Android移动开发

Android Mobile Application Development

第5讲 Event事件

吴以凡
计算机学院 一教505
yfwu@hdu.edu.cn

事件Event

Android事件处理机制

- Android提供了两种方式的事件处理：
  - 基于监听器的事件处理：为Android界面组件绑定特定的事件监听器
  - 基于回调的事件处理：重写Android组件特定的回调函数

Android事件处理机制

- 与基于回调的事件处理相比，基于监听的事件处理属于更具“面向对象”性质的事件处理方式
- 在监听器模型中，主要涉及三类对象：
  - 事件源（Event Source）：产生事件的来源，通常是各种组件，如按钮等
  - 事件（Event）：封装了界面组件上发生的特定事件的具体信息
  - 事件监听器（Event Listener）：负责监听事件源发生的事件，并对不同的事件做相应的处理

基于监听的事件处理

- 基于监听的事件处理机制的步骤
  - 1. 获取普通界面的组件即事件源
    - 布局文件中设置组件：

```
<Button
    android:id="@+id/btn1"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:text="获取Ediext内容"/>
</LinearLayout>
```

基于监听的事件处理

- 基于监听的事件处理机制的步骤
  - 2.实现事件的监听器类，该监听器类是一个特殊的Java类，必须实现一个XXXListener接口

```
public class MainActivity extends Activity
implements View.OnClickListener {
    EditText editText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
}
```

基于监听的事件处理

- 基于监听的事件处理机制的步骤
  - 3.实现监听器类必须实现的onXXX()方法，该方法将会作为事件处理器

```
public class MainActivity extends Activity
implements View.OnClickListener {
    EditText editText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        …
        @Override
        public void onClick(View view) {
            …
        }
```

基于监听的事件处理

- 基于监听的事件处理机制的步骤
  - 4.调用事件源的setXxxListener方法，将事件监听器对象注册给事件源

```
Button btn1=(Button)findViewById(R.id.btn1);
btn1.setOnClickListener(this); // 为按钮绑定事件监听器
```

当事件源上发生指定事件时，Android会触发事件监听器，由事件监听器调用相应的onXXX()方法来处理事件。

基于回调的事件处理

- 基于回调的事件处理模型中，事件源和事件监听器是合一的，没有独立的事件监听器存在
- 当用户在GUI组件上触发某事件时，由该组件自身特定的函数负责处理该事件。通常通过重写（Override）组件类的事件处理函数实现事件的处理
- 为了使用回调机制类处理GUI组件上所发生的事件，需要通过继承GUI组件类，并重写该类的事件处理方法来实现

基于回调的事件处理

- Android为所有的GUI组件都提供了一些事件处理的回调方法
- 例如，View类包含如下方法：
  - boolean onKeyDown(int keycode,KeyEvent event)用户在该组件上按下某个按键时触发的方法。
  - boolean onKeyLongPress(int keycode,KeyEvent event)用户在该组件上长按某个组件时触发的方法。
  - boolean onKeyUp(int keycode,KeyEvent event)用户在该组件上松开某个按键时触发的方法。

基于回调的事件处理

- 示例：自定义按钮
  - 定义一个TestButton类继承Button类，重写该类的onTouchEvent方法来负责处理按钮上的键盘事件

```
public class TestButton extends Button {
   public testButton(Context context, AttributeSet attrs) {
      super(context, attrs);
   }
   /* 重写 onTouchEvent触碰事件的回调方法 */
   @Override
   public boolean onTouchEvent(MotionEvent event) {
      Log.i("测试CallBack", "我是TestButton，你触碰了我: " + event.getAction());
      Toast.makeText(getContext(), "我是TestButton，你触碰了我: " +
                        event.getAction(), Toast.LENGTH_SHORT).show();
      return true; //返回true，表示事件不会向外层(即父容器)扩散
   }
}
```

基于回调的事件处理

- 示例：自定义按钮
  - 布局文件中使用了这个自定义TestButton

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   tools:context="com.jxust.cn.chapter4_callback.MainActivity">
   <com.hdu.ccourse.callback_example.TestButton
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textSize="18sp"
      android:text="测试基于回调的事件处理机制"/>
</LinearLayout>
```

Activity中View的单击事件

- 4种基于监听的方法实现View的单击事件
  - 布局文件中设置android:onClick属性为Activity中的某个方法
  - 主Activity类实现OnClickListener接口，重写onClick()方法，通过switch()语句判断哪个按钮被点击
  - 创建onClickListener的匿名内部类传入按钮的setOnClickListener()参数中，适合按钮比较少的情况下使用
  - 创建内部类实现OnClickListener接口并重写其onClick()方法

Activity中View的单击事件

- 布局文件中设置android:onClick属性

```
<!--布局文件中添加点击事件为其制定方法名-->
<Button android:onClick="myClick">
```

```
public void myClick(View view){
    Intent intent=new Intent(MainActivity.this, SecondActivity.class);
    startActivity(intent);
}
```

Activity中View的单击事件

- 主Activity类实现OnClickListener接口

```
public class MainActivity extends Activity implements View.OnClickListener {
   register = (Button)findViewById(R.id.register);
   register.setOnClickListener(this);
      @Override
      public void onClick(View view) {
         switch (view.getId()){
            case R.id.register:
            break;
         }
      }
   ...
}
```

Activity中View的单击事件

- 创建onClickListener的匿名内部类
  - 缺点：临时使用一次,复用性不高

```
protected void onCreate(Bundle savedInstanceState) {
   Button button=(Button)findViewById(R.id.button);
   button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
          ...
       }
   });
}
```

Activity中View的单击事件

- 创建内部类实现OnClickListener接口并重写其onClick()方法
  - 优点:可以在该Activity中进行复用,可直接访问Activity类的所有控件

```
public class MainActivity extends Activity {
   private Button btnshow;
   protected void onCreate(Bundle savedInstanceState) {
      btnshow = (Button) findViewById(R.id.btnshow);
      //直接new一个内部类对象作为参数
      btnshow.setOnClickListener(new BtnClickListener());
   }
   //定义一个内部类,实现View.OnClickListener接口,并重写onClick()方法
   class BtnClickListener implements View.OnClickListener
   {
      @Override
      public void onClick(View v) {    ...     }
   }
}
```