河北师范大学软件学院
Software College of Hebei Normal University

Android 基础开发

第三章 第一节
Android事件处理

教学目标

- 掌握在Android中两种事件监听器的使用

目录

Android中的事件处理

- Android用户界面的主要作用是：显示视图界面和捕获用户动作。当用户在界面上执行某一个动作时，会触发某一个事件，而Android程序应该对用户动作作出响应，即事件处理。
- 对于Android中的事件处理需要考虑以下问题：
  - 接受事件的是谁？
  - 事件都有什么类型？
  - 谁对事件做出响应？

谁可以接受事件？

- Android中视图界面中可以触发事件的视图元素：
  - 视图组件（TextView、Button、ListView项目、ImageView、……）。
  - 布局容器（LinearLayout、RelativeLayout、……）。

可以接收什么类型的事件？

- Android中可以接收的事件类型：
  - 点击/触摸类事件：onClick、onLongClick、onTouch、onTouchEvent、onTrackballEvent等。
  - 键盘类事件：onKey、onKeyDown、onKeyUp等。
  - 状态改变类事件：onFocusChange、onCheckedChange、onItemSelected。
  - .....

谁来进行事件处理？

- Android中提供了两种方式的事件处理机制：
  - 基于监听器的事件处理机制：
    - 把事件源（触发事件的对象）和事件的处理器（监听器）分离；
    - 实现机制：为View绑定特定的事件监听器。
  - 基于回调的事件处理机制
    - 事件源的事件由自己处理，即事件源和事件处理整合在一起（同一个类中）。
    - 实现机制：重写事件源的特定回调方法。

