事件

监听器

View提供了丰富的监听器接口，用来处理和用户的交互。

- OnClickListener - 点击事件
- OnLongClickListener - 长按事件
- OnTouchListener - 触摸事件，可以处理点击、滑动、长按等操作
- OnFocusChangeListener - 焦点改变事件
- OnKeyListener - 按键事件
- OnDragListener - 拖拽事件
- OnHoverListener - 悬停事件

回调

触摸事件

跟`OnTouchListener`一样，`View`提供了`onTouchEvent()`方法来处理触摸事件。

```
public class MyView extends View {
    public MyView(Context context) {
        super(context);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        // 处理触摸事件
        Log.v("MyView", event.toString());
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                // 按下
                break;
            case MotionEvent.ACTION_MOVE:
                // 移动
                break;
            case MotionEvent.ACTION_UP:
                // 抬起
                break;
        }
        return true;
    }
}
```

键盘事件

- 键盘回调函数
  - `onKeyDown()` - 按键按下
  - `onKeyUp()` - 按键抬起
- 键盘回调触发，需要`View`能获取焦点，调用`setFocusable()`方法设置。

```
public class MyView extends View {
    public MyView(Context context) {
        super(context);
        // 设置获取焦点
        setFocusable(true);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
```

```
        // 处理按键按下事件
        Log.v("MyView", "onKeyDown" + keyCode + " " + event.toString());
        return super.onKeyDown(keyCode, event);
    }

    @Override
    public boolean onKeyUp(int keyCode, KeyEvent event) {
        // 处理按键抬起事件
        Log.v("MyView", "onKeyUp" + keyCode + " " + event.toString());
        return super.onKeyUp(keyCode, event);
    }
}
```

消息队列（Handler）

- `Handler`是Android中处理异步消息的工具，它可以将一个任务切换到主线程中运行。
- `Handler`可以发送和处理消息，也可以执行延时任务。
- `Handler`是线程和主线程之间的桥梁，它可以将任务切换到主线程中执行，从而避免了在子线程中进行UI操作的问题。
- `Handler`的使用步骤：
    - 创建`Handler`对象
    - 重写`handleMessage()`方法，处理接收到的消息
    - 使用`sendEmptyMessage()`或`sendMessage()`方法发送消息

```
public class MyView extends View {
    private Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            // 处理接收到的消息
            Log.v("MyView", "handleMessage" + msg.what);
        }
    }

    public MyView(Context context) {
        super(context);
    }

    public void doSomething() {
        // 执行耗时操作
        new Thread(new Runnable() {
            @Override
            public void run() {
                // 执行耗时操作
                // 发送消息
                handler.sendEmptyMessage(0);
            }
        }).start();
    }
}
```

Intent

- `Intent`是Android中用于启动Activity、启动Service、发送广播等操作的工具。
- `Intent`可以携带数据，也可以指定目标组件。

创建和启动`Activity`活动

启动活动有很多方式：

- 使用`startActivity()`方法启动活动
- 使用`startActivityForResult()`方法启动活动并等待返回结果
    - 新活动关闭后，当前活动会收到`onActivityResult()`方法的回调
- 使用`startActivityIfNeeded()`方法启动活动并等待返回结果
    - 新活动关闭后，当前活动会收到`onActivityResult()`方法的回调
    - 如果新活动已经存在，则不会重新创建新活动，而是直接使用已存在的活动

###创建活动

- 创建一个新的活动类，继承自`Activity`或其子类

```
public class MyActivity extends AppCompatActivity {

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
    }
}
```

- 在AndroidManifest.xml文件中注册活动

```
<activity android:name=".MyActivity" />
```

使用startActivity()方法启动活动

```
Intent intent = new Intent(this, MyActivity.class);
startActivity(intent);
```

使用startActivityForResult()方法启动活动并等待返回结果

- 启动活动并等待返回结果

```
Intent intent = new Intent(this, MyActivity.class);
startActivityForResult(intent, 1);
```

- 创建活动并返回结果

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);

        // 按钮点击事件
        findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 返回结果
                Intent intent = new Intent();
                intent.putExtra("key", "value");
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }
}
```

- 处理返回结果

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1 && resultCode == RESULT_OK) {
        // 处理返回结果
        String value = data.getStringExtra("key");
        Log.v("MyActivity", "requestCode: " + requestCode + " resultCode: " + resultCode + " value: " + value);
        }
    }
}
```

数据传输

传输数据有两种方式：

- 使用Intent携带数据
- 使用Bundle携带数据

使用Intent携带数据

- 发送数据

```
Intent intent = new Intent(this, MyActivity.class);
intent.putExtra("key", "value");
startActivity(intent);
```

- 接收数据

```
// 获取Intent
Intent intent = getIntent();
// 获取数据
String value = intent.getStringExtra("key");
```

使用Bundle携带数据

- 发送数据

```
Intent intent = new Intent(this, MyActivity.class);
// 存入数据
Bundle bundle = new Bundle();
bundle.putString("key", "value");
intent.putExtras(bundle);
startActivity(intent);
```

- 接收数据

```
// 获取Intent
Intent intent = getIntent();
// 获取数据
Bundle bundle = intent.getExtras();
String value = bundle.getString("key");
```

隐式Intent

隐式启动活动，不需要指定目标组件，只需要指定动作和数据类型，系统会根据动作和数据类型自动匹配目标组件，并启动活动，可以实现跨应用的组件启动。

启动其他应用的活动

- 打开浏览器

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://baidu.com"));
startActivity(intent);
```

- 打开拨号器

```
Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:10086"));
startActivity(intent);
```

- 发送短信

```
Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.parse("smsto:10086"));
intent.putExtra("sms_body", "Hello World");
startActivity(intent);
```

创建隐式方式启动的活动

- 创建一个浏览器活动

```
public class BrowserActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_browser);

        // 获取Intent
        Intent intent = getIntent();
        // 获取数据
        String url = intent.getData().toString();
        // 加载网页
        WebView webView = findViewById(R.id.webView);
        webView.loadUrl(url);
    }
}
```

- 在AndroidManifest.xml文件中注册活动

```
<activity android:name=".BrowserActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="http" />
        <data android:scheme="https" />
    </intent-filter>
</activity>
```

- 启动自定义的浏览器活动

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://baidu.com"));
startActivity(intent);
```