自定义一个MyButton类继承Button类,然后重写onKeyLongPress方法;接着在xml文件中通过全限定类名调用自定义的view
效果图如下：

- 2.4.2 Date & Time组件(上)
- 2.4.3 Date & Time组件(下)
- 2.4.4 Adapter基础讲解
- 2.4.5 ListView简单实用
- 2.4.6 BaseAdapter优化
- 2.4.7 ListView的焦点问题
- 2.4.8 ListView之checkbox错...
- 2.4.9 ListView的数据更新问题
- 2.5.0 构建一个可复用的自定...
- 2.5.1 ListView Item多布局的...
- 2.5.2 GridView(网格视图)的...
- 2.5.3 Spinner(列表选项框)...
- 2.5.4 AutoCompleteTextVie...
- 2.5.5 ExpandableListView(...
- 2.5.6 ViewFlipper(翻转视图)...
- 2.5.7 Toast(吐司)的基本使用
- 2.5.8 Notification(状态栏通...
- 2.5.9 AlertDialog(对话框)详解
- 2.6.0 其他几种常用对话框基...
- 2.6.1 PopupWindow(悬浮框...
- 2.6.2 菜单(Menu)
- 2.6.3 ViewPager的简单使用
- 2.6.4 DrawerLayout(官方侧...
- 3.1.1 基于监听的事件处理机制
- 3.2 基于回调的事件处理机制
- 3.3 Handler消息传递机制浅析
- 3.4 TouchListener PK OnTo...
- 3.5 监听EditText的内容变化
- 3.6 响应系统设置的事件(Co...
- 3.7 AsyncTask异步任务
- 3.8 Gestures(手势)
- 4.1.1 Activity初学乍练
- 4.1.2 Activity初窥门径
- 4.1.3 Activity登堂入室
- 4.2.1 Service初涉
- 4.2.2 Service进阶
- 4.2.3 Service精通
- 4.3.1 BroadcastReceiver牛...
- 4.3.2 BroadcastReceiver庖...
- 4.4.1 ContentProvider初探



一个简单的按钮,点击按钮后触发onTouchEvent事件,当我们按模拟器上的键盘时,按下触发onKeyDown,离开键盘时触发onKeyUp事件!我们通过Logcat进行查看!



实现代码：MyButton.java

```
public class MyButton extends Button{
    private static String TAG = "呵呵";
    public MyButton(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    //重写键盘按下触发的事件
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        super.onKeyDown(keyCode, event);
        Log.i(TAG, "onKeyDown方法被调用");
        return true;
    }

    //重写弹起键盘触发的事件
    @Override
    public boolean onKeyUp(int keyCode, KeyEvent event) {
        super.onKeyUp(keyCode, event);
        Log.i(TAG, "onKeyUp方法被调用");
        return true;
    }

    //组件被触摸了
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        Log.i(TAG, "onTouchEvent方法被调用");
        return true;
    }
}
```

布局文件：

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲—...

5.2.2 Fragment实例精讲—...

5.2.3 Fragment实例精讲—...

5.2.4 Fragment实例精讲—...

5.2.5 Fragment实例精讲—...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载 (1)

7.3.3 Android 文件下载 (2)

7.4 Android 调用 WebService

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScrip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

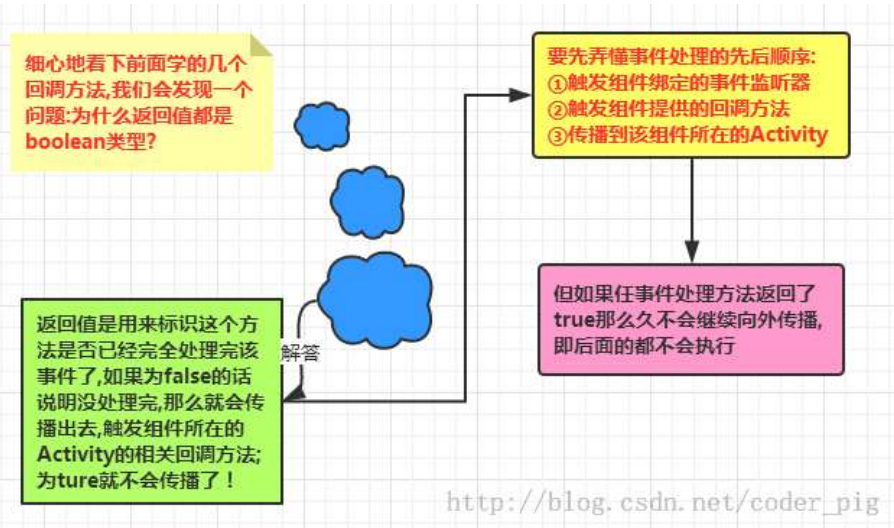
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MyActivity">