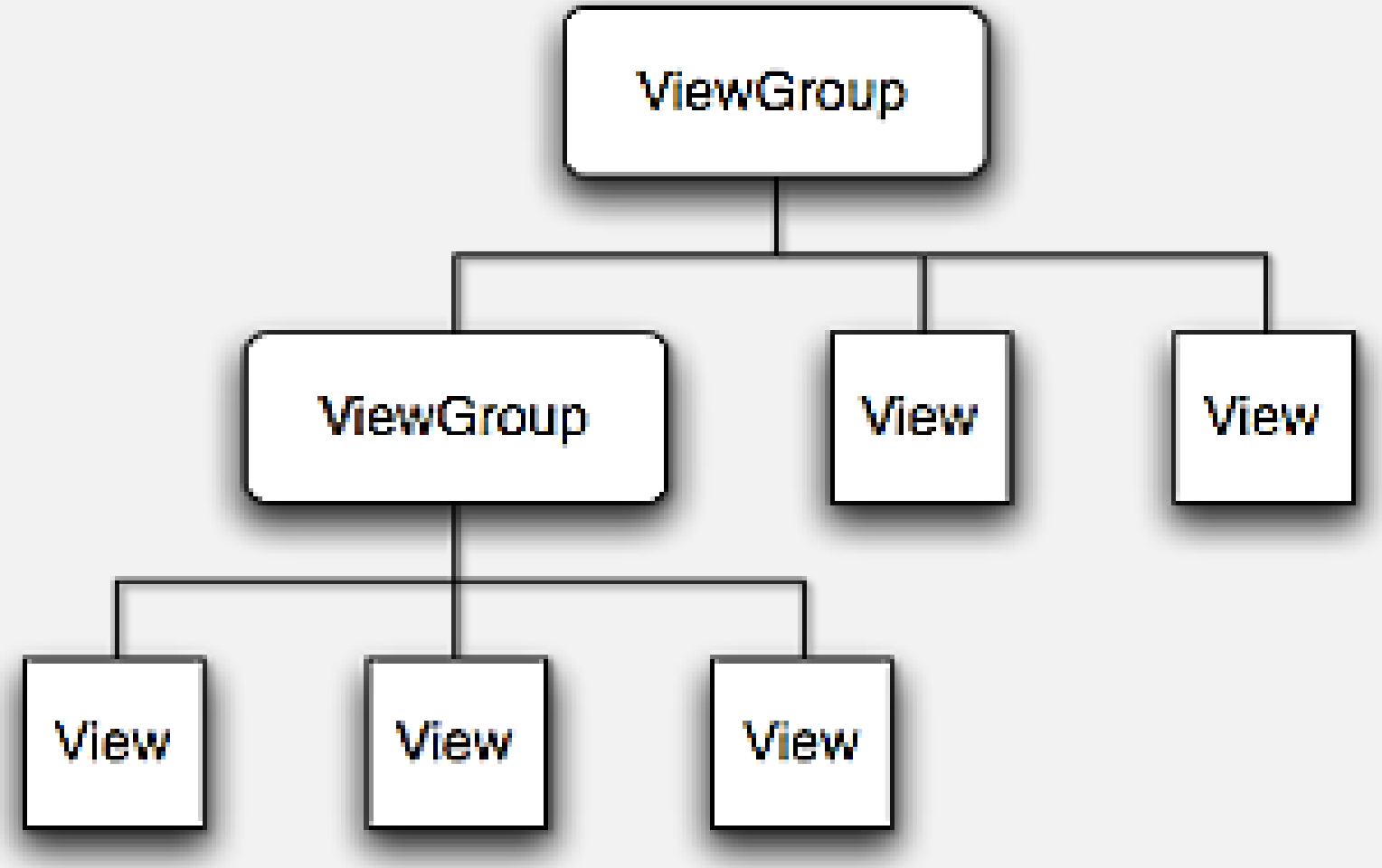
事件传递机制

- 在理解Android事件传递机制之前需要先了解一些基础知识：
  - 事件首先到达Activity的顶层视图。
  - 在事件传递模型中三个基本概念及对应方法：
    - 事件分发（dispatchTouchEvent）；
    - 事件拦截（onInterceptTouchEvent）；
    - 事件响应（onTouchEvent）。
  - ViewGroup会进行事件分发、拦截、响应操作。
  - View只会进行事件分发、响应操作（分发方法直接跳转到响应方法）。

事件传递机制

- 事件分发：
  - 父层到子层（Activity -> ViewGroup -> View），若父层拦截事件，则不再向子层确认。
- 事件回溯：
  - 事件源对象到父层（View->ViewGroup->Activity），若子层已经处理事件，不需要再向父层传播，事件处理结束。

事件传递机制



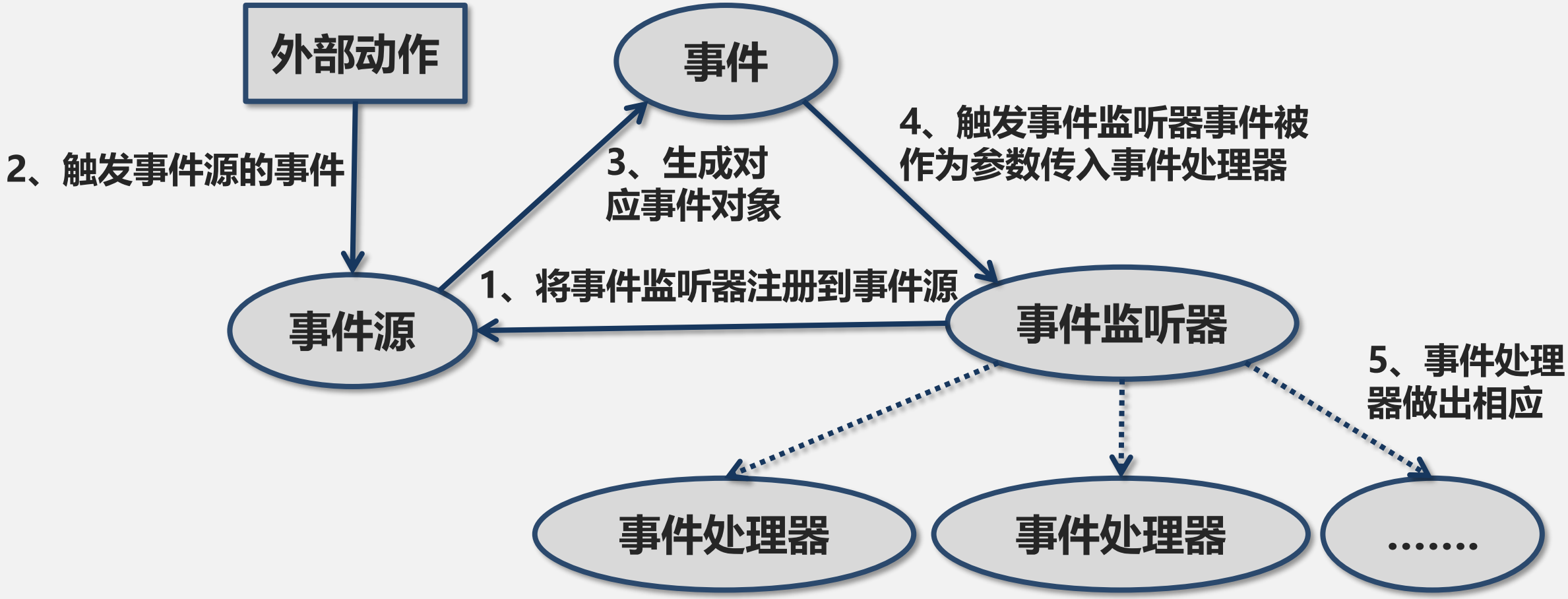
目录

基于监听器的事件处理

- 基于监听器的事件处理把事件的事件源和处理器进行分离，便于处理器处理多个视图触发的事件。
- 在实践监听的处理模型中，涉及到3个核心对象：
  - 事件源：事件发生的场所（即View或ViewGroup）。
  - 事件：封装了界面上发生的特定事情（通常是一次用户操作）；若需要获得事件详细信息，可以访问Event对象。
  - 事件监听器：负责监听事件源所发生的事件，并做出响应。

基于监听器的事件处理

- 基于监听器事件处理的工作机制：



基于监听器的事件处理

- 基于监听器事件处理的实现方法：
  - 匿名内部类对象做为事件监听器；
  - 内部类对象做为事件监听器：最常用的方式；
    - 适用于一个监听器响应多个元素事件的情况。
  - 直接在XML布局文件中绑定监听器：最简单的方式，但不利于代码维护；
  - Activity本身做为事件监听器。

基于监听器的事件处理

- 匿名内部类对象做为事件监听器
  - 在Activity中获取事件源，并绑定监听器

```
Button button = findViewById(R.id.btn_test);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO
    }
});
```

基于监听器的事件处理

- 直接在XML布局文件中绑定监听器
  - 在XML布局标签中绑定监听器

```
<Button android:id="@+id/btn_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="handlerButtonClick" />
```

  - 在Activity中实现监听器处理操作

```
public class MainActivity extends Activity {
    public void handlerButtonClick() {
        // TODO
    }
}
```

基于监听器的事件处理

- 内部类对象做为事件监听器
  - 在Activity中创建内部类

```
class CheckBoxCheckedChangeHandler
        implements CompoundButton.OnCheckedChangeListener {
    public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
        switch (buttonView.getId()) {
            case R.id.cb_book:  ... break;
            case R.id.cb_music: ... break;
        }
    }
}
```

基于监听器的事件处理

- 内部类对象做为事件监听器
  - 在Activity中获取事件源，分别绑定事件监听器

```
CheckBox cbBook = findViewById(R.id.cb_book);
CheckBox cbMusic = findViewById(R.id.cb_music);
cbBook.setOnCheckedChangeListener(
        new CheckBoxCheckedChangeHandler());
cbMusic.setOnCheckedChangeListener(
        new CheckBoxCheckedChangeHandler());
```

基于监听器的事件处理

- Activity本身做为事件监听器
  - Activity类应该实现相应事件接口

```
public class MainActivity extends FragmentActivity
        implements View.OnClickListener {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = findViewById(R.id.btn_test);
        button.setOnClickListener(this);
    }
    public void onClick(View v) { // TODO }
}
```

基于监听器的事件处理

- 学习监听器事件处理的三种方法
  - 重点掌握匿名内部类和内部类对象的方法
- 学习键盘事件
  - 掌握键盘事件onKey、onKeyDown、onKeyUp的区别
  - 掌握事件对象KeyEvent的使用

目录

基于回调的事件处理

- 基于回调的事件处理机制，事件源与事件监听器是统一的（或者说，事件监听器存在于事件源View对象中），一般适用于某个视图组件绑定特定行为中。
- 为了实现回调机制，Android为所有的UI组件提供了一系列事件处理方法，如View组件：
  - onKeyUp、onKeyDown、onKeyLongPress等
  - onTouchEvent、onTrackballEvent等
  - ……（具体参考Android开发文档）

基于回调的事件处理

- 基于回调的事件处理机制的使用方法：
  - 自定义视图类，并添加事件回调函数

```
public class MyButton extends Button {
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN: // TODO
                break;
            ......
        }
        return super.onTouchEvent(event);
    }
}
```

基于回调的事件处理

- 学习基于回调的事件处理方法
  - 掌握自定义视图类的使用方法
    - 自定义视图类
    - 在XML布局文件中使用自定义视图类
  - 掌握基于回调的事件处理方法
    - 学习onTouchEvent事件的使用（事件对象的常用方法和常用属性）

基于回调的事件传播

- 几乎所有的事件处理方法都有一个boolean类型的返回值，用于标示是否完全处理该事件：
  - 如果返回true，表明已经完全处理该事件，不会再传播
  - 如果返回false，表明未完全处理该事件，会在传播

事件处理机制汇总

- 对比两种事件处理模型，两种各有优势：
  - 基于监听的事件模型分工更明确，事件源与处理器分开具有更好的维护性
  - 事件处理机制确保事件监听器会被优先触发

内容回顾

1. 事件处理概述
2. 基于监听器的事件处理
3. 基于回调的事件处理

Thank you!