    <example.jay.com.mybutton.MyButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮"/>
```

代码解析：

因为我们直接重写了Button的三个回调方法,当发生点击事件后就不需要我们在Java文件中进行 事件监听器的绑定就可以完成回调,即组件会处理对应的事件,即事件由事件源(组件)自身处理！

2) 基于回调的事件传播：



综上,就是如果是否向外传播取决于方法的返回值是时true还是false;

代码示例：

```
public class MyButton extends Button{
    private static String TAG = "呵呵";
    public MyButton(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    //重写键盘按下触发的事件
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        super.onKeyDown(keyCode,event);
        Log.i(TAG, "自定义按钮的onKeyDown方法被调用");
        return false;
    }
}
```

- 8.3.3 Paint API之—— Mask...
- 8.3.4 Paint API之—— Xferm...
- 8.3.5 Paint API之—— Xferm...
- 8.3.6 Paint API之—— Xferm...
- 8.3.7 Paint API之—— Xferm...
- 8.3.8 Paint API之—— Xferm...
- 8.3.9 Paint API之—— Color...
- 8.3.10 Paint API之—— Colo...
- 8.3.11 Paint API之—— Colo...
- 8.3.12 Paint API之—— Path...
- 8.3.13 Paint API之—— Sha...
- 8.3.14 Paint几个枚举/常量值...
- 8.3.15 Paint API之——Type...
- 8.3.16 Canvas API详解(Part 1)
- 8.3.17 Canvas API详解(Part...
- 8.3.18 Canvas API详解(Part...
- 8.4.1 Android动画合集之帧...
- 8.4.2 Android动画合集之补...
- 8.4.3 Android动画合集之属...
- 8.4.4 Android动画合集之属...
- 9.1 使用SoundPool播放音...
- 9.2 MediaPlayer播放音频与...
- 9.3 使用Camera拍照
- 9.4 使用MediaRecord录音
- 10.1 TelephonyManager(电...
- 10.2 SmsManager(短信管理...
- 10.3 AudioManager(音频管...
- 10.4 Vibrator(振动器)
- 10.5 AlarmManager(闹钟服务)
- 10.6 PowerManager(电源服...
- 10.7 WindowManager(窗口...
- 10.8 LayoutInflater(布局服务)
- 10.9 WallpaperManager(壁...
- 10.10 传感器专题(1)——相...
- 10.11 传感器专题(2)——方...
- 10.12 传感器专题(3)——加...
- 10.12 传感器专题(4)——其...
- 10.14 Android GPS初涉
- 11.0 《2015最新Android基...

main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MyActivity">

    <example.jay.com.mybutton.MyButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="自定义按钮"
        android:id="@+id/btn_my"/>

</LinearLayout>
```

MainActivity.java :

```
public class MyActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);

        Button btn = (Button)findViewById(R.id.btn_my);
        btn.setOnKeyListener(new View.OnKeyListener() {
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent
event) {
                if(event.getAction() == KeyEvent.ACTION_DOWN)
                {
                    Log.i("呵呵", "监听器的onKeyDown方法被调用");
                }
                return false;
            }
        });
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        super.onKeyDown(keyCode, event);
        Log.i("呵呵", "Activity的onKeyDown方法被调用");
        return false;
    }
}
```

运行截图：

结果分析：从上面的运行结果,我们就可以知道,传播的顺序是: **监听器**--->**view**组件的回调方法--->**Activity**的回调方法了;

本节小结

本节对Android事件处理机制中的基于回调的事件处理机制进行了讲解！核心就是事件传播的顺序 监听器优先，然后到View组件自身，最后再到Activity；返回值false继续传播，true终止传播~！

在线实例

- [HTML 实例](#)
- [CSS 实例](#)
- [JavaScript 实例](#)
- [Ajax 实例](#)
- [jQuery 实例](#)
- [XML 实例](#)
- [Java 实例](#)

字符集&工具

- [HTML 字符集设置](#)
- [HTML ASCII 字符集](#)
- [HTML ISO-8859-1](#)
- [HTML 实体符号](#)
- [HTML 拾色器](#)
- [JSON 格式化工具](#)

最新更新

- [Swift 正式开源](#)
- [PHP 7 正式发布](#)
- [Shell 编程快速入门](#)
- [Shell 文件包含](#)
- [Shell 输入/输出...](#)
- [Shell printf 命令](#)
- [Shell 基本运算符](#)

消息反馈

- [免责声明](#)
- [关于我们](#)
- [文章归档](#)

关注微信



Copyright © 2013-2015 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